5-2007

Intrusion detection and response model for mobile ad hoc networks.

Sathishkumar Alampalayam 1974-
University of Louisville

INTRUSION DETECTION AND RESPONSE MODEL FOR MOBILE AD HOC NETWORKS

by

Sathish Kumar Alampalayam P.
B.E., Bharathiar University, 1996
M.S., University of Louisville, 2001
M.B.A., University of Louisville, 2001

A Dissertation
Submitted to the faculty of the
Graduate School of the University of Louisville
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2007

INTRUSION DETECTION AND RESPONSE MODEL FOR MOBILE AD HOC NETWORKS

by

Sathish Kumar Alampalayam P
B.E., Bharathiar University, 1996
M.S., University of Louisville, 2001
M.B.A., University of Louisville, 2001

A Dissertation approved on

_____

(Date)

By the following Reading Committee:

_____

Dissertation Director, Prof. Anup Kumar

_____

Dissertation Associate Director, Prof. S.Srinivasan

_____

Prof. Rammohan K. Ragade

_____

Prof. Mehmed M. Kantardzic

_____

Prof. Julius P. Wong

_____

Prof. Dar-Jen Chang

DEDICATION

The dissertation is dedicated to

My Parents, My Teachers, Almighty, My Wife and My Son.

ACKNOWLEDGEMENTS

The author would like to thank his dissertation advisor, Prof. Anup Kumar, for his patience, guidance, encouragement, advice and constant moral support throughout the course of his graduate studies and dissertation work. The author is grateful to Prof. Kumar for letting him continue his professional career in the industry, while pursuing his graduate studies and research. His insightful comments and suggestions led him to a better understanding of the scope for his research.

The author also conveys his sincere appreciation to Prof. S.Srinivasan for his patience, suggestions, and also for providing valuable comments that helped to improve the contents of dissertation. The author would also like to thank Prof. Rammohan K. Ragade, for his guidance, advice and constant moral support throughout the course of his graduate studies and research. The author also wishes to thank Prof. Mehmed M. Kantardzic, Prof. Julius P. Wong and Prof. Dar-Jen Cheng for their comments to improve the dissertation contents and serving on the dissertation committee.

The author also wishes to thank George Abraham and Bin Xie for their help in the simulation of ad hoc network using NS2 under Linux environment.

Finally, the author would like to express thanks to his colleagues, friends and relatives for their love, patience and understanding while pursuing his graduate studies and research.

ABSTRACT

INTRUSION DETECTION AND RESPONSE MODEL FOR MOBILE AD HOC NETWORKS

Sathish Kumar Alampalyam. P

May 12' 2007

This dissertation presents a research whose objective is to design and develop an intrusion detection and response model for Mobile Ad hoc NETworks (MANET). Mobile ad hoc networks are infrastructure-free, pervasive and ubiquitous in nature, without any centralized authority. These unique MANET characteristics present several changes to secure them.

The proposed security model is called the Intrusion Detection and Response for Mobile Ad hoc Networks (IDRMAN). The goal of the proposed model is to provide a security framework that will detect various attacks and take appropriate measures to control the attack automatically. This model is based on identifying critical system parameters of a MANET that are affected by various types of attacks, and continuously monitoring the values of these parameters to detect and respond to attacks.

This dissertation explains the design and development of the detection framework and the response framework of the IDRMAN. The main aspects of the detection framework are data mining using CART to identify attack sensitive network parameters from the wealth of raw network data, statistical processing using six sigma to identify the thresholds for the attack sensitive parameters and quantification of the MANET node state through a measure called the Threat Index (TI) using fuzzy logic

methodology. The main aspects of the response framework are intruder identification and intruder isolation through response action plans.

The effectiveness of the detection and response framework is mathematically analyzed using probability techniques. The detection framework is also evaluated by performance comparison experiments with related models, and through performance evaluation experiments from scalability perspective. Performance metrics used for assessing the detection aspect of the proposed model are detection rate and false positive rate at different node mobility speed. Performance evaluation experiments for scalability are with respect to the size of the MANET, where more and more mobile nodes are added into the MANET at varied mobility speed. The results of both the mathematical analysis and the performance evaluation experiments demonstrate that the IDRMAN model is an effective and viable security model for MANET.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

FIGURE PAGE

FIGURE PAGE

CHAPTER I

INTRODUCTION

1.1 Background

A Mobile Ad hoc Network (MANET) is a collection of wireless mobile nodes forming a temporary network without any established infrastructure or centralized authority. In a MANET, each wireless mobile node operates not only as an end-system, but also as a router to forward packets. The nodes are free to move about and organize themselves into a network. MANET does not require any fixed infrastructure such as base stations; therefore, it is an attractive networking option for connecting mobile devices quickly and spontaneously. For instance, first responders at a disaster site or soldiers in a battlefield must provide their own communications. A MANET is a possible solution for this need to quickly establish communications in a mobile, transient and infrastructure-less environment. This is one of many applications where MANET's can be used. Mobile ad-hoc networks are the future of wireless networks. Nodes in these networks will generate both user and application traffic and perform various network functions.

In the last decade, wired and wireless computer network revolution has changed the computing scenario. The possibilities and opportunities due to this revolution are limitless; unfortunately, so too are the risks and chances of attacks due to intrusion by malicious nodes [19]. Intrusion is defined as an attack or a deliberate unauthorized

attempt to access information, manipulate information, or render a system unreliable or unusable [78]. According to [24], Threat can be defined as "the potential possibility of a deliberate unauthorized attempt to a)access information, b) manipulate information and c) render a system unreliable or unusable". Authentication mechanisms are designed to protect a system from unauthorized access to its resources and data. However, at present, completely preventing breaches of security seems unrealistic, especially in cellular Internet, wireless and mobile ad hoc network [8, 20]. A Personal Area Network (PAN) level firewall as envisioned for the next generation wireless networks can protect only if the users are at home and not when the users are roaming [9]. Even if such a firewall is provided, the communication would get fragmented by these 'check points' on the network, as each firewall needs maintenance of activities like log control, software update etc., creating unnecessary overhead. Thus existing technologies like firewalls and Virtual Private Network (VPN) sandboxes cannot be directly applied to the wireless mobile world. Even if the firewall concept were achieved by creating a private extranet (VPN) which extends the firewall protected domain to wherever the user moves, this would still lead to inefficient routing. Security is a fundamental concern for mobile network based system. Harrison et al [7] identify security as a "severe concern" and regard it as the primary obstacle to adopting mobile systems. Until recently, the main research focus has been on improving the protocols for multi-hop routing, performance and scalability of the ad hoc networks [1]. Though the performance and scalability have their place in wireless MANET research, the current and future applications of the ad hoc networks has forced the research community to look at dependability and security aspects of ad hoc networks.

By security we mean protecting nodes from damages due to either voluntary or accidental attacks [60]. This protection is provided by predicting an attack by monitoring a set of metrics measured from the ad hoc network, and then responding and modifying the security of the network based on the vulnerability level at a given time. Intrusion detection is the process of detecting and responding to malicious activity that is aimed at attacking the network [24, 25]. Several techniques for detecting intrusions have been studied [56, 75, 79]. Network firewalls often implement security policies at the front-end to protect computers from malicious attacks [33]. Due to the sophistication of hackers and attacks from the insiders, regular intrusion prevention techniques like firewalls, encryption, digital signature and authentication fail to detect various attacks. The objective of the IDA is to provide another layer of defense not only for authentication related attacks but also for sophisticated and internal attacks. An overview of the existing IDA techniques can be found in [24, 25, 81]. There are several weaknesses in the current IDA applied to MANET [22]. In spite of successful invention of technologies in the field of security and cryptography to secure computer network systems, malicious users still succeed in attacking the network with devastating effects.

Security in mobile ad hoc network is essential even for basic network functions like routing which are carried out by the nodes themselves rather than specialized routers. The intruder in the ad hoc network can come from anywhere, along any direction, and target any communication channel in the network. Compare this with a wired network where the intruder gains physical access to the wired link or can pass through security holes at firewalls and routers. Since the infrastructure-free mobile ad hoc network does not have a clear line of defense, every node must be prepared for the adversary. The

centralized or hierarchical network security solution for the existing wired and infrastructure-based cellular wireless networks will not work properly for mobile ad hoc networks [1]. Securing the ad hoc networks, like any other field of computers, is based on the principle of confidentiality and integrity. These principles exist in every field, but the presence of malicious nodes, selfish nodes, covert channels and eavesdroppers in the mobile ad hoc network makes this an extremely important and challenging problem [2]. In the past several years, there has been a surge of network security research in the field of information assurance that has focused on protecting the network using techniques such as authentication and encryption. These techniques are applicable in the wired and infrastructure-based cellular network. In the case of infrastructure-free mobile ad hoc networks these techniques are not applicable [1]. In the infrastructure-free networks, the nodes themselves perform basic network functions like routing and packet forwarding. Therefore, mobile ad hoc network security is a pressing issue, which needs immediate research attention [3, 4, 5, 6]. Providing security services in the mobile computing environment is challenging because it is more vulnerable for intrusion and eavesdropping. The challenge of mobile ad hoc network security has attracted several researchers with the aim of securing mobile ad hoc computer networks. Intrusion Detection Approach (IDA) is thus an active research area in the field of mobile ad hoc network security [10-18, 24, 25, 63, 65, 68, 70].

1.2 Security Attacks in MANET

A MANET can be subjected to active attacks and passive attacks. Active attacks refer to the direct attacks by a hostile entity during execution or transmission phase. Routing attacks where malicious nodes attack legitimate nodes by means of routing data modification and routing table deletion/forging are examples of active attacks. Passive attacks refer to the indirect attacks by an entity in the network during collaboration. Some of the examples of passive attacks include actions like selfishness, eavesdropping and traffic analysis. Attacks like Denial of Service (DoS) can be either active or passive attacks depending on the nature of attack and the malicious node. In this section we describe some of these attacks.

Active attack in MANET – Routing attack:

Routing attack is a significant problem because nodes within the ad hoc network themselves performs routing functions and the security concepts are not incorporated in most of the routing protocols. Also, routing tables form the basis of network operations and any corruption to the routing table may lead to significant adverse consequences.

Designing a secure ad hoc network routing protocol is a challenge for the following reasons: Firstly, routing relies on the trustworthiness of all the nodes involved and it is difficult to distinguish selfish nodes from normal nodes. Secondly, rapid mobility of nodes that perform the role of routing and network topology makes the design of a secure routing protocol more difficult. Active routing attacks differ in their behavior depending on the nature of the routing protocol. In the case of link-state routing protocol, a router sends information about its neighbors. Hence a malicious router can send incorrect updates about its neighbors, or remain silent if the link state of the neighbor has

actually changed. However, in the case of distance-vector protocols, routers can send wrong and potentially dangerous updates regarding any nodes in the network, since the nodes do not have the full network topology. These attacks in case of both link-state and distance-vector protocols are very difficult to prevent if the routers exhibit Byzantine faults [35].

Routing attacks on the mobile ad hoc networks can be reactive routing protocol attacks or proactive routing protocol attacks based on the type of protocol used for routing. In reactive routing protocol, like on-demand routing protocol, a node attempts to discover a route to a destination only when it has a packet to send to that destination. In proactive routing protocol, like table-driven routing protocol, entries in the table are updated periodically to perform routing. Since both reactive and proactive routing protocols exhibit different characteristics in state information, exchange and route computation, they are exposed to different types of vulnerabilities, which provide unique set of challenges for securing them. In the next section, we identify different types of routing protocol attacks in the mobile ad hoc networks.

The routing attacks can be classified into two general categories: resource-disruption and resource-consumption attacks. In resource-disruption attack, the attacker attempts to cause legitimate data packets to be routed in dysfunctional ways. In a resource-consumption attack, the attacker attempts to consume valuable network resources, like bandwidth, power or storage. Some of the important and common methods of routing attacks are:

Router Protocol Poisoning: In this attack an intruder causes the disruption by poisoning the routing protocol. Securing these attacks is important because the routing protocol

forms the basis of network operations, and any corruption of the protocol may lead to significant consequences. These attacks on the mobile ad hoc networks can lead to looping, congestion, sub optimal routing and partitioning [36]. Thus, they can ultimately affect the performance of an ad hoc network.

Injecting incorrect information in the routing table: In this type of routing attack, malicious nodes or an intruder would inject incorrect routing information, which in turn would poison the routing tables. These attacks would result in the artificial partitioning of the network, and the hosts residing in one partition would not be able to communicate with hosts residing in the other partition.

Routing Loop Attacks: In this attack, intruder or malicious nodes update the routing table to create a loop, so that packets can traverse in the network without reaching the destination, thereby conserving energy and bandwidth.

Passive attack in MANET – Selfishness:

Passive attacks could be caused by selfishness, eavesdropping and traffic analysis. In this section we explain selfishness attacks to give an idea of passive attacks. In the selfishness attacks, the selfish node abuses constrained resources, such as battery power, for its own benefit [38]. They do not intend to directly damage other nodes in the network. Attackers may also get hold of a node and modify its behavior to make it malicious, so the node would perform selfish attacks in need of resources [6]. These attacks have limited effectiveness compared to the routing-table "poisoning" and DoS attacks [37]. This is because, the attacks are limited to a part of the network rather than the whole network as in the case of routing protocol attacks.

Some of the common types of selfish node attacks in mobile ad hoc network are packet mistreatment and energy consumption attacks. In this kind of attack, a node in mobile ad hoc network does not perform the expected network functions, like packet forwarding or routing, and later claims that the transaction or communication never took place [37]. It could be deliberate or accidental, due to false repudiation of a transaction or due to scarce resources in the mobile ad hoc networks.

Packet mistreatment or interception: In this kind of attack, a selfish node does not perform the function of packet forwarding. As mentioned earlier, interruption of packets may reduce the overall throughput of the network. In a specialized form of packet discarding, selfish nodes do not forward the packets to host destination, but to itself. This result in black hole and DoS attacks.

Energy consumption: In this kind of attacks, nodes try to save significant battery power by not performing networking functions such as routing. This is due to the fact that in ad hoc network most of the energy is consumed by routing of packets. For instance, experiments have shown that if the average hop from source to destination is 5, approximately 80% of the available energy is spent in sending packets from source to destination by packet forwarding [37].

Denial of Service Attack:

In this attack, a malicious node or attacker could spam other nodes causing resource constraints by repeatedly sending packets to another node and may place undue burden on the packet handling routines of the recipient. Nodes can also intentionally distribute false or useless information to prevent hosts from completing their tasks correctly or in a timely manner. DoS attacks are very easy to generate but are very

difficult to detect and hence they are popular type of attack for the hackers. In a typical DoS attack, the attacker node spoofs its IP address and uses multiple intermediate nodes to overwhelm other nodes with traffic. DoS attacks are typically used to take important hosts out of action for a few hours, resulting in unavailability of service for all the users served by the host. It can also be used to disrupt the services of the intermediate routers.

Generally MANET DoS attacks can be categorized into two main types: (a) active and (b) passive attacks.

Active DoS attack can be defined as the direct denial of service attacks on a node by another hostile node through packet flooding, packet modification, deletion or forging of packets or routing table. Following are some of the common types of active DoS attacks by selfish nodes or adversaries: replay of expired routing information, bogus nodes create traffic by bombarding the neighboring nodes with the packets, radio jamming, flooding centralized resource with the requests, ability to change routing protocol to operate as the user wants, Byzantine failure, sleep deprivation torture (Battery Exhaustion) and injecting incorrect routing information.

Passive DoS attack can be defined as the indirect denial of service attacks on an entity by another hostile entity during its execution or transmission. These types of attacks are possible when a malicious or selfish node does not perform the requested function like routing or packet transfer. Some of the other common types of passive DoS attacks are "gray hole" attacks, "idle status" attack and snooping. In gray hole attacks, the selfish node selectively discard the packets that it needs to forward. The snooping attacks can occur from an eavesdropper who can decipher and mistreat the transmitted information. In "idle status" passive attack, the selfish node enters into idle status most of the time,

such that the neighbors are not even aware of its existence. Only when the node wishes to communicate with other nodes, it starts the routing protocol. Though the "idle status" behavior could be legitimate behavior, such a selfish node may not adequately contribute to the network.

1.3 Limitations of Intrusion Prevention

Encryption and Authentication are the existing intrusion prevention methods. Intrusion preventive measures such as encryption and authentication can reduce intrusion but not eliminate them [18]. Encryption and authentication can defend only the normal nodes in MANET but not the malicious nodes or compromised mobile nodes, which most of the time carry private keys [18].

Zhang et al., have explained the following case to illustrate how in spite of several intrusion prevention measures like authentication/encryption there are always some weak links that one can exploit to break in. In Summer 2001, an Internet worm called 'Code Red' has been found to affect Windows-based servers. Organizations rely on firewalls to protect their Intranet from these Internet based worms. However, the Code Red worm has been detected within the Intranet. This is due to the reason that more and more mobile business persons and researchers carry laptops and they use wireless Internet access when they congregate in the public venues (e.g. conferences). Another example to strengthen this case: in a recent IETF meeting it was detected that at least 10% of the laptops were infected with the 'Code Red' worm. When these laptops are integrated back in to their organization, one can imagine the impact to their Intranet by these viruses and worms thus defeating the purpose of firewalls.

This shows that encryption and authentication cannot defend against compromised nodes and the fact that such nodes already carry private keys makes the network even more vulnerable. The dynamic nature of the ad hoc network also means that trust between nodes in the network is virtually non-existent. Without trust, preventive measures are unproductive and measures that rely on a certain level of trust between nodes are susceptible to attacks themselves. Hence there is the need for intrusion detection as it provides a second line of defense.

1.4 Intrusion Detection Approaches

Intrusion detection is defined as the method to identify "any set of actions that attempt to compromise the integrity, confidentiality, or availability of a resource" [21]. It is pertaining to techniques that attempt to detect intrusion into a computer or a network by observation of actions, security logs, or audit data [80]. Audit data are information, which any intrusion detection scheme can work on to determine if any intrusion has occurred. The audit data may be obtained on the host, in the application or the system log file through host based intrusion detection system or from the network through network based intrusion detection system. Intrusion Detection Approach (IDA) serves as an alarm mechanism for a computer system or network. It detects the security compromises that happen to a computer network or system and then issues an alarm message to an entity, such as a site security officer so that the entity can take some actions against the intrusion. An IDA contains an audit data collection agent, which keep track of the activities within the system, a detector which analyzes the audit data and issues an output report to the site security officer.

In the context of wireless ad hoc network, we need to identify any malicious nodes either from outside the network trying to break in, or nodes inside the network that have turned malicious. Malicious nodes can easily disrupt or partition the network using the various forms of attacks. Detection of break-ins or attempts is done either manually or via software rule based systems that operate on logs or other information available on the network. We are interested in using automated systems that can study the audit data via certain mechanisms or rules. When working on intrusion detection, some primary assumptions are to be made. The first is that the user and program activities are observable, that is the information regarding the usage of a system by a user or program must be recordable and analyzable. The second and more important is that normal and intrusive behaviors have distinct characteristics [18]. IDA systems have different classifications explained as follows:

Anomaly detection vs. misuse detection:

In order to detect an intrusion attack, one needs to make use of a model of intrusion. That is, we need to know what an IDA should look out for. There are basically two types of models employed in current IDA: anomaly detection and misuse detection.

The first model hypothesizes its detection upon the profile of a user's (or a group of users') normal behavior [77]. It analyzes the user's current session and compares them to the profile representing the user's normal behavior. It then reports any significant deviations to a designated system administrator. As it catches sessions which are not normal, this model is referred to as an 'anomaly' detection model. Anomaly detection bases its idea on statistical behavior modeling and anomaly detectors look for behavior that deviates from normal system use. A typical anomaly detection system takes in audit

data for analysis. The audit data is transformed to a format statistically comparable to the profile of a user. The user's profile is generated dynamically by the system (usually using a baseline rule laid by the system administrator) initially and subsequently updated based on the user's usage. Thresholds are normally always associated to all the profiles. If any comparison between the audit data and the user's profile resulted in deviation crossing a threshold set, an alarm of intrusion is declared. This type of detection system is well suited to detect unknown or previously not encountered attacks. Our model is based on anomaly detection approach.

The second type of model bases its detection upon a comparison of parameters of the user's session and the user's commands to a rule base of techniques used by attackers to penetrate a system. Known attack methods are what this model looks for in a user's behavior. Since this model looks for patterns known to cause security problems, it is called a 'misuse' detection model. Misuse detection bases its idea on precedence and rules, and misuse detectors look for behavior that matches a known attack scenario. A typical misuse detection system takes in audit data for analysis and compares the data to large databases of attack signatures. The attack signatures are normally specified as rules with respect to timing information and are also referred to as known attack patterns. If any comparison between the audit data and the known attack patterns described resulted in a match, an alarm of intrusion is sounded. This type of detection systems is useful in networks with highly dynamic behavioral patterns but like a virus detection system, it is only as good as the database of attack signatures that it uses to compare with.

Host-based vs. network-based intrusion detection:

Most IDA take either a network-based or a host-based approach in recognizing and detecting attacks. In either case, these products look for specific patterns that usually indicate malicious or suspicious intent. An IDA is network-based when it looks for these patterns in network traffic. It is host-based when it looks for patterns in the host system log files.

Network-based approach (NIDA) listens to the network, and capture and examine individual packets flowing through a network [73]. That is, they use raw network packets as the data source. They typically utilize a network adapter running in promiscuous mode to monitor and analyze all traffic in real-time as it travels across the network. They are able to look at the payload within a packet, to see which particular host application is being accessed, and to raise alerts when attacker tries to exploit a bug in such code. NIDA are typically host-independent but can also be a software package installed on dedicated workstation. A side effect of NIDA is that its active scanning can slow down the network considerably [23]. Hence usage of it on an ad hoc network needs to be evaluated. Our model is based on NIDA, since it captures and examines the network traffic for intrusion detection rather than system information.

Host-based approach (HIDA) is concerned with what is happening on each individual host. They are able to detect actions such as repeated failed access attempts or changes to critical system files, and normally operate by accessing log files or monitoring real-time system usage. In order for a HIDA to function, clients have to be installed on every host in the network. These clients reside on the hosts as processes and perform analysis on the audit data gathered locally, at the expense of the already limited resources

of the hosts. Hence care has to be taken to ensure that the HIDA client running on a host in an ad hoc network does not drain resources more than necessary.

Online detection vs. offline detection:

Intrusion detection systems can further be classified according to the timelines of the audit data being gathered and processed. Audit data can be gathered and processed while the host is online (connected to the network) or offline (disconnected from the network). When a system is performing intrusion detection in online mode, the audit data is processed in real-time. A host-based system will gather information about a host as long as the host is connected to the network. A network-based system will monitor the network traffic of the hosts throughout the time they are connected. Any intrusion detected is immediately notified to other hosts. Our model is based on real time online detection. By 'real time' we mean that threat detection is done at the same rate that the network information is captured. By 'online detection', we mean that the network information is captured and threat is detected when the nodes are connected to the network.

When a system is performing intrusion detection in offline mode, the audit data is not processed in real-time but periodically. A host-based system will gather information about a host even if it is not connected to the network. Even if the host is connected, detection is done as scheduled by the system. A network-based system will monitor the network traffic of the hosts periodically as can be in the case of polling. Any intrusion detected is still immediately notified to other hosts but a delay is expected. A typical technique of an offline intrusion detection system is data mining.

1.5 Intrusion Response Approaches

In the event an intrusive behavior is detected, it is desirable to take corrective action to thwart the attack and to ensure safety of the network. Such counter measures are referred to as Intrusion Response Approach (IRA). Although intrusion response framework is related and coexists with the intrusion detection framework, it receives considerably less attention than IDA owing to the inherent complexity in developing and deploying response in an automated fashion. As such, traditionally, triggering an intrusion response is left as a part of the administrator's responsibility.

In the context of MANET, we need to identify the intruder and take a proper evasive or corrective action to isolate the intruder from the network and protect the MANET. We are interested in an automated framework that can identify the intruder through the audit data by means of mechanisms or rules. IRA has different classifications as listed and is explained as follows:

Passive response vs. Active response:

Passive response systems do not attempt to minimize damage already caused by the attack or prevent further attacks. Their main goal is to notify the authority and provide the attack information.

Active responses on the other hand aim to minimize the damage done by the attacker and attempt to locate or harm the intruder. Our model is based on active response approach since it identifies the intruder and mitigates the attack by isolating the intruder.

Manual response vs. Automatic response:

Manual response approaches provide lower degree automation than automatic response approaches but they provide higher degree automation compared to notification-only approach.

Automatic responses provide immediate response to the intrusion through automated decision making process. Although intrusion detection systems are greatly automated nowadays, automatic intrusion response support is still very limited. Our model is based on automatic response due to automated decision making process.

Static response vs. Adaptive response:

Majority of the IRA are static due to the reason that the response selection mechanism remains the same during the attack period. These systems can be periodically upgraded by the administrator; however, such support is manual. Although this approach takes a conservative view of the system and environment, it is simple and easy to maintain.

Adaptive responses on the other hand dynamically adjust response selection to the changing environment during the attack time. Adaptation mechanism can be represented in several ways: (a) adjustment of system resources devoted to intrusion response such as activation of additional IDS, or (b) consideration of success and failure of responses previously made by the system. Our approach is based on static response.

Proactive response vs. Delayed response:

Proactive response approach allows to foresee the incoming intrusion before the attack has effected the resource. Such prediction is generally hard and often relies on the probability measures and analysis of current user/system behavior. Proactive nature of

the response also requires that the detection and response frameworks are tightly coupled such that responses can be fired as soon as a likelihood of attack is identified. Although proactive detection of the attack and early response is a desired feature, it is often hard to guarantee 100% correctness of the triggered action.

In the Delayed response approach, response action is delayed until the attack has been confirmed. Such assurance may be provided through the confidence metrics of IDS. Generally the delayed response leaves more time to the attacker, consequently allowing more damage to the network. Our approach is based on delayed response since the responses are fired as soon as the attack is detected.

Independent response vs. Cooperative response:

Independent response approach handles intrusion independently at the node it was detected. A host-based IDS detecting an intrusion on a single machine will trigger a local independent response such as terminating a process or shutting down the host etc.,

Cooperative response approach refers to a set of response systems that combine effort to respond to an intrusion. Cooperative approach consists of several independent approaches that are capable of detecting and responding to intrusions locally; however the final strategy is determined and applied globally. Since network based IDS are built in such a cooperative manner, our model has a cooperative response approach.

1.6 Problem Statement and Research Objectives

As discussed in the sections above, security is a fundamental concern for MANET. To address these security related issues, there is a need for an efficient and effective intrusion detection and response framework that could detect and respond to the security related attacks at a node level for MANET. This research work attempts to address this need. So the problem statement for the dissertation can be stated as follows:

To design an intrusion detection and response security model, which would detect and control the security related attacks at the node level for mobile ad hoc network in wireless environment.

The objective of this research is to develop such an efficient and effective intrusion detection and response framework with the following features:

- Identify the attack sensitive network parameters and their threshold values to construct the detection and response models.
- Detect threat at the node level effectively in the MANET environment.
- Identify the intruder that caused the threat.
- Respond to the intruder and attacks, thereby controlling the attack and protecting the MANET.
- Design and construct the MANET intrusion detection and response framework such that it is protocol independent.

1.7 Research Contribution

In this research work we have designed and developed a security model called the Intrusion Detection and Response for Mobile Ad hoc Networks (IDRMAN) with the following main contributions:

- Identification of significant attack sensitive parameters through machine learning based decision trees concept; identification of their threshold values using six sigma concept to differentiate their normal, uncertain and vulnerable states for their application in MANET intrusion detection and response.
- Formulation of the measure called Threat Index for effective Intrusion Detection on MANET. Threat Index is computed using fuzzy logic.
- Intruder identification and response framework model for attack control and protection of MANET mobile nodes that are under threat by identifying the intruder and subjecting appropriate response plan.
- Protocol independent infrastructure for protecting the MANET from active attacks by measuring critical parameters in the underlying MANET infrastructure. The proposed model continuously monitors the online network data and efficiently detects the attacks, irrespective of the protocol used in MANET.

1.8 Overview of the Dissertation

In Chapter II, we summarize the existing security schemes for MANET based on security classification. We have also summarized the related work that has been done in IDA for MANET. We have also reviewed the related work done in Intrusion Response for MANET. We have also presented a detailed review of two related MANET IDA schemes that we have used later for the performance evaluation experiments in order to evaluate and compare the performance of the detection aspect of the proposed model. In this chapter, we have also listed the security feature requirements of IDA and MANET. We also describe the characteristics of MANETs and why they are particularly vulnerable to attacks. Finally, in this chapter we have presented the limitations of the existing MANET security schemes, MANET IDA's and MANET IRA's. While presenting the limitations, we have also proposed metrics for the performance evaluation experiments of the proposed model in order to quantify the improvements attained in MANET security with the proposed model

In Chapter III, we describe the detection framework of the Intrusion Detection and Response model for Mobile Ad hoc Networks (IDRMAN). This chapter explains the concepts of classification trees and six-sigma that are used to identify the attack sensitive metrics and their thresholds respectively. The concepts of fuzzy logic are introduced to explain the calculation of the metric called Threat Index (TI) that is used to detect if a node is under threat or not. We have also explained the methodology of threshold identification for the threat index. An overall algorithm for the detection aspect of the model is then presented. Finally the chapter has a proposition that mathematically analyzes the metric (Threat Index) used in the detection framework.

In Chapter IV we propose the intruder identification and response operations methodology. This chapter presents an overall algorithm for the intruder identification and response aspect of the model. In this chapter, we have also stated a proposition that evaluates the effectiveness of the response algorithm. Finally, we have explained how the proposed Intrusion Detection and Response model can be applied in MANET environment for several known attacks.

Chapter V explains the simulation experiments carried out to demonstrate the validity of the intrusion detection and response model described in Chapters III and IV. This chapter describes the experiments that were conducted to identify the significant network parameters and their threshold values. This chapter then presents the simulation experiments and results for the Intrusion Detection and Response aspect of the model when AODV and DSDV protocols are used for MANET simulation. Experiments to compare the detection aspect of the proposed model with related MANET intrusion detection approaches, and experiments to demonstrate the scalability feature of the proposed model with respect to the size of the MANET are also presented in this chapter.

Chapter VI presents the conclusion. This chapter summarizes the work, review the contributions and points out the further developments that can be done to this model.

CHAPTER II

LITERATURE REVIEW

2.1 Classification and Review of MANET Security Schemes

In this section, we classify the MANET security work into four broad categories based on the type of attack: authentication, denial of service, selfish node, and routing [55]. In ad hoc networks, a mobile node or host may depend on other node(s) to route or forward a packet to its destination. The security of these nodes could be compromised by an external attacker or due to the selfish nature of other nodes. This would create a severe threat of Denial of Service (DoS) and routing attacks where malicious nodes combine and deny the services to legitimate nodes. Unlike nodes in a wired network, the nodes of MANET may have less processing power as well as battery life and consequently would try to conserve resources. In this scenario, the usual authentication and encryption methods would not apply to a MANET the same way they would in a wired network [32]. However, both authentication and encryption are even more important in a MANET [27, 28]. Steiner et al have developed a Group key Diffie-Hellman (GDH) model that provides a flexible solution to group key management. Yi et al [42] have developed the MOCA (MObile Certification Authority) protocol that helps manage heterogeneous mobile nodes as part of a MANET. MOCA uses Public Key Infrastructure (PKI) technology.

The impact of authentication attacks is quite widespread and it includes unauthorized access, denial of service, masquerading, information leakage, and domain

hijacking. Capkun et al [40] have developed some solutions using a concept that they introduce, called Maximum Degree Algorithm (MDA), for preventing denial of service due to poor key management. Avoine et al [45] have developed a cryptography-based fair key exchange model called Guardian Angel. This model uses a probabilistic approach without involving a trusted third party in key exchange.

We noted earlier some of the problems due to selfish nodes not performing their role properly in a MANET. Actions of a selfish node could lead to congestion, lower throughput and denial of service. Buttyan et al [39] have shown by simulation that a selfish node does not participate actively in packet forwarding in order to conserve electrical energy. This study shows that typically every node spends 80% of the energy in forwarding packets. This work also introduces a special counter called nuglet counter that is used to keep track of selfish behavior of nodes. In trying to solve the selfish node problem, Michiardi et al [43] have developed a model called CORE (COllaborative REputation). Under CORE's approach, every node monitors the behavior of the neighboring nodes for a particular requested function and collects data about the execution of that function. If the observed result of the function matches with the expected result, then the observation takes a positive value. This mechanism allows a node to detect if any of its neighbors are selfish nodes and gradually isolate them.

As seen above, the problem of selfish behavior by nodes in a MANET is something significant that needs to be addressed. In a MANET, many nodes try to conserve battery life and consequently resort to selfish behavior by dropping packets rather than forwarding them as they are supposed to do in a network. Buchegger, et al [44] study the vulnerabilities exposed by selfish nodes in a MANET. Buchegger et al [44]

introduce a new protocol called CONFIDANT (Cooperation Of Nodes – Fairness In Distributed Ad hoc NETworks) to address this problem. Each node maintains reputation indexes about each of its neighbors based on their behavior and use these indexes to isolate misbehaving nodes.

Routing is an important aspect of moving packets around in a network. It is a challenging problem because nodes within the ad hoc network themselves performs routing function and the security concepts were not incorporated into the routing protocols when they were designed. It is important because the routing table forms the basis of the network operations and any corruption of routing table may lead to significant consequences. Routing attacks in mobile ad hoc network are more challenging since routing relies on the trustworthiness of all the nodes involved and it is difficult to distinguish selfish nodes from normal nodes. Basically there are two methods used for routing: AODV (Ad hoc On-demand Distance Vector) routing and DSDV (Destination Sequenced Distance Vector) routing. These two methods can be classified as reactive and proactive respectively since AODV method discovers a route only when needed whereas the DSDV method maintains a dynamic routing table at all times.

A reactive routing method was proposed by Yang et al [49]. In this method, a unified network layer prevention method known as Self Organized Security (SOS) scheme that uses AODV routing is used. This scheme takes a self-organized approach by exploiting full localized design, without assuming any *apriori* trust or secret association between nodes. In this model, each node has a token in order to participate in the network operations, and its local neighbors collaboratively monitor it to detect any misbehavior in routing or packet forwarding services. Upon expiration of the token, each node renews its

token via its multiple neighbors. The period of the validity of a node's token is dependent on how long it has stayed and behaved well in the network. A well-behaving node accumulates its credit and renews its token less frequently as time evolves. In essence, this security solution exploits collaboration among local nodes to protect the network layer without completely trusting any individual node.

Another reactive scheme, called Techniques for Intrusion-Resistant Ad Hoc Routing Algorithms (TIARA) was proposed by Ramanujam et al to detect and eliminate DoS [46]. This model presents a new approach for building intrusion resistant ad hoc networks in the wake of DoS attacks using wireless router extensions. This approach relies on extending the capabilities of existing ad hoc routing algorithms to handle intruders without modifying the existing routing algorithms. This scheme proposes a new network layer mechanism for detecting and recovering from intruder induced malicious faults that work in concert with existing ad hoc routing algorithms and augment their capabilities.

Hu et al [47] have developed a DSDV-based secure routing method called SEAD (Secure Efficient Ad hoc Distance vector). This method uses efficient one-way hash functions and does not use symmetric cryptographic operations in the protocol in order to support the nodes of limited CPU processing capability and to guard against Denial-of-Service (DoS) attacks. The primary reason for this is due to the fact that the nodes in an ad hoc network are unable to verify asymmetric signatures quickly enough for routing protocols to decide on the routing path.

Routing attacks differ in their execution depending on the nature of the routing protocol. In the case of link state routing protocol such as AODV, a router sends

information about its neighbors. Hence, a malicious router can send incorrect updates about its neighbors or remain silent if the link state of the neighbor has actually changed. However, in case of distance vector protocols such as DSDV, routers can send wrong and potentially dangerous updates regarding any nodes in the network since the nodes do not have the full network topology. Zapata et al [35] studies the behavior of routers in the presence of Byzantine faults. They use an On-demand Secure Routing Protocol (OSRP) that defines a reliability metric based on past records and use it to select the secure path. Reliability metric is represented by a list of link weights where high weights correspond to low reliability. Each node in the network maintains its own list, referred to as a weight list, and dynamically updates that list when it detects faults. Faulty links are identified using a secure adaptive probing technique that is embedded in the normal packet stream. These links are avoided using a secure route discovery protocol that incorporates the reliability metric. This protocol achieves these functionality by three successive phases: Route discovery with fault avoidance phase whose input is source node's weight list and output is the full least weight path from the source node to the destination node, Byzantine fault detection phase whose input is the full weight path and output is a faulty link and link weight management phase which takes a faulty link as an input and whose output is the weight list which in turn is used by the route discovery phase to avoid faulty paths. This is a very efficient approach to detect secure routes. In a related paper, Zapata [35] discusses a method for secure ad hoc routing.

Zhou et al. [54] have an alternative solution for the problems with AODV and DSDV routing methods. They have developed a hybrid approach using both AODV and DSDV methods. This method, known as the Key Management Service (KMS), defends

routing from denial of service attacks in ad hoc networks by taking advantage of multiple routes between nodes. Due to the dynamic changes in topology, the routing protocols of ad hoc network need to handle outdated routing information, which is similar to that of the compromised routing attacks. The principle here is that as long as there are enough proper nodes, the routing protocol would be able to find the routes working around the compromised nodes. Thus, if the nodes can find multiple routes, nodes can switch to an alternate route when a fault has been detected in the primary route. This method also uses replication and new cryptographic schemes, such as threshold cryptography, to build a highly secure and highly available key management service, which forms the core of the security framework.

In addition to the methods discussed above, there are some additional methods proposed in the literature to handle various forms of attacks. For example, the Secure Routing Protocol (SRP) by Papadimitratos et al [36] guarantees correct route discovery, so that fabricated, compromised, or replayed route replies are rejected or never reach the route requester. SRP assumes a security association between the end-points of a path only and so intermediate nodes do not have to be trusted for the route discovery. This is achieved by requiring that the request along with a unique random query identifier reach the destination, where a route reply is constructed and a message authentication code is computed over the path and returned to the source. The authors prove the correctness of the protocol analytically.

Another preventive solution for DoS attacks in ad hoc wireless networks is proposed by Luo et al [50]. In this solution they distribute the functionality of authentication servers, thus enabling each node in the network to collaboratively self-

secure themselves. This is achieved by using the certificate-based approach. This scheme supports ubiquitous security for mobile nodes, scales to network size, and is robust against adversary break-ins. In this method centralized management is minimized and the nodes in the network collaboratively self-secure themselves. This scheme proposes a suite of fully distributed and localized protocols that facilitate practical deployment. It also features communication efficiency to conserve the wireless channel bandwidth and independency from both the underlying transport layer protocols and the network layer routing protocols.

The ARIADNE method developed in Europe is another important secure on-demand routing protocol. Developed by Hu et al [51], ARIADNE (Alliance of Remote Instructional Authoring and Distributed Networks for Europe) prevents attackers from tampering with uncompromised routes consisting of uncompromised nodes. It is based on Dynamic Source Routing (DSR) approach and relies on symmetric cryptography only. ARIADNE protocol is designed in three stages: The first stage presents a mechanism that enables the target to verify the authenticity of the *Route Request*. Second stage presents a key management protocol that relies on synchronized clocks, digital signatures, and standard MAC (Message Authentication Code) for authenticating data in *Route Requests* and *Route Reply*. The final stage presents an efficient per-hop hashing technique to verify that no node is missing from the node list in the *Request*. Hu et al present simulations that show that the performance is close to DSR without optimizations.

Marti et al [53] have taken another variation on the DSR method. This method shows increased throughput in mobile ad hoc networks by complementing DSR with a *watchdog* for detection of denied packet forwarding and a *pathrater* for trust management

and routing policy rating that every path uses, thus enabling nodes to avoid malicious nodes in their routes as a detective and reactive protection measure. This reaction does not punish malicious nodes that do not cooperate, but actually relieves them of the burden of forwarding for others while having their messages forwarded, and it allows nodes to use better paths and thus increase their throughput.

The traditional Secure Routing Protocol (SRP) is well suited for a wired network. In developing a similar protocol for MANETs, Yi et al [52] propose a new routing technique called Security-Aware ad hoc Routing (SAR) that incorporates security attributes as parameters into ad hoc route discovery. SAR enables the use of security as a negotiable metric to improve the relevance of the routes discovered by ad hoc routing protocols. Ad hoc routing protocols enable nodes in ad hoc networks communicate with their neighbors through Route REQuest (RREQ) packets and Route REPly (RREP) packets. In SAR, the security metrics are embedded into RREQ packets. Intermediate nodes receive these packets with particular security level and process these packets or forward the packets depending on the security level of the intermediate node. If it cannot provide required security level, RREQ packets are dropped. Otherwise RREP packets are sent back to the source from destination or intermediate nodes. This approach, though resource intensive is a useful alternative for preventing attacks.

So far we have looked at research that addresses authentication, denial of service, selfish node and routing protocol attacks in a MANET. One of the main requirements in a MANET is for each node to let other nodes know of their presence and readiness to participate in the MANET. In a wireless local area network, an Access Point (AP) is used to let the mobile nodes communicate with other nodes on the network. In a

MANET, there is no Access Point and so each node must know the other nodes that participate in the MANET. One way to let the other nodes know of their presence, a mobile node sends out beacon signals. Binkley et al [48] propose an authenticated routing protocol to address link security issues in this regard. This proposal also reduces the DoS threats like replay attacks caused by an Address Resolution Protocol (ARP) or ad hoc routing protocol *spoof,* which would destroy a link-layer route to a host. This protocol transmits beacons similar to that of mobile IP agents. When a host node or agent receives the transmitted beacons, they authenticate them and if it is authentic, they add the MAC-to-IP address binding contained in the beacon into their table of authentic bindings.

Another security scheme proposed by Kong et al [57] and Luo et al [29] supports ubiquitous security services for mobile hosts through threshold secret sharing mechanism where they distribute certificate authority functions. These methods are based on RSA cryptography and provide distributed localized certificate services like certificate issuing, renewal and revocation. These methods employ localized certification schemes to enable ubiquitous services. This model uses RSA system key pair denoted by $\{S_k, P_k\}$ where S_k is the system secret/private key and is used to sign certificates for all entities in the network. P_k is the system public key which verifies the certificate signed by S_k. In this scheme, S_k is shared among network entities but not visible or known by any component in the network, except at the boot strapping phase. Each entity V_i also maintains a secret share P_{vi} and a RSA personal public and private key pair $\{sk_i, pk_i\}$ besides the system key pair. Thus, it uses the concept of threshold secret sharing and updating each entity's secret share periodically to further enhance robustness against break-ins. This scheme scales to network size and is robust against break-ins. In the threshold secret sharing

mechanism each entity holds a secret share and multiple entities in a local neighborhood jointly provide complete services.

There are several open issues in the models that were reviewed. The important among them are explained as follows: The GDH method needs further study for the detection and resolution of inconsistent certificates, improvement of certificate graph models and enhancing the use of existing PKI infrastructure. The MOCA method uses a unicast approach that only exploits information in the local routing cache. One useful extension would be to devise a way for a node to browse neighboring nodes' routing tables. This would help in avoiding flooding. The CORE method considers only attacks from selfish nodes but not from active intruders. Hence one has to extend this method for intruder attacks as well. The solution for attack by selfish nodes presented in the nuglets method is focused just on packet forwarding attacks. Application-level issues like mutual provision of information services in an ad hoc network have to be addressed in order to better utilize the nuglet counter. The CONFIDANT method assumes that nodes are authenticated and that no node can pretend to be another in order to get rid of a bad reputation. This assumption could lead to misplaced trust in systems. The Guardian Angel method is not a comprehensive security scheme since it does not take into account the attacks like packet forwarding and denial of service or routing attacks, which are commonplace today. The summary of all the methods presented above to address the security issues related to MANET is illustrated in Table 2.1.

Table 2.1 Classification of attacks and security schemes in MANET

Method	Type of attack				Open Issues
	Authentication	Routing	Selfish	DoS	
GDH [40]	Yes	No	No	No	Mechanism for certificate issues
MOCA [42]	Yes	Yes	No	No	Does not use the support of PKI
CORE [43]	No	No	Yes	No	Considers only selfish node attack
Nuglets [39]	Yes	Yes	Yes	No	Scheme is not generalized
CONFIDANT [44]	No	Yes	Yes	No	Assumes nodes are authenticated
Guardian Angel [45]	Yes	No	Yes	No	Does not support varied attacks
TIARA [46]	Yes	Yes	No	Yes	Not a generalized scheme
SEAD [47]	Yes	Yes	No	Yes	Packet forwarding
Beacon [48]	Yes	Yes	No	No	Scalability and Key Management
SOS [49]	Yes	Yes	No	Yes	Scalability issues
SRP [36]	Yes	Yes	No	Yes	Unfair utilization of resources
ARIADNE [51]	Yes	Yes	No	Yes	Not optimized
SAR [52]	Yes	Yes	No	No	Packet mistreatment attacks
OSRP [35]	Yes	Yes	No	No	Fixed but not adaptive threshold
WatchDog/Pathrater [52]	No	Yes	No	No	Assumes no apriori relationship

2.2 Review of Intrusion Detection Approaches in MANET

The following are some of the popular IDA models for MANET that we studied in our literature survey. Kachirski and Guha proposed an IDS model which is efficient and bandwidth-conscious [63]. It targets intrusion at multiple levels and fits the distributed nature of IDA for MANET. The method has clusters and the IDA on cluster head employs independent detection decision-making after gathering information from other nodes. It utilizes mobile agent for communication among various nodes. This model provides a framework to work with multiple types of audit data. It is expandable, meaning, if the IDA needs to work with new types of audit data, it can do so by just incorporating extra agents that can monitor the new type of audit data. Unfortunately, its performance is not verified by any implementation. Once its performance is proved to be

on an acceptable level, this framework can serve as a generic and expandable architecture for commercial products, since having a possibility to add in more functionality is an important property for successful products. Because it utilizes the cluster heads, it is supposed to make the MANET more efficient by limiting the resources usage for IDA purposes to only a few nodes. Such a framework can be applied in an environment where the security requirement is medium and efficiency requirement is high. Also, it may easily be expanded for multi-layered MANET.

IDS model for wireless mobile ad hoc networks proposed by Zhang and Lee implements local and collaborative decision making with anomaly detection [18]. In this approach, individual IDA agents can work by themselves and also collaborate in decision making. Each IDA agent runs on a node and monitors local activities. If a node detects local intrusion with strong evidence, then the node concludes that intrusion has happened and initiates an alarm response. However, if the evidence is not strong enough but needs investigation in a wider area in the network, then the IDA agent can start collaboration procedure which is a distributed consensus algorithm. This model provides a framework that fits the distributed nature of MANET. It also works with multiple types of audit data. If the IDA needs to work with new types of data, it can add in more data collection module in the IDA agent. It uses data mining as the local intrusion detection mechanism. The data mining is supposed to be superior in terms of both detection rate and false alarm rate. Also, because this IDA does not use mobile agents for communication, it can be designed for high security need, if it can find an effective way to protect from Byzantine nodes. This framework is designed for flat MANET. In a large multi-layered MANET, it can work in a subsection of the MANET.

Huang and Lee have proposed a cluster-based scheme in which a cluster head is elected by a group of nodes in a neighborhood (citizen nodes) and the head node monitor the citizen nodes [64]. Once the cluster head is elected, the other nodes need to transmit the features they obtain locally to the cluster head. This IDA uses anomaly detection implemented with data mining as its detection technique [64]. This model improves the efficiency of MANET by limiting the resources usage for IDA purposes to only a few nodes. The implementation proves it can also achieve satisfactory level of detection rate. Such a framework can be applied in environments where the security requirement is medium but efficiency requirement is high. Also, it may easily be expanded for multi-layered MANET [75].

Patrick and Camp have designed architecture for ad hoc networks, where each node runs a local IDA [65]. Each node detects intrusion locally and uses external data to confirm the detection. The nodes use mobile agents to communicate and collaborate. This model provides a scalable architecture by using mobile agents. If the IDA needs more functionality, it can just incorporate more mobile agents with new tasks. It is supposed to reduce network traffic for intrusion detection purpose. However, since this architecture relies heavily on the use of mobile agents, it incurs computational complexity in creating and managing all the agents. This architecture needs an implementation to verify its performance.

Bo, Wu and Pooch have proposed an IDA model which uses collaboration mechanism with anomaly detection [58]. In this model, a network is divided into logical zones. Each zone has a gateway node and individual nodes. Individual nodes have an IDA agent to detect intrusion activities individually. Once an individual node detects

intrusion, it generates an alert message. Gateway node aggregates and correlates the alerts generated by the nodes in its zone. An algorithm is used to aggregate the alerts based on the similarities in the attributes of the alert [75]. Only gateway nodes utilize the alert to initiate an alarm [58]. This method does not use mobile agents but has gateway nodes, which work just like a cluster head. This architecture can be applied in environment where the requirement for IDA performance and security is high and MANET resources are generally available.

Guha et al., have proposed a model that utilizes cluster with cluster head and employs the independent decision making module [67]. It also utilizes the mobile agent for communication among nodes. The intrusion detection engine is a case-based agent designed with the principle of artificial intelligence. This model can automatically construct anomaly model but has high computational costs.

Huang et al., have proposed a detection algorithm scheme that uses the statistics of packets, namely, the relation between different features such as the correlation between the number of packets dropped and the percentage of change in routing table [68]. This algorithm can be used as an intrusion detection engine in other IDA architectures. This model has low overhead, but was designed only for one routing protocol-OLSR and needs modification for other protocols.

Tseng et al., have proposed an IDS system where the normal behavior of critical objects in the network is constructed with the normal specification first. Then the actual behavior is compared to the normal specification [69]. It uses distributed network monitor to trace the request-reply flow in the routing protocol. The network monitor runs a specification based detection algorithm to make decisions [76, 77]. This model is novel

with no conventional local detection mechanism, but has low efficiency since packet is checked at each hop.

Neighborhood Watch, an IDS protocol proposed by Sowjanya and Shah has two neighboring nodes of which one node is used to ensure that the packets are not modified while traveling in the network [70]. This is done by comparing the information in each packet at each hop. It has two modes: passive mode - to protect a single host and active mode - to collaboratively protect the nodes in a cluster. In active mode, a cluster head starts a voting algorithm to determine whether intrusion really happens.

Puttini et al., have proposed an IDS architecture where information in the management information base (MIB) is used as input data [71]. It also uses mobile agent and a collaborative decision making mechanism. This model is distributed and efficient in use, with high scalability and can detect attack at multiple levels, but has security, computational cost and management problems related to mobile agents.

IDS Model proposed by Brutch and Ko is a statistical anomaly detection algorithm [72]. It works by first assuming that the audit trail generated from a host has been converted to a canonical audit trail (CAT) format. It then uses a CAT file to generate session vectors representing the activities of the users' sessions. These session vectors are then analyzed against specific types of intrusive activities to calculate "anomaly scores". If the scores cross some thresholds, warnings reports are generated. The algorithm analyzes a session vector in three steps: 1) it calculates a Bernoulli vector, 2) it calculates the weighted intrusion score, and 3) it calculates the suspicion quotient. The Bernoulli vector is generated from the session vectors as well as some threshold vectors. It is a simple binary vector in which the values in the vector are set to one if the

corresponding arbitrary counts fall outside the threshold for a particular user group. The weighted intrusion score is generated for a particular session and for a particular intrusion type. It can be used to assign a suspicion value to the session. This suspicion value, or suspicion quotient, for a session is determined by what percentage of random sessions have a weighted intrusion score less than or equal to the weighted intrusion score of the current session. It describes how closely a session resembles the intrusion type as compared to all other sessions. The Haystack algorithm gets its name by being the algorithm implemented in the IDA called Haystack. Haystack is a host-based system, which attempts to detect several types of intrusions: attempted break-ins, masquerade attacks, penetration of the security system, leakage of information, denial of service, and malicious use. It was initially developed for use in the US military network. This algorithm is designed for use in a secured wired military network. If in a wireless ad hoc environment, it requires a designated node to act as a central administrator and all the other nodes to allow the central administrator to retrieve audit trails from them. The central administrator can be pre-designated by the human initiator of the ad hoc network or can be assigned by programming. The audit trails requested can be submitted by the nodes themselves or by the mobile agents allowed to run on the nodes.

An IDS approach, Indra, proposed by Janakiraman et al., is a distributed intrusion detection scheme based on sharing information between trusted peers in a network to guard the network as a whole against intrusion attempts [74]. It is a detection tool that takes a proactive and P2P approach to network security. The basic idea behind this model is cross monitoring or simply called “neighborhood watch”, and is very simple. In this method, the hosts on the P2P network join together to form some sort of an immune

system where each host distributes information on attempted attacks among the interested peers in the network. Such information is usually gathered by the intended victim of an attack and by notifying its adjacent hosts, an alarm can be sounded. This allows the system to react proactively or retroactively. When an alarm is sounded, subsequent attacks to other hosts are repelled straightaway as the adjacent hosts would have forewarned other hosts. Alternatively, it is also possible for hosts in the network to detect other hosts as being under attack. This is effective if the network is a shared medium but the same effect can be achieved by having Indra installed on network gateways or on a machine attached to a "snoop" port of a network switch. The functionality of this model is achieved by a set of daemons. Each interested host on the network runs a special security daemon, which watches out for intrusion, attempts and also enforces access control based on its memory of earlier attempts. Other daemons that belong to different classes help to look out for suspicious activities, aggregate the warning notifications, or communicate with other hosts. These daemons could be configured by the system administrator for different levels of security. Though the scheme is easy to understand, there are practical difficulties at the levels of communication, trust and policy. Indra has gone on to address these issues. In Indra, trust is an important issue. This is especially true in an intrusion detection system that lacks a centralized trusted authority to provide digital certificates. That is, a network without a certification authority (CA) will not have its host nodes trusting one another for the information-sharing idea in this model to work. This model requires a certain level of trust between hosts so that the daemons running on the hosts can trust the messages received from other hosts. Unfortunately, a wireless ad hoc network is one such network where trust is virtually non-existent. In the wireless ad

hoc environment, a host is supposed to trust no other host except itself. This poses a problem when trying to deploy this model as the intrusion detection scheme in the ad hoc network. This model also tries to address the trusting issue. The usual decentralized alternate to central CA is the web-of-trust model. In the prototype version of this model, certificates for hosts are obtained from trusted key servers for certifying among peers, but variants of the web-of-trust model are used in real situations. Communication in this model is done with the use of daemons between trusted hosts. Therefore, all the hosts and subsequent hosts joining into the wireless ad hoc network will need to have the various daemons running before they are allowed to operate in the network. So far, this model has been investigated on several networks and models of communication. In general, this model can be deployed on any peer-to-peer network as long as mechanisms which provide a node to propagate information to a randomized subset of its neighbors, or to create multicast trees are present. Policy in a wireless ad hoc network, regardless of the intrusion detection scheme employed, has to be pre-defined and agreed upon by the hosts joining the network. Alternatively, all these can be set as a pre-condition for hosts belonging to the network.

Most of the surveyed models use packets and network traffic related information such as updates in routing table or request-reply flow in the network. Among the ones that use packets related information, IDS approach proposed in [67, 69] uses the information inside the packets header directly, such as network address or port number. Other models using packet or network traffic related information mainly use statistical data processes from packet information, such as the statistics of the number of packets received and sent or the statistics of change in routing table. IDS Model as described in

[68] utilizes the statistics derived from packet or traffic related statistics, for instance, the correlation between the number of packets dropped and the percentages of updates in routing table. Intrusion Detection approaches illustrated in [18, 63] allow the IDA to work on different types of audit data or the possibility to adapt to different types of audit data. This property is valuable and should be an important consideration for the future design of IDA. Most of the architectures detect only the fact that an intrusion happens. Some models go further to obtain more information, such as the type of attack and the location of the intruder. For instance, Zone based IDA can detect both the type and location of the attack [58].

Some of the intrusion detection models utilize cluster head or gateway nodes [63, 64, 67, 70]. The advantage of cluster head is that some of the resource consuming computation, such as intrusion detection, can be carried out only on some nodes of the network. Therefore, most other nodes can focus on the real work of network traffic. The cluster head usually collects information from cluster member to make the detection decision. In some methods, the original input data is further processed or formatted before it is sent to the cluster head. By doing this, the network traffic for transferring such data is reduced. The computation on the cluster head can also be reduced because the incoming data from member nodes is already formatted for the IDA use. The security communication between the cluster head and its member nodes should receive attention of research.

Most of the methods in our review, except the model proposed in [69], utilize anomaly detection. The anomaly detection is more suitable than misuse detection in MANET. In MANET, the anomaly detection has a weakness: the profile of normal

behavior needs to be updated periodically. This places a heavy burden on the limited network resources.

All the architectures need some form of communication between the IDA running on different nodes. The communication can be done with mobile agents. Some of the models in our review utilize mobile agents [63, 65, 67, 71]. The objective of using mobile agents is to reduce the network traffic and leave more resources for the real work of network. However, such architectures, where the mobile nodes allow the mobile agents to carry out computation on them, also open the door for attacks. Therefore, the security mechanism that protects nodes from malicious code is very important [26, 31]. And such mechanism may make the mobile agents less powerful and efficient, which is just one of the important considerations for using mobile agents [22]. Also, mobile agent management, such as the creation, migration, operation and termination of mobile agents, is also quite challenging [30]. Those architectures, which do not use mobile agents, rely on network protocols to exchange data and collaborate in intrusion decision making [61, 62]. Such protocols need to be secure and robust [59]. At the same time, such communication uses lot of the bandwidth resources which is very limited in MANET [1]. Some of the Intrusion Detection architectures utilize collaborative decision-making in intrusion detection [18, 58, 70, 71]. Others use independent decision making. The objective of using collaborative decision-making is to include information from different nodes in the decision making so as to make more accurate decisions. The collaborative decision making has some weaknesses in terms of security. It is more easily compromised under the attacks such as denial of service and spoofed intrusion.

In the mobile agents based global distributed and modular secure architecture [58], the intrusion detection approach is provided by local IDS (LIDS) entities. These entities are located on each node of the mobile ad hoc network (MANET) and collaborate with other LIDes through the use of mobile agents. Due to the lack of the centralization in MANET, some of the tasks required for the intrusion detection processes should be executed in a distributed and cooperative manner. Mobile agents are an alternative to the client-sever distribution model. Mobile agents can provide a first element of response to the problem of the scalability for the global intrusion detection process. When a node joins the network, it does so with running LIDS and a mobile agent platform. It can therefore immediately take part in the global cooperative intrusion detection process. This modular architecture is divided into three main processes: data collection, detection algorithm design and alert management. Each of the elements in the model: Sensor, Analyzer and Manager are related with one of the intrusion detection processes. A sensor collects data from a data source, an analyzer processes the collected data for detecting signs of events that might have security concerns and the manager stands for the management interface of the whole process, besides doing alert correlation and response initiation. Activities monitored by the sensor in the data source may be mapped in events which are passed to the analyzer where they are submitted to the hybrid (misuse and anomaly detection) intrusion detection algorithm. When the analyzer finds events with relevant security concerns, alert are generated to the manager. All raw data collection and pre-processing is performed locally in the same LIDS. While executing all raw data collection and abstraction locally, node detects attacks against some of its neighbors. Thus, whenever some (high level) message needs to be processed remotely, a mobile

agent is dispatched to the remote node carrying the data and possibly the code needed for the remote message processing. Mobile agents are created, received (from a remote host) and managed in the mobile agent framework. These mobile agent platforms should also provide security services (e.g. server authentication, agent and server code integrity, access control to local resources, etc) related to agent activities.

Data mining algorithms implemented on each mobile node can be used to analyze audit data and thereafter generate intrusion detection models. Data mining generally refers to the process of extracting useful models from large repositories of data [64]. There are several algorithms proposed in the literature that are particularly useful for mining audit data for anomaly detection. Classification is the process by which a data item is mapped into one of the several predefined categories. The classification algorithms normally produce "classifiers" that can be in the form of decision trees or rules. Sufficient "normal" and "abnormal" audit data must be gathered before a classification algorithm can be applied to learn a classifier that can categorize new unseen audit data as belonging to the normal class or the abnormal class. Sequence analysis involves the analysis of frequent sequential patterns of audit data in order to gain insight into the temporal and statistical nature of many attacks as well as the normal behavior of users and programs. The statistical information collected can then be incorporated into intrusion detection models.

2.3 Review of Intrusion Response Approaches in MANET

Although intrusion response component is related and coexist with the intrusion detection framework, it receives considerably less attention than detection framework owing to the inherent complexity in developing and deploying response in an automated

fashion [125]. Most of the security models generate an alarm informing the administrator, who then decides the response. However, it is desirable that the response consists of an automated corrective action to protect the network from an identical future attack.

There are few IDA models that provide the integrated detection and response feature. Zhang et al., in their framework have explained that local response module triggers action local to the mobile node and the global response module coordinates actions among neighboring nodes, such as the IDS agents in the network electing a remedy work [18]. They have also explained that the type of response depends on the type of intrusion, the type of protocols, applications and the confidence in the evidence with examples. However, they have not provided any implementation details regarding the intrusion response aspect of the model. Similarly, there is no documentation on the simulation or experimental results on the response aspect of the model. However, there is a detailed explanation on the experimental results of the detection framework of the model. Thus, even though the idea of integrated detection and response model seems feasible, it appears that the implementation and simulation have not been conducted. Similarly, few related IDA models propose response actions/frameworks for responding to the attacks once it is detected [119-124]. However the response system incorporating all those actions is not implemented.

There are few intrusion prevention approaches described in the literature for MANET security. Puttini et al., have proposed a secure routing protocol that combines a certificate based authentication service with intrusion detection model to provide preventive and corrective protections for MANET [71]. Bhargava et al., have proposed a security model for AODV routing protocol to prevent attacks in MANET [118].

2.4 Requirements of IDA for MANET

IDA for MANET must work with localized and partial audit data. In MANET, the audit data is always localized and partial because MANET does not have a fixed infrastructure such as firewall or gateway like in wired network to collect complete and global audit data [18]. It is more difficult for IDA in MANET to distinguish between normal and intrusion traffic. In wireless network, there is often no clear line between normal/abnormal activities. In wireless network the connection is not stable and mobile nodes can join and leave the network at any time. For instance, a node, which is temporarily out of synchronization, may send packets that could be considered packets of attack activities [18]. IDA should utilize minimum resources. The wireless network does not have stable connection, and the physical resources of the network and the devices, such as bandwidth and power are limited. Disconnection can happen at any time [18]. In addition, the communication between nodes for IDA purpose should not use much bandwidth resources.

Encryption in communication for wireless mobile networks is difficult to achieve. The communication between IDA on different nodes must be secure to not allow attackers gain the access to such communication. However, encryption in MANET is a difficult task itself. In wired network, because of the requirement of physical connection for access, this problem is less obvious. IDA cannot assume any node is secure. Unlike in a wired network, MANET nodes can be very easily compromised. Therefore, in cooperative algorithm, the IDA must not assume that any node can be fully trusted. IDA must address high false alarm rate problem. It is difficult to obtain enough audit data to make an intrusion detection decision, because the bandwidth of MANET is much

restricted compared with wired network. As a result, IDA in MANET can easily result in either having too much false alarm or missing many attacks [57].

Thus, the most important requirement feature is to find an appropriate architecture of IDA that will fit the mobile and ad hoc nature of the wireless network. Also, finding a way to effectively use the audit data source in wireless network with anomaly detection is also very important. As mentioned earlier, the audit data in wireless network is often partial and local. Another requirement of the security model is to find a way to effectively distinguish attack traffic from normal traffic, especially the normal traffic that seems abnormal due to factors such as poor network connections. Otherwise, the IDA will have a high false alarm rate [18].

Levente Buttyan identified the requirements of IDA for MANET as follows [5]:

- IDA must be truly distributed, which means IDA must detect intrusion on each node, but nodes can collaborate in making decision on whether to issue an alarm.
- IDA should be able to deal with local and partial audit data; IDA may need to sense anomaly happened on other hops [18, 53, 58].
- IDA should be able to deal with the problem that there is no clear line between normal/abnormal; IDA need to obtain high detection rate and low false alarm.
- Given the resources constraints on wireless network, IDA should not consume too much resource, including power. Therefore, IDA should have run-time efficiency.

2.5 Review of Related MANET IDA Schemes for Performance Evaluation with the Proposed Model

Two related models - Integration of Mobility and Intrusion Detection for Wireless Ad hoc Networks (MIDWAN) by Sun et al., and Cooperative Intrusion Detection System for Ad hoc Networks (CIDSAN) by Huang and Lee are chosen for the detailed review and performance evaluation experiments with the proposed model. MIDWAN is chosen because, as per our literature survey, this model appears to be the latest anomaly detection based IDA for MANET in the literature [58]. The second model, CIDSAN is also an anomaly-based model [64]. This appears to be the latest model as referenced by Sun et al., in their manuscript [58]. This model by Huang and Lee is an improvement of their earlier work with Zhang [18]. Since the proposed model is based on anomaly detection approach, misuse detection based IDA and Mobile Agent based IDA are not chosen for this detailed related model review and performance evaluation experimentations.

Related Scheme 1: MIDWAN by Sun et al.:

MIDWAN is based on Markov Chain based Anomaly detection model for MANETs. In this model, each node includes a local IDS agent. The internal structure of the IDS agent is shown in the Figure 2.1. The data collection module is responsible for collecting security related data from various audit sources and preprocesses them to the input format required by detection engines. The detection engine then uses the data to perform intrusion detection tasks locally. A Markov Chain based anomaly detection algorithm is used as the local detection model.

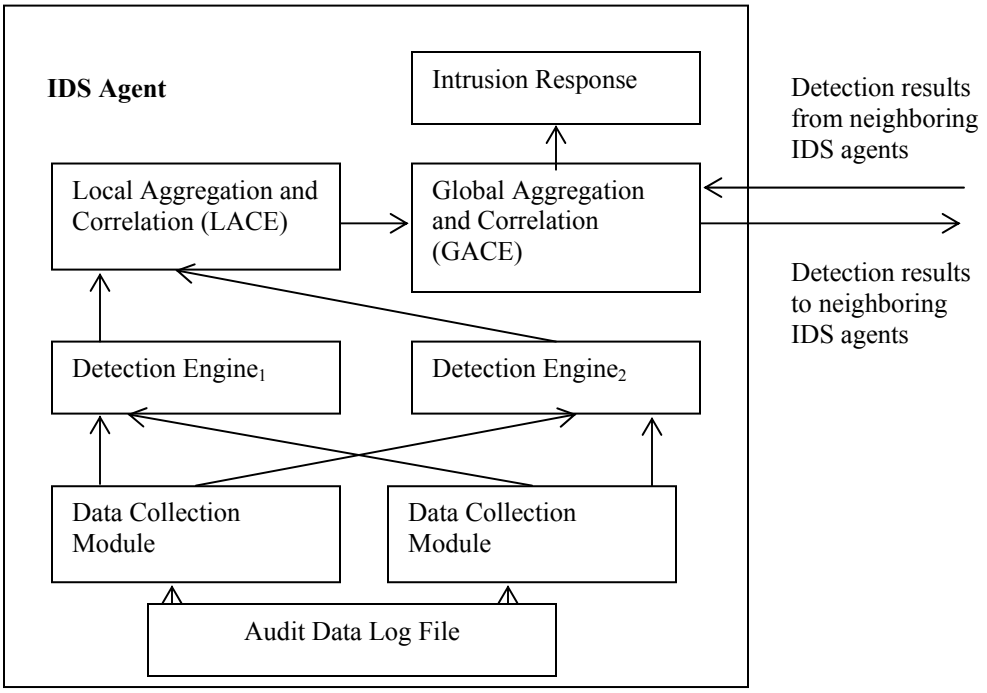


Figure 2.1 Related IDA for MANET -MIDWAN Proposed by Sun et al.

The Local Aggregation and Correlation Engine (LACE) locally aggregate and correlate detection results from different detection engines in the IDS agent. The Global Aggregation and Correlation Engine (GACE) aggregate and correlate the alert information from a wider area in order to make a better decision. The intrusion response module handles the generated alarms.

They have implemented a Markov Chain based anomaly detection approach for MANET IDS. Specifically they construct a Markov Chain from the discretized routing table changes. Using the immediate previous *w* consecutive events (the routing table changes), also known as *from_state,* they predict the transition probability of the next state, *to_state*. This transition probability is then used to calculate the distance and classify the observed activities.

MIDWAN Simulation Experimentation:

Three mobility models, the Random Waypoint model (RW), the Random Drunken model (RD), and the Obstacle Mobility (OM) Model are used by Sun et al., in their performance evaluation experiments. Since a node's moving speed is one of the most commonly used metrics in measuring the performance of MANETs, the speed could be used to reflect MANET dynamics. The larger the moving speed, the more dynamic the MANET is. Hence they have measured the IDS performance under varying speeds.

Simulation was performed by Sun et al., using GloMoSim to evaluate the performance of their IDS using different metrics. The following are the parameters that were used in the simulation. Number of nodes was set to 30 within an area of 1000 x 500 sq m. CBR traffic by source destination pairs are used to generate traffic. The speed was varied from 0 m/s to 50 m/s. At each mobility level, they have run the simulation for 100 minutes to get the normal data, and collect the normal data from all nodes to generate a normal data trace. For each trace they have collected PCR and PCH feature every 3 seconds after a warm up period of 300 seconds.

They then simulate the routing disruption attack scenario in GloMoSim. The following threat model has been implemented in the DSR protocol. In order to effectively disrupt the routing logic, a node compromised by an attacker actively sends falsified Routing REPly (RREP) packets into the network. Purpose here is to effectively disrupt the routing logic of the victims or the whole network. Using RREP packets the attacker induces the victim to form a short path to itself, resulting in routing black-hole attacks.

The metrics that they have used for performance evaluation are:

False Positive Ratio: It is defined as the percentage of decisions in which normal data are flagged as anomalous.

Detection Ratio: It is defined as the total number of detections divided by the number of victims in the anomalous data.

To investigate the impact of different mobility models, they have collected the same amount of training data, test data, and abnormal data at different mobility levels and they have used the same procedure to build the classifier.

MIDWAN Results:

The results that they obtained for the above mentioned metrics is given below. The results of the performance evaluation comparing our proposed model with the related models are discussed in Chapter 5.

False Positive Ratio:

In general as speed increases the false positive rate increases for all the mobility models. Their model resulted in false positive rate of about 10% at 5 m/s but it increases significantly to about 25% at about 20m/s and increase further at mobility levels higher than 20m/s.

Detection Ratio:

They have observed that the detection ratio decreases with the increase of the speed for each of the mobility model used. Their model has detection rate of about 97% until 10 m/s. As speed increases the detection rate slightly falls to 90% at 40 m/s. They have also observed that the overall detection ratio of the RD model is simiar to RW model and OM model.

Related Scheme 2: CIDSAN by Huang and Lee:

CIDSAN model proposes a cluster-based detection scheme where periodically, fairly and randomly a node is elected as the ID monitoring agent for a cluster of neighboring MANET nodes. Compared to a model where each node is its own ID agent, in this scheme the responsibility of detection is shared among nodes in the cluster.

A cluster is defined as a group of nodes that are *close* to each other. The criterion of 'close' is that a node in the cluster, the *clusterhead,* has all other members, known *as citizens,* in its 1-hop vicinity. As a special case, a node that cannot be reached by anyone else forms a single node cluster (SNC). The *size* of the cluster is defined as the number of nodes in the cluster and is denoted as S_c.

It is essential that the clusterhead assignment be fair and secure. By fairness, authors mean that every node should have a fair chance to serve as a clusterhead. Fairness has two components, *fair election*, and *equal service time*. Assuming that every node is equally eligible, *fair election* implies randomness in election decision, while *equal service time* can be implemented by periodical fair re-election. By security the authors mean that none of the nodes manipulate the selection process to increase (or decrease) the chances of it (or another node) being selected as clusterhead. Obviously if randomness of the election process can be guaranteed, then security is also guaranteed.

The authors have also proposed techniques to guarantee the fairness and security of the election process. First, each node *i* contributes a random value R_i to the input. Then a common selection function is used by all nodes to compute an integer from 0 to S_c-1 from a total of S_c inputs. The output of the selection function must have a uniform distribution in [0, S_c-1]. The selection function they use is a simple XOR function. ie., f

(R_0, R_1, R_2, … R_{Sc}-1) =($\oplus$ R_i MOD S_c). XOR function assures that as long as one input is random, the output is random. The random values are fully exchanged within the cluster and the selection function is computed in a distributed manner on each node to decide the cluster head. This ensures that the same cluster head be computed by all cluster members.

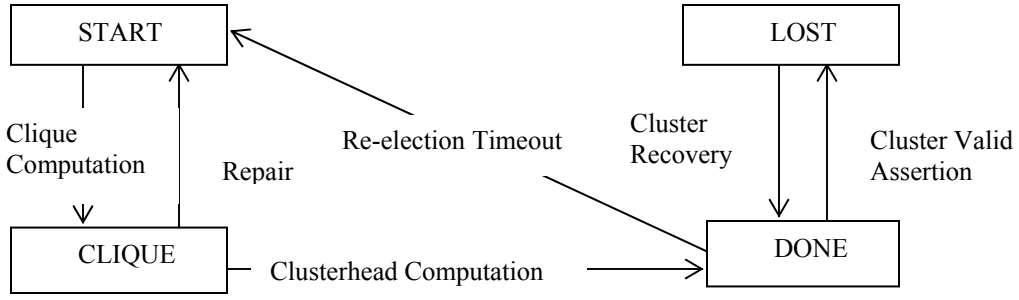


Figure 2.2 Related IDA for MANET - CIDSAN Proposed by Huang et al.,

As shown in the Figure 2.2, all nodes are in an INITIAL state. They temporarily assume themselves as Single Node Clusters, so that they do intrusion detection for themselves, just as in the per-node based approach. Then initial clusterhead setup is performed using clique computation and cluster head computation steps.

Clique Computation Step:

A clique is defined as a group of nodes where every pair of members can communicate via a direct wireless link. It can be noted that the definition of *clique* is stricter than the definition of *cluster*. However, the clique requirement can be relaxed after the clusterhead has been computed. That is, only the clusterhead needs to have direct links with all members. After this step every node is aware of its clique members.

Clusterhead computation Step:

The purpose of this step is to randomly select one node in the clique as the clusterhead based on the techniques explained earlier. Once the clusterhead is determined, it copies the clique member list to its citizen list.

Cluster Valid Assertion Step:

All nodes should perform this step periodically when they are in DONE state. There are two parts in this step.

a. Since the network topology tends to change in MANET, connections between the elected clusterhead and its citizen nodes may be broken from time to time. If a link between a citizen Z and clusterhead H has been broken, Z will check if it is in another cluster. If not, it enters LOST state and activates the Cluster Recovery step. Also, Z is removed from H's citizen list. If there are no more citizens in it list, H becomes a citizen if it belongs to another cluster. Otherwise, H enters LOST state and activates the Cluster Recovery Step.

b. Even if no membership change has occurred, the clusterhead cannot function forever because it is neither fair in terms of service, nor secure. Hence a mandatory re-election timeout, T_r is enforced. Once the T_r expires, all nodes in the cluster enters the INITIAL state and start a new clusterhead setup round.

Cluster Recovery Step:

In the case that a citizen loses its connection with previous clusterhead or clusterhead loses all its citizens, it enters LOST state and initiates the Cluster Recovery Step to re-discover a new clusterhead.

The authors further discuss that since clusters can overlap, a node can belong to multiple clusters. In that case a node in the multiple clusters should perform the Clusterhead Computation Protocol for each of its clusters independently. In the Clusterhead Computation Step, the authors assume that the topology remains static during computation. In mobile environment, this assumption does not always hold. A remedy is for each cluster member to monitor the neighborhood actively. Once a link is broken, a REPAIR message is broadcast by both ends of the link, and all other nodes in the cluster will be aware of that. All nodes in the cluster then re-enter INITIAL state and start a new clusterhead setup round by initiating the Clique Computation step.

The authors also discuss that a malicious node has a $1/S_c$ chance to be elected as the clusterhead. It can then launch certain attacks without being detected. If this chance is not acceptable, multiple rounds of clusterhead computation can be used to elect multiple clusterheads, each running separate IDS to monitor the cluster. In the extreme case, IDS could be run on each node. Thus there is a tradeoff between efficiency and security.

Anomaly Detection:

Authors believe that a strong feature correlation exists in normal behavior, and that such correlation can be used to detect deviations caused by abnormal (intrusive) activities. They have developed a cross-feature analysis anomaly detection approach that explores the correlations between each feature and all other features.

This approach computes a classifier C_i for each f_i using $\{f_1, f_2,\ldots f_{i-1}, f_{i+1},\ldots f_L\}$ where $\{f_1, f_2,\ldots f_{i-1}, f_{i+1},\ldots f_L\}$ is the feature set. C_i can be learned from a set of training data. It predicts the most likely value of f_i based on the values of other features.

They have solved the anomaly detection problem i.e., whether a record is normal or not, as follows. Given a record $x = <v_1, v_2,\ldots, v_L>$, first apply each C_i to compute $p_i(v_i|v_1, v_2,\ldots v_L)$, i.e., the probability of v_i given the values of other features. Then compute the average probability $\Sigma_i\ p_i / L$ and compare it with a threshold. An alarm is raised if it is lower than the threshold because it implies the record is highly unlikely.

The authors have defined a total of 141 features in their study. These features capture the basic view of network topology and routing operations, as well as traffic patterns and statistics. They have conducted experiments using the ns-2 simulator and summarized results for AODV protocol.

In their experiments they have implemented different types of intrusions like sleep deprivation using malicious flooding, denial-of-service, selfishness using packet drop and routing loop using spoofing. They have also designed and implemented rules to detect the attack type in the anomaly detection model. They have used Random Waypoint model as the Node mobility model. They have varied Mobility speed from 5 m/s to 40 m/s. The number of connections has been selected to be 10 within 500 x 1500 sq.m area. They have used AODV used as the MANET protocol.

They have used trace data of normal runs for training the anomaly detection models. They then run the attacks and collect the trace data for evaluating the models. For example, if total running time is 10000 seconds, and the sampling rate by which the feature values are computed is 5 seconds, then the trace data has 2000 data points or events.

Each event is labeled as normal or abnormal according to when and for how long an attack is running (and how long the effect lasts). While evaluating an anomaly

detection model, they compute how many abnormal events are correctly identified (ie. the detection rate) and how many normal events are incorrectly identified as anomalies (ie., false alarm rate).

In their experiments an anomaly detection model computes features and detects anomalies on each node locally. As long as one detector (on one node) identifies an abnormal event, they count it as an anomaly alert (true detection or false alarm) for the MANET. The intuition is that when there is an intrusion, they need to have at least one node that detects the anomaly.

CIDSAN Simulation Experimentation:

The authors conducted experiments with different mobility rates using ns-2. They have used Detection Rate and False Positive rate metrics to compare cluster based and per node based detection models.

CIDSAN Results:

They have found that the detection accuracy metric of cluster based approach (84%) is a little less compared to per node based approach (89%). Also the performance of cluster based approach is more sensitive to mobility compared to per node based approach. This is due to the reason that in a highly dynamic scenario, it is more likely that the correlation among route and location related features are not as regular as that in a per-node based scheme. However, even under a reasonably high mobility level (40 m/s), the detection rate is still higher than 80%. Similarly, false positive rate metric of cluster based approach (5%) is a little more compared to per node based approach (less than 1%). It has to be noted that the detection and false positive rate metric results for all their simulated intrusions follow the similar patterns.

2.6 Limitations of the Existing Related Schemes

Limitations of existing Mobile Ad hoc Network security schemes:

The scheme GDH needs further exploration of mechanisms for the detection and resolution of inconsistent certificates, improvement of certificate graph models and making use of existing PKI infrastructure [40]. Scheme MDA does not provide authentication of the participants. In addition, more formal arguments need to be developed to support optimality claims [41]. Unicast approaches by the scheme MOCA only exploit information in the local routing cache. One potential extension is to let a node browse into neighboring nodes' routing tables. For example, a node may be short of one or two cached routes and that would lead to flooding. If the node has a way to peek into the neighbors' routing tables and find a couple of new cached routes, it can avoid flooding. Potential overhead for this approach would be the extra communication required between neighbors to exchange the information in routing tables. Whether the benefit would surpass the overhead is an interesting question to investigate [42]. All the unicast based approaches in the MOCA protocol do not take into account the direction of Certification REQuests (CREQs). At a worst case, all the MOCAs picked by its unicast approach could reside on one side of the network from the requesting node. Then it is possible that all the CREQs are sent into one direction sharing the same next hop nodes, potentially causing unnecessary contention. This leads to a failure or at least delayed responses. One possible solution for such a scenario is to utilize the next hop field in the cached routing table entries. For example, by selecting a set of MOCAs with all the different next hops, one can expect to have a spatial load balancing effect in that each CREQ will go out in different directions [42]. The scheme CORE considers only attacks

from selfish nodes but not from active intruders. Hence the scheme needs to be extended and tested for intruder attacks as well. Also there is no definition of formal method to analytically prove robustness of CORE [43]. The solution for attack by selfish nodes, presented in Nuglets model is focused just on packet forwarding attacks. This model also does not address application-level issues like mutual provision of information services in an ad hoc network [39]. The CONFIDANT protocol assumes that nodes are authenticated and that no node can pretend to be another in order to get rid of a bad reputation [44]. The Guardian Angel model is not a comprehensive security scheme and does not take into account the attacks like packet forwarding and denial of service or routing attacks [45].

The SEAD approach does not incorporate mechanisms to detect and expose nodes that advertise routes but do not forward packets [47]. In the Beacon scheme, scalability is an issue if there are large numbers of nodes compared to the available bandwidth. The proposed model assumes all nodes in a network share a symmetric key used only for beacon authentication. In addition to problems with scalability, every agent and mobile node at the site has to know the network authentication key. The symmetric keys might be replaced with public key cryptography. Public-key signature and verification of beacons and Mobile-IP registration messages is feasible, even though transmitting such a signature requires more link bandwidth. Every node can possess its own key and simply sign its beacons and registrations. The distribution of certificates such that mobile nodes and agents can verify a beacon is again a higher-level problem [48]. SOS model provides fully localized design, easy support of dynamic node membership, limited intrusion tolerance capacity and decreasing overhead over time. While these characteristics are

appealing, this scheme also has limitations as this is achieved at the increased computational overhead (associated with asymmetric cryptography primitives) compared with other hash function based designs [49]. In the TRUST model when a new node enters the system, it assumes that the node already has an initial certificate. This results in the problem of registering users. Also when two ad hoc networks merge, this model does not provide mechanisms for nodes originated from different networks to certify and authenticate each other [50]. In SRP model, fair utilization of network resources is an issue. Possible ways to dismay nodes from broadcasting at the highest possible rate is still an issue [36]. Since the ARIADNE model does not possess the optimizations of DSR, the resulting protocol is less efficient than the highly optimized version of DSR that runs in a trusted environment [51]. An important aspect of OSRP scheme is that the algorithm can be used to detect a fault. However, it is difficult to design such a scheme that is resistant to a large number of adversaries. The method suggested in this paper uses a fixed threshold scheme. This scheme does not explore other methods, such as adaptive threshold or probabilistic schemes which may provide superior performance and extensibility. Also this scheme does not provide means of protecting routing against traditional denial of service attacks [35]. The Watchdog and Pathrater model assumes that there are no apriori trust relationships. Performance of model is bound to suffer when trusted node lists in ad hoc networks are also taken into account. Also, in this model, all the simulations are based on Constant Bit Rate (CBR) data with no reliability requirements. The analysis should be extended to explain how the routing extensions perform with TCP flows common to network applications [53].

Limitations of existing Intrusion Detection and Intrusion Response Approaches:

The misuse detection systems use patterns of known attacks to match and identify those intrusions [66]. Although it can accurately and efficiently detect instances of known attacks, it lacks the ability to adapt in detecting new type of attacks. The anomaly detection systems on other hand detect intrusions by finding deviations from the established user profiles. Anomaly detection should detect new types of intrusions but it could have higher false positive rate [67]. Traditionally, IDA are developed using expert knowledge of the system and attack methods [68]. Due to the complexity of modern network system and sophistication of attackers, expert knowledge engineering is often very limited and unreliable [18]. Some IDA schemes are very sensitive to the data representation. For instance, these schemes may fail to generalize an unseen data if the representation contains irrelevant information. In some instance, it has been observed that training of IDA requires a noise-free data (the data that is labeled 'normal') [63]. It has been observed that the existing IDA performs poorly in detection as well as the false positive rates at higher mobility rates [58, 64, 72]. It has recently been observed that Denial of Service (DoS) attacks are targeted even against the IDA [18]. Thus, IDA themselves needs to be protected. An IDA should also be able to distinguish an attack from an internal system fault.

The identification of intruder and appropriate response techniques to protect MANET still represents a challenging issue. The need to coordinate intrusion detection and response techniques and the need to respond and control the identified attacks effectively, require further research. It can be noted that though the response concepts are explained in the existing intrusion detection models, implementation details and results

for the response framework are not provided to demonstrate and validate their response techniques. Also according to our literature review, we observe that none of the existing models has proposed an intrusion control approach for MANET, such that detection and response are done continuously to protect the MANET.

To summarize, the related existing MANET intrusion detection and intrusion response approaches suffer from one or more of the following limitations:

- Lower detection rate when mobility is used as a parameter.
- Higher false positive rate when mobility is used as a parameter.
- Appropriate response techniques to protect MANET after threat detection.

The current schemes thus have practical problems in intrusion detection and real time response. The proposed Intrusion Detection and Response model for Mobile Ad hoc Networks (IDRMAN) security model addresses many of these limitations. To demonstrate that the proposed approach does have these features and performs better in terms of the above proposed metrics, we conducted performance evaluation experiments comparing the proposed approach with the related existing intrusion detection approaches using these metrics. These experiments and results are explained in Chapter 5.

CHAPTER III

IDRMAN SECURITY MODEL – DETECTION FRAMEWORK

3.1 Overview of IDRMAN

Intrusion Detection and Response model for Mobile Ad hoc Networks (IDRMAN) detects an attack and responds to the detected attack. An attack is detected by identifying the set of network parameters that will be affected during an attack, identifying the thresholds for these parameters, continuously monitoring these parameters, quantifying the network vulnerability by applying fuzzy logic on the measured values of the network parameters and comparing the quantified values against the reference thresholds of the threat detection metric. The set of network parameters that will be affected during an attack are referred to as significant parameters and are identified using data mining techniques. The thresholds for these significant parameters are identified using Six Sigma methodology. The network vulnerability level is quantified into a numerical value called the Threat Index (TI) computed using Fuzzy logic. Based on the threat level indicated by TI, the response mechanism is invoked (which is explained in Chapter 4) which then identifies the intruder and responds to the attack.

This chapter is organized as follows: The rationale and the architecture of the IDRMAN model are presented in Sections 3.2 and 3.3 respectively. Sections 3.4 through 3.6 explain the IDRMAN detection framework in detail. Mathematical analysis to validate TI metric is presented in Section 3.7.

3.2 Rationale of IDRMAN

The IDRMAN model seeks to detect and respond to intrusive behavior by identifying and analyzing network parameters that deviate from a predicted norm during an attack. It is a Statistical-Based Intrusion Detection (SBID) system which is based on the premise that intrusions can be detected by inspecting a system's audit trail data for anomaly, and that the audit trail data is noticeably different when an intrusion occurs. Before unusual activity can be detected, SBID systems require a characterization of system activity that is considered "normal". Any sequence of system events deviating from the “normal” profile by a statistically significant amount is flagged as an intrusion attempt [75]. Based on this, in our model, we identify network parameters which are affected during attacks, and characterize their values when the system is normal without any attack. Our philosophy is that, by identifying and continuously monitoring critical network parameters that are affected by various types of attacks, we could measure the relative change in parameter values from the “normal” values and detect an attack. Once an attack is detected, proper level of protection measures could be applied and hence, nodes causing these attacks could be blocked from accessing the system or the network.

To strengthen our premise of using network parameters as the audit data to detect and respond to intrusion, we explain three different attacks and their effect on the network parameters in the following paragraphs of this section.

DoS attack - Bombardment of packets by an intruder node on MANET host node:

In a type of DoS attack, an intruder bombards packets on a MANET host node servicing multiple mobile nodes. In this attack model, the intruder creates huge traffic in the link between the intruder and the host resulting in the quick exhaustion of the

MANET hosts' precious resources. This kind of DoS attack results in the inability of the host node to serve the other genuine nodes in MANET fairly.

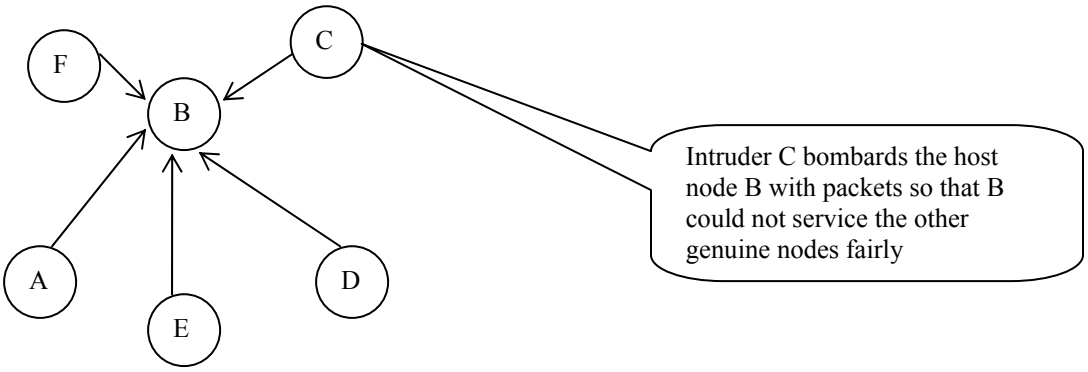


Figure 3.1 DoS Attack

DoS attack is depicted in Figure 3.1, where node B is a host node and C is the intruder. The intruder node C creates a huge traffic resulting in the exhaustion of the node B's resources. This results in the inability of node B to serve genuine nodes A, D, E and F fairly. Thus, DoS attacks on the mobile ad hoc networks can lead to network performance degradation. Some of the critical parameters with respect to MANET that are affected by this type of DoS attacks are:

Packet Drop: Due to DoS attacks, packets in the link may be dropped due to exhaustion of the hosts' resources.

Queue Length: The inability of the host to service the request from other nodes because of DoS attacks results in increase in queue length on the links between the host and other nodes.

Energy consumption: The bombardment of packets due to traffic and servicing them result in the consumption of significant battery power (a constrained resource in MANET) in the link between intruder and host.

Active attack: 'Poisoning' routing table by intruder causing routing loop:

In this kind of attack, an intruder injects incorrect routing information, which in turn poisons the routing table or protocol [89]. The intruder may also update the routing table to create a loop, so that packets traverse in the network without reaching the destination [84].

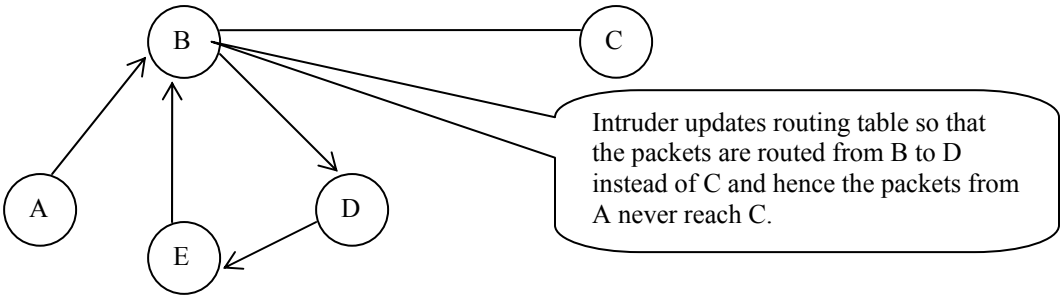


Figure 3.2 Routing Loop Attack

In the MANET shown in Figure 3.2, let us assume that packets are supposed to traverse from source node A to destination node C. However, the intruder updates the routing table so that the packets traverse from B to D instead of C, and hence the packets from A never reach C. This also causes congestion on domains served by nodes A, D and E, due to the bombardment of packets whose actual destination was C. Thus the attack can lead to network performance degradation. Some of the critical parameters of mobile ad hoc networks that are affected by this type of active routing attacks are:

Throughput: Due to the poisoning of routing protocol or table, packets may never reach the destination. Thus, there is a relative increase in the number of lost packets from the genuine nodes during the attack.

Total number of packets dropped due to no routing information: Due to poisoning of the routing protocol or table, packets may be dropped from the network for want of routing information. The drop count of packets from genuine nodes increases due to the attack.

Passive attack: Packet discarding by selfish nodes:

It has been demonstrated that statistically over 60% of the resources are taken up by packet routing function at the nodes [6]. If a node has turned selfish, it does not route to conserve energy, thus affecting other nodes on the ad hoc network. A selfish node simply does not perform its intended function of forwarding the packet to a proper destination node and routes all packets to itself and later discards them. The motivation in these attacks by selfish nodes is to save significant battery power, instead of performing networking functions such as packet forwarding and routing.

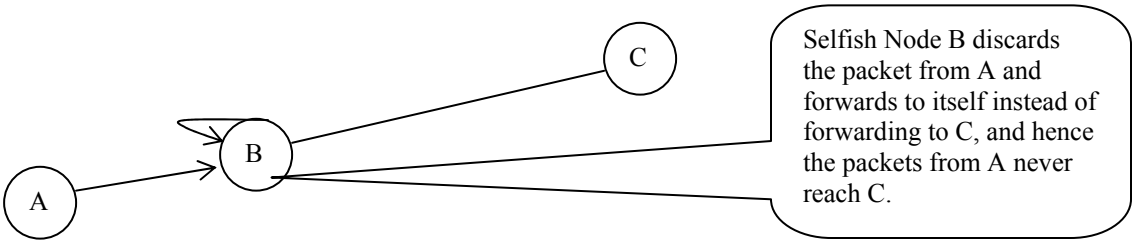


Figure 3.3 Packet Mistreatment Attack

As shown in Figure 3.3, the packets are supposed to traverse from source node A to destination node C. However, selfish node B discards the packets from A and hence the packets from A never reach C. This results in 'black hole' attacks. This in turn may also result in DoS or deadlock issues that result in performance degradation. Some of the critical parameters that are affected by this kind of passive attack with respect to ad hoc networks are:

Throughput: Due to packet mistreatment by selfish nodes, packets do not generally reach host nodes. This results in packet loss, and hence a significant decrease in the measurement of throughput for the destination host nodes.

Packet drop rate: Packets are discarded by selfish nodes and hence there is a significant increase in the packet drop rate for the collaborating selfish nodes in ad hoc network.

Energy consumption: Since battery power is a constrained resource in mobile ad hoc networks, the selfish nodes discard packets to save battery power by means of techniques like sleep deprivation [6]. Hence, there will be a relative decrease in the measurement of power or energy consumed for group of selfish nodes within the network.

These examples of different attacks clearly indicate that for both active and passive attacks, the network parameters are significantly affected and deviate from their normal levels. Thus identifying the appropriate network parameters for each attack, and continuously monitoring them will be an effective method to detect and respond to intrusions.

3.3 Architecture of IDRMAN

The proposed IDRMAN model uses a feedback control scheme that is analogous to the human biological model wherein an attack is detected by measuring body parameters like temperature, blood sugar and blood pressure level and comparing them against their normal values. Once an attack on the body is detected, it is treated to bring the body to the normal state. Similarly, in this security model various parameters of an ad hoc node or a set of ad hoc nodes are monitored. If these parameters change rapidly in a given time frame, the appropriate threat is detected, intruder is identified and a corrective action is taken. The challenge here is the identification of critical parameters and their threshold values to predict the intrusion accurately and efficiently in mobile ad hoc networks.

The basic framework of the model is shown in Figure 3.4 and is explained as follows.

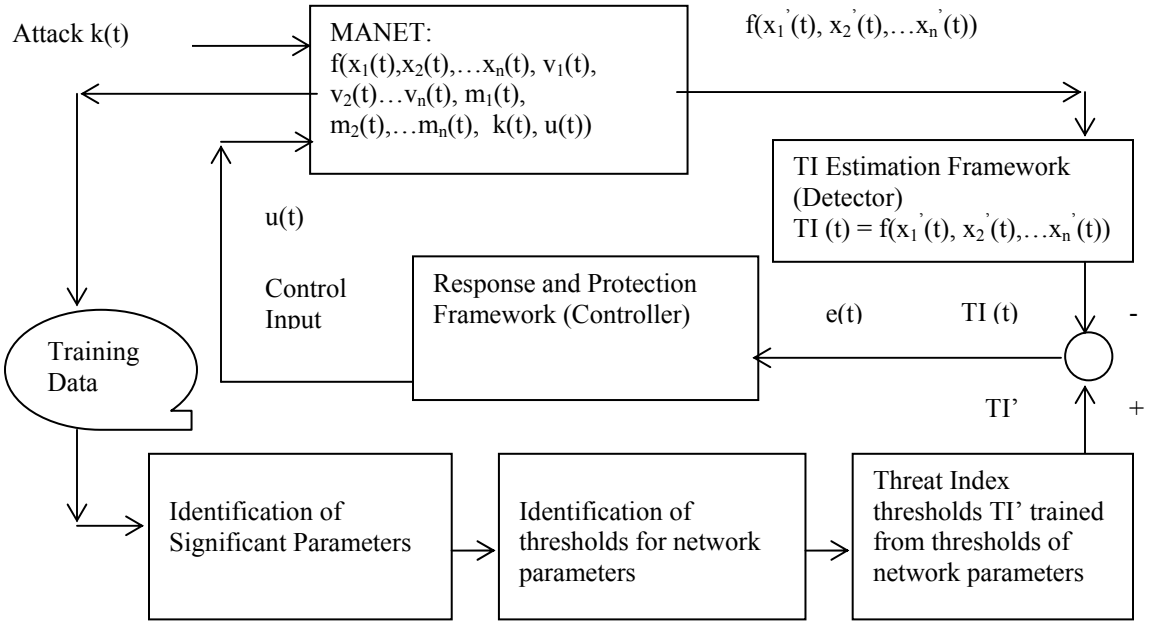


Figure 3.4 Feedback Control Model of IDRMAN

In the Figure 3.4, MANET is represented as a function: $f(x_1(t), x_2(t),\ldots x_n(t), v_1(t), v_2(t)\ldots v_n(t), m_1(t), m_2(t),\ldots m_n(t), k(t), u(t))$, where $x_n(t)$ represents the significant attack sensitive network parameters, $v_n(t)$ represents the network parameters which are not significant in representing the node vulnerability, $m_n(t)$ represents the mobility parameters, k(t) represents the attack and u(t) represents the control input. $x_n'(t)$ represents the modified values of the significant attack sensitive network parameter due to the influence of the attack k(t) and the control input u(t). TI for a node is calculated by the detection framework from the attack sensitive network parameters, $x_n'(t)$ using fuzzy logic. The computed Threat Index TI(t) is compared with the threshold values of the Threat Index TI'. The Threat Index thresholds (TI') are obtained with the help of the training dataset where the state of each record is labeled. Data records collected from

simulation environment with and without attack are used as training dataset for identifying the Threat Index thresholds. As shown in Figure 3.4, the training data is derived from the MANET and is used in the identification of significant parameters and the thresholds of these parameters and the threat index. If the computed TI(t) of a node is greater than or equal to vulnerable state threshold reference TI', the node is identified to be under threat. Upon detecting that a node is under threat, the neighboring nodes are subjected to the response and protection algorithm in the response framework. This response algorithm identifies the intruder and sends the control signal u(t) to isolate the intruder from the MANET. The control signal u(t) varies depending upon the type of the intrusion. The different types of control actions are explained in Chapter 4. This control signal reconfigures the MANET and modifies $f(x_1'(t+1), x_2'(t+1),\ldots x_n'(t+1))$ such that TI(t+1) reaches the steady normal state. It should however be noted that $f(x_1'(t+1), x_2'(t+1),\ldots x_n'(t+1))$ also depend on any new attack k(t+1).

Since IDRMAN is online control based model, we need to determine the optimal response time in order for the online controller to work properly. The desired minimum response time can be estimated based on the various computation tasks in the intrusion detection algorithm and intrusion response algorithm. Based on the time complexity analysis, the intrusion detection and response algorithm's time complexity is turned out to be $O(N^3)$. The response time also depends on the speed of the processor. If a processor execute 1 billion instructions per second, for $O(N^3)$ algorithm, it will take one second for N = 1000. Here N indicates that there are 1000 nodes and each node has 1000 neighboring nodes and there are 1000 parameters to analyze. Thus the minimum response

time threshold in the model can be set depending on number of Nodes in MANET and the number of significant parameters and speed of the processor.

The proposed security model is distributed and cooperative, where every node in the wireless mobile ad hoc network participates in intrusion detection and response. Each node is responsible for detecting signs of intrusion locally and independently, but neighboring nodes can collaboratively investigate in a broader range. The proposed model works by propagating the intrusion detection state information among neighboring nodes and the local node as depicted in the Figure 3.5.

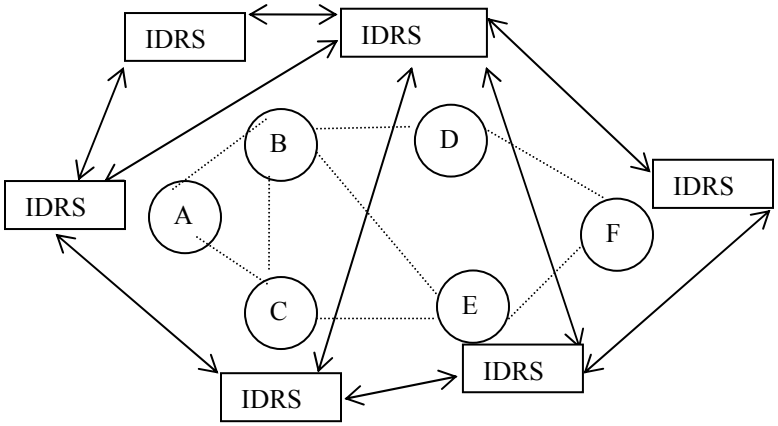


Figure 3.5 Distributed Intrusion Detection and Response System (IDRS)

The need for the cooperative and distributed intrusion detection/response arises because mobile ad hoc networks are dynamic and they typically lack a central entity. Hence, each node responds based on the intrusion reports from other nodes in a distributed manner [90].

3.4 The IDRMAN Detection Framework Mechanism

The Intrusion Detection Metric -Threat Index:

The IDRMAN detection framework quantifies an attack or vulnerability with a metric known as Threat Index (TI). Threat Index (TI) is the metric used by the detection framework to detect if the node is under attack or not. Threat Index is a number, which takes values between 1 and 10. When there is no attack, the network is in the normal state (NS) and this is indicated by the TI range from 1 to 4; when there is an attack, the network is in the vulnerable state (VS) and is indicated by the TI range from 7 to 10; the intermediate state of the network between normal and vulnerable state is referred to as the uncertain state (US) which is indicated by the TI range from 4 to 7. The TI thresholds, (4, 7) for the uncertain and vulnerable state are obtained by means of TI threshold training algorithm on the training data. TI at any instant of time is computed by applying fuzzy logic on the measured values of the significant parameters at that time. By comparing the computed TI with the TI thresholds, the node is classified as being in the normal, uncertain or vulnerable state and this classification detects the attack.

The Threat Index resembles as well as differs from the metric known as Vulnerability Index (VI), which is proposed in the literature to represent the state of the wired Internet node. VI metric was proposed by Qu et al., in their work to detect the vulnerability of the Internet node [96]. TI resembles VI in that both analyze and quantify the vulnerability of a node/network. They both detect attack points in the network and examine how critical network components behave during an attack. Both TI and VI are calculated through network metrics and system state characterization of those metrics by means of threshold values. However, they are different in that, while fuzzy logic is used

to calculate TI, VI is evaluated using max-plus computing, minus-plus and division-plus computing vector functions. Using fuzzy logic for TI aids the multivariate input analysis by easily and readily combining inputs from widely varying sources [102]. Also, TI is applied to evaluate vulnerability for the MANET node, while VI is applied to evaluate vulnerability of the node in Internet. As illustrated in Figure 3.4, TI is controlled through feedback control mechanism by referencing the actual TI value calculated by the estimator (detector) with the TI threshold value TI', which is not the case in VI. Another point for difference between TI and VI is that, for TI, significant attack sensitive network parameters are identified through classification trees methodology with the help of training dataset while for VI the selection of metrics is arbitrary and is based on user experience.

Steps in Threat Index Computation - Theory and Illustration:

Each step of the threat detection framework, whose objective is to calculate the Threat Index to detect an attack, is explained below with theory and illustration.

Step 1: Log/Data Fetch:

In this step, the raw ad hoc network data is collected from MANET and fed to the detection framework on a real time basis. This step pre-processes the collected data from MANET to make it suitable for threat index evaluation as well as intruder identification. This framework fetches data for both training as well as testing. The training data is used to identify significant parameters and thresholds of those parameters. The testing data is used to perform the actual detection based on the training.

Step 2: Identification of Significant parameters/Threat metrics:

This section provides the theoretical foundation of using data mining to identify the significant parameters for the IDRMAN. These significant parameters are used in both the detection and response framework in our model [97]. The measured values of these significant parameters are used to calculate TI in the detection mechanism and to identify the intruder in the response mechanism.

Traditionally, Intrusion Detection Approaches (IDA) are designed using expert knowledge of the system and attack methods [10]. Due to the complexity of modern network systems and sophistication of attackers, expert knowledge engineering is often very limited and unreliable [12]. One of the problems in the existing IDA is the computational overhead, which sometimes can be unacceptably excessive. Analyzing system logs requires large memory and processor resources. Usually an IDA is trained over this huge amount of system audit information, which increases the complexity of intrusion detection algorithms dramatically [11]. Long term training and testing is not suitable for the requirements of real-time detection and response, which may further limit the practical use of an IDA. The problem gets worse when the input audit information involves a high dimensionality of the data. Most of the existing algorithms assume a low dimensionality of the data. Another issue is the noise-tolerance performance of IDA. Some IDA schemes are very sensitive to the data representation. For instance these schemes may fail to generalize an unseen data if the representation contains irrelevant information. In some instance, it has been observed that training of IDA requires a noise free data (the data that is labeled ‘normal’) [10]. An IDA should also be able to distinguish an attack from an internal system fault.

Thus, current IDA has practical problems in intrusion detection and they are also limited in the dimension of the input variables and in selecting qualitative and quantitative variables that can be used to predict the intrusion efficiently and accurately. To address the limitations of the existing approaches and to overcome the challenges in efficiently detecting the attacks, there is a need for intrusion detection approach based on statistics and machine learning concepts. CART is one such tool based on the data mining concepts. Data mining is defined as "the nontrivial extraction of implicit, previously unknown, and potentially useful information from data [95]." Data mining automatically sifts large complex data, searching for and isolating significant patterns and relationships. The discovered knowledge is then used to generate reliable predictive models for detecting threats and attacks [10]. Data mining could be integrated with the intrusion detection system very well [12]. Many existing intrusion detection systems are constructed by manual encoding of expert security knowledge that makes the changes in these systems slow and expensive [15]. A data-mining framework can be used for adaptively building intrusion detection systems by utilizing the auditing programs to extract an extensive set of features that describe each network connection or host session. Data mining approaches like CART can be used to learn the rules that accurately capture the behavior of intrusions and normal activities. These rules can be then used for intrusion detection. The association rules and the frequent episodes that are computed from audit data can be used to uncover important attack sensitive parameters. CART is also an excellent tool in being pre-processing complement to other data analysis techniques. For example, CART's outputs (predicted values) can be used as inputs to

improve the predictive accuracy of fuzzy logic, neural nets, threshold evaluation and logistic regression.

The main application of CART and data mining in the proposed model is to identify the significant attack sensitive network parameters from the wealth of raw network data. Due to CART, our model performs the variable selection and the intrusion detection efficiently with low false positive rate and with less processing, training and testing time. Due to the identification of significant attack sensitive network parameters, the results indicate that the model performs better as the dimension of the input data decreases, without compromising detection accuracy, a feature essential for on-line real time detection of the resource constraint networks [84].

Analysis of experimental results using the DARPA benchmark dataset shows that the CART approach performs better compared to related models like random projection and principal component analysis. Experimental results indicate that the proposed security model using the CART methodology can be used to identify the significant parameters, which then can be used for evaluating vulnerability and detect intrusion out of raw network data efficiently [84].

In our model, the number of significant parameters used in the threat detection and intruder identification is decided by the variable importance tool in CART. If the variable importance for a network parameter is identified as 100%, it is selected for the threat detection and intruder response. For instance, in our experimentation, PL, QL and EC are selected since their importance is identified as 100% in the CART variable importance table. Variable importance is based on the contribution of predictors during the construction of the tree. Importance of a variable is determined by playing a role in

the tree construction, either as a main splitter or as a surrogate. The following paragraphs explain in detail, the concept of variable importance identification from a dataset using classification trees with an example.

Classification Trees:

Let $Y_{1,}Y_{2,...}Y_{n,}$ O be random variables where Y_i has domain Dom(Y_i) and O has domain Dom (O) = {1,…,J}. Here $Y_{1,}$…, Y_n are the predictor attributes and n is the number of predictor attributes. O is the class label. J denotes the number of class labels, which are 1, 2,…,J. i.e, J represents the set of possible values that the class label, O can have in its domain, Dom(O). It does not have any relation to any other subscript used. C, denotes a classifier function which is simply a mapping from inputs to outputs. However, we restrict the classifier function to have a particular structure as explained below:

Classifier C is a function, C: Dom(Y_1) x … x Dom (Y_n) → Dom (O). Let S = Dom(Y_1) x … x Dom (Y_n) x Dom (O) be the set of events. i.e. it is the set of allowable input-output pairs. Here x represents the cartesian product or cross product. Thus, S is the cartesian product of the domains of all inputs and output. This kind of formalization allows one to explain what happens in the input-output space. The underlying assumption is that classification generates datasets according to probability distribution μ over the set of events S.

To explain the function C and probability μ, with respect to the simulation scenario used in the thesis, let us consider the dataset, D that has the records sourced from the MANET subjected to security attack as shown in Figure 5.3. The network parameters in the dataset form the random variables, $Y_{1,}Y_{2,...}Y_n$. Here domain Dom(Y_i) represents the range of possible values of the MANET parameter, Y_i. $1<i<n$, is the number of

predictor attributes and $Y_{1,}\ldots,Y_n$ represents the predictor attributes. Let O be the class label that represents the attack. Hence Dom(O) contains range of possible predicted attacks. For example, let $Y_{1,}Y_{2,}Y_{3,}Y_4$ be four MANET parameters that denotes Number of collisions, Queue Length, Energy Consumption and Packet Loss respectively. Let O be the class label that represent an attack say DoS, Masquerade, Authentication etc.,. Then the set of events, S can be represented as Dom(QL) x Dom(NC) x Dom(EC) x Dom(PL) x Dom (Attacks). In this scenario, function C takes the network parameters, Y_i as the input to predict the output attack, O. The classifier function C for this MANET scenario can be represented mathematically as

C: Dom(QL) x Dom(NC)x Dom(EC) x Dom(PL)→ Dom (Attacks). (3.1)

In this MANET scenario, the probability μ over the set of events, S: Dom (QL) x Dom (NC) x Dom (EC) x Dom (PL) x Dom (Attacks) represents the likelihood that these MANET parameters would predict the output class, i.e the attack using the classifier C.

For a given classifier, C, and a given probability measure, μ over S, the Misclassification rate (generalization error) of the classifier C is a function: $MC_\mu(C)=\mu[C(Y_1,\ldots,Y_n) \neq O]$. For a given training dataset, D, of N independent identically distributed samples from S, sampled according to probability distribution μ, a classifier C, is a function that minimizes the misclassification rate (generalization error) function: $MC_\mu(C)$ [87].

Misclassification Rate, $MC_\mu(C)$ could be evaluated through test sample estimate and resubstitution estimate explained as follows. Let us divide dataset D into D_1 and D_2. Let C be the classifier constructed using the training dataset D_1, and let D_2 be an

independent testing dataset with N tuples. Using these assumptions, we can quantify the misclassification rate $MC_\mu(C)$ of C by means of resubstitution estimate $R(C, D_2)$ [126].

Let d_i be a tuple such that $d_i \in D_2$ and $d_i = \{Y_{i1}, Y_{i2}, \ldots Y_{im}\}$. Here Y_{ij} are the random variables that make up D_2. Let $O_{di} = C(d_i)$ be the class label predicted by C for d_i. If the true class O'_{di} of d_i is different from O_{di} then d_i is said to be misclassified by C. Then resubstitution estimate $R(C, D_2)$ of C with respect to D_2 is the fraction of the number of tuples in D_2 misclassified by C.

$$\text{Thus, } R(C, D_2) = |\{d_i \in D_2 \text{ and C misclassifies } d_i\}| / |D_2| = 1/N \times \sum_{i=1}^{N} 1_{\{c(d_i) \neq O'_{d_i}\}}$$

where $1_{\{A\}} = \{1 \text{ if A is True; } 0 \text{ if A is false}\}$.

$$\text{Then } MC_\mu(C) = R(C, D_2) = 1/N \times \sum_{i=1}^{N} 1_{\{c(d_i) \neq O'_{d_i}\}} \quad (3.2)$$

As an example, let there be 100 records in Testing Dataset D_2, whose fields consists of DoS attack sensitive parameters like QL, NC, EC, PL and other network parameters. Let the classifier C, trained using Dataset D_1 identify 4 records as Normal when the true label is DoS. Similarly let C identify 4 records as DoS when the true label is Normal. Then as per equation 3.2,

$$MC_\mu(C) = R(C, D_2) = |\{d \in D_2 \text{ and C misclassifies } d\}|/|D_2| = 1/N \times \sum_{i=1}^{N} 1_{\{c(d_i) \neq O'_{d_i}\}} = 8/100.$$

In general, the classifier construction problem is very hard to solve if we allow the classifier to be an arbitrary function. Arguments rooted in statistical learning theory [116] suggest that we have to restrict the function of classifier to have a particular structure in order to solve this problem. Hence, we restrict the function to a decision tree structure.

A decision tree is a special type of classifier. It is a directed acyclic graph, T, in the form of a tree. The root of the tree, Root(T) does not have any incoming edges. Every other node has exactly one incoming edge and may have zero or more outgoing edges. A node ∓, without outgoing edges is a leaf node, while those with outgoing edges are called internal nodes. Each leaf node is labeled with one class label, while each internal node, ∓, is labeled with one predictor attribute $Y_{T,}$ where $Y_T \in \{Y_{1,\ldots,} Y_n\}$ is called the split attribute. Let the label of the node ∓ be denoted by Label(∓). Each edge (∓, ∓') from an internal node ∓, to one of its children ∓' has a predicate $q_{(∓, ∓')}$ associated with it, where $q_{(∓, ∓')}$ involves only the splitting attribute Y_T of node n. A set of predicates Q is non-overlapping, if the conjunction of any two predicates in Q evaluates to false. A set of predicates Q is exhaustive if the disjunction of all predicates in Q evaluates to true. Let the splitting predicates of ∓ are the set of predicates $Q_∓$ on the outgoing edges of an internal node ∓. Splitting criteria of ∓, denoted by crit(∓), is the combined information of splitting attribute and splitting predicates. For a given decision tree T, we can define the associated classifier in the following recursive manner:

$$C(y_1,\ldots, y_n, ∓) = \begin{cases} \text{Label } (∓); \text{ if } ∓ \text{ is a leaf node.} \\ C(y_1,\ldots, y_n, ∓_j); \text{ if } ∓ \text{ is an internal node} \end{cases} \quad (3.3)$$

and $∓_j$ is children node of ∓, y_i is label of ∓, and $q_{(∓, ∓j)}(y_i) = \text{true}$

$$D_T(y_1\ldots, y_n) = C(y_1,\ldots, y_n, \text{Root}(∓)) \quad (3.4)$$

As per the above definitions, if the tree T is a well-formed decision tree, then the function $D_T()$ is also a well defined classifier which can be called as a decision tree classifier or classification tree. Thus, for a given dataset $D = \{\omega_1,\ldots, \omega_N\}$, where ω_i are

independent identically distributed random samples from a probability distribution μ over S, a classification tree T that minimizes the misclassification rate, $MC\mu(D_T)$ needs to be constructed.

Following Figure 3.6 illustrates the classification tree induction schema.

BuildTree(I/P: Node Ŧ, dataset D, split selection method, O/P: classification tree T for D).
1. Apply V to D to find the split attribute X for node Ŧ.
2. Let n be the number of children for Ŧ.
3. if (Ŧ splits)
    Partition D into $D_1,\ldots, D_n$ and label node Ŧ with split attribute X.
    Create children nodes $Ŧ_1,\ldots,Ŧ_n$ of Ŧ and label the edge(Ŧ, Ŧi) with predicate $q(Ŧ, Ŧ_i)$
    for each i Є{1,…n}
        BuildTree(Ŧi, Di, V)
    end for each
  else
    Label T with the majority class label of D

Figure 3.6 Classification Tree Induction Schema Algorithm

A classification tree is usually constructed in two phases. In the first phase - growth phase, an overly large classification tree is constructed from the training data. Most classification tree construction algorithms grow the tree top-down in the greedy way as shown in the above algorithm, which takes split selection method as an argument. The most popular split selection method that has been widely used for the classification tree construction is the Gini index rule.

Gini rule is used as a measure of how well splitting rule separates classes in the parent node. Gini index, originally proposed by Breiman et al is given by the equation 3.5 [88].

$$GINI(t) = 1 - \sum_{j=1}^{n} [P_{j/t}^2] \tag{3.5}$$

Where $P_{j/t}$ is the relative frequency of class j at node t; this index measures the impurity of node and has a maximum value, when records are equally distributed among all classes, which implies the least interesting information. The minimum value of (0.0) indicates all records belong to one class, implying the most interesting information. The equation 3.6 computes the quality of split, when a node p is split into k partition (children).

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i) \tag{3.6}$$

Here k is number of children nodes, n_i is the number of records at child i and n is the number of records at node p.

In the second phase, pruning is used to generate a sequence of simpler and simpler trees, each of which is a candidate for the appropriately-fit final tree. In the pruning phase, the final size of the tree T is determined with the goal to minimize an approximation of misclassification rate, $MC_{\mu}(D_T)$ where D_T is a decision tree classifier or classification tree and μ is a probability distribution and is calculated as defined earlier.

Pruning is the process of removing leaves and branches to improve the performance of the decision tree when it moves from the training data (where the classification is known) to real-world applications (where the classification is unknown -- it is what one is trying to predict). The tree-building algorithm makes the best split at the root node where there are the largest number of records and hence, a lot of information. Each subsequent split has a smaller and less representative population with which to work. Towards the end, idiosyncrasies of training records at a particular node display patterns that are peculiar only to those records. These patterns can become meaningless

and sometimes harmful for prediction if one tries to extend rules based on them to larger populations. As in our example, say the classification tree is trying to predict the attack type and it comes to a node containing one record ‘X’ with an anomaly and several other records. Classification algorithm can decrease the diversity at that node by a new rule that states that “records like ‘X’ are tall and thus classify the training data. In a wider universe this rule is not really helpful. Pruning methods solve this kind of problem. They let the tree grow to maximum size, and then remove smaller branches that fail to generalize. Also, since the tree is grown from the training data set, when it has reached full structure it usually suffers from over-fitting (i.e. it is "explaining" random elements of the training data that are not likely to be features of the larger population of data). This results in poor performance on real life data. Therefore, it has to be pruned using the validation data set.

Example to illustrate the application of decision trees for parameter identification:

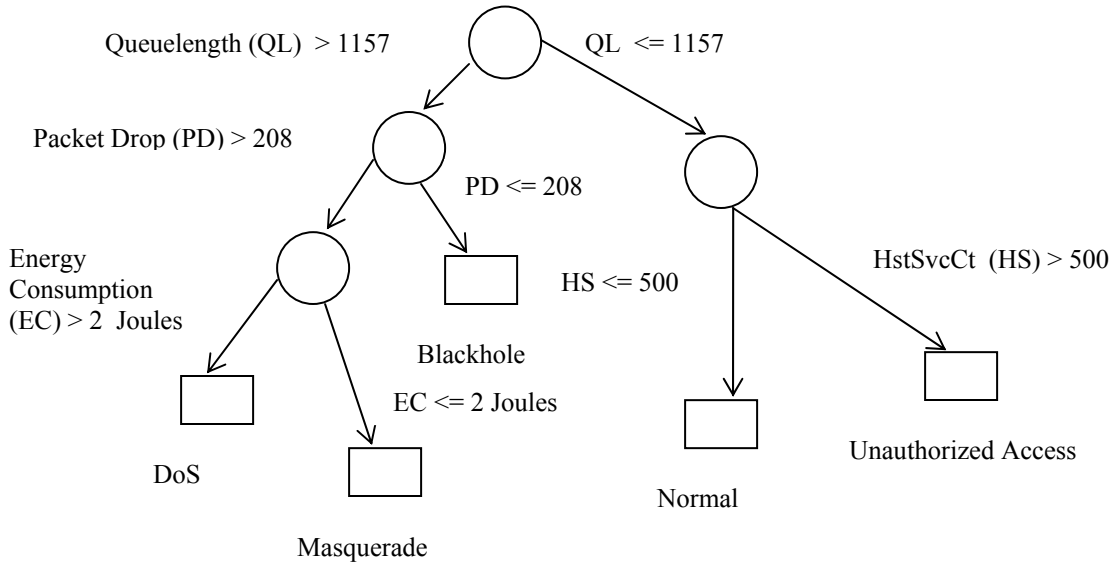


Figure 3.7 Example for Classification Trees

Let us consider the following example to explain the concept of classification trees as used in the thesis. Let us consider a dataset for training that has the records sourced from the network that is subjected to security attack. At the beginning, all of the records in the training set are together in one big box. The algorithm then systematically tries breaking up the records into two parts, examining one variable at a time and splitting the records on the basis of a dividing line in that variable (say, Queuelength > 1157 or Queuelength <= 1157). The objective is to attain as homogeneous set of labels (say, "DoS", “Masquerade”, “black hole” “unauthorized access” or "normal") as possible in each partition. This splitting or partitioning is then applied to each of the new partitions. The process continues until no more useful splits can be found.

The process then starts with a training dataset consisting of pre-classified records. Pre-classified means that the target field, or dependent variable, has a known class or label ("DoS" or "Normal" for example). The goal is to build a tree that distinguishes

among the classes. For simplicity, let us assume that there are only two target classes and that each split is binary partitioning. The splitting criterion easily generalizes to multiple classes, and any multi-way partitioning can be achieved through repeated binary splits. To choose the best splitter at a node, the algorithm considers each input field in turn. In essence, each field is sorted. Then, every possible split is tried and considered, and the best split is the one which produces the largest decrease in the diversity of the classification label within each partition (this is just another way of saying "the increase in homogeneity"). This is repeated for all fields, and the winner is chosen as the best splitter for that node. The process is continued at the next node and, in this manner, a full tree is generated. The most important application of the classification trees in this dissertation is to identify the significant splitters (say, Queuelength, Packet Drop, Energy Consumption) that is used to distinguish classes (say, “DoS”, “Masquerade”, “Black Hole”, “Unauthorized Access”, “Normal”) efficiently.

Step 3 - Identification of thresholds for the significant parameters:

In order to perform the threat detection, intruder identification and response operations, we need to identify the normal, uncertain and vulnerable states for the significant network parameters that have been identified in Step 2. Six-sigma concept is used to calculate the Upper Control Limit (UCL) and Lower Control Limit (LCL) values in order to differentiate the normal, uncertain and vulnerable state of the significant network parameters. These thresholds are used in the fuzzy membership functions applied for calculating the Threat Index (explained in Step 5). In the response framework these threshold values for the significant parameters are used to update the counters and flag the intruder for response actions (explained in Chapter 4). A similar approach of

setting the width of UCL and LCL for calculating thresholds to differentiate normal state from abnormal state of network node is proposed in [105].

Six-sigma is a data driven approach used to measure quality and is a methodology for eliminating defects [83]. Six-sigma is used in the model to achieve extremely low failure rates in intrusion detection. Since six-sigma is six standard deviation, in theory, a six sigma would be approximately two failures per billion attempts. In practice, due to a drift of plus or minus 1.5, six sigma status means less than 3.4 failures per million. This is an extremely low rate of failure. Here failure refers to the defect in the intrusion detection. Please note that the six-sigma model is being used only for obtaining the thresholds of the significant parameters. Also, the six sigma concept is used only from the statistical variability point of view and not from the enterprise point of view. So the issues like management aspect and tracing aspect which are applicable for the enterprise application do not come in to picture in our model It has been demonstrated that six sigma methodologies, integrated with rigorous statistics, can be flexible, powerful and successful without being either overly simplistic or inordinately cumbersome [83].

The equations that are used to calculate upper and lower control limit values to differentiate normal state, uncertain state and vulnerable state for significant attack sensitive parameters are given below. The upper and lower control limits for the significant parameters are computed using the training data sourced from MANET.

Theoretical control limits of UCL and LCL for uncertain state are represented as:

$$UCL_{us} = \mu + \frac{3\sigma}{\sqrt{N}} \tag{3.7}$$

$$LCL_{us} = \mu - \frac{3\sigma}{\sqrt{N}} \tag{3.8}$$

Theoretical control limits of UCL and LCL for vulnerable state are represented as:

$$UCL_{vs} = \mu + \frac{6\sigma}{\sqrt{N}} \tag{3.9}$$

$$LCL_{vs} = \mu - \frac{6\sigma}{\sqrt{N}} \tag{3.10}$$

Here μ represents the mean of the N data items, $(\sigma)^2$ represents the variance which is the average of the square of the distance between each point in a total population (N) and the mean (μ), and σ represents the standard deviation, which is the square root of the variance.

When the lower control levels are negative and if those negative values do not make sense for a particular parameter (For eg. packet drop and queue length can never be practically negative), only UCL_{us} and UCL_{vs} are considered to determine the thresholds for the normal, uncertain and vulnerable state. Here values greater than or equal to UCL_{vs} correspond to the vulnerable state. The values smaller than or equal to UCL_{us} correspond to the normal state, and values between UCL_{vs} and UCL_{us} correspond to the uncertain state.

If the lower control levels are not negative or if the negative values are relevant for a particular parameter, then both LCL_{us} and LCL_{vs} are considered in addition to UCL_{us} and UCL_{vs} for determining the thresholds. Here values greater than UCL_{vs} or smaller than LCL_{vs} correspond to the vulnerable state. The values between LCL_{us} and UCL_{us}

correspond to the normal state, and values between UCL_{vs} and UCL_{us} or between LCL_{vs} and LCL_{us} correspond to the uncertain state. The relationship between LCL and UCL with VS, US and NS is illustrated in the Figure 3.8.

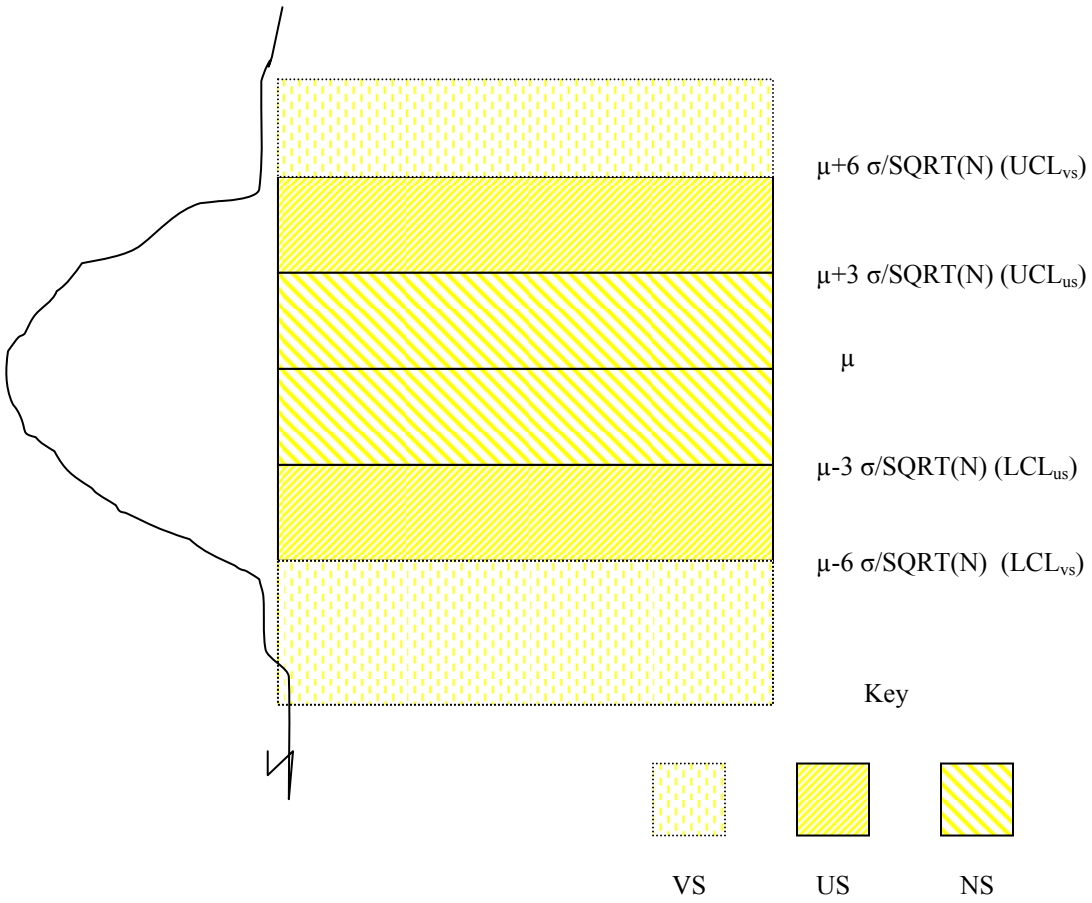


Figure 3.8 Relationship of UCL and LCL with VS, US and NS

Example to illustrate the six sigma methodology for threshold identification:

The example below explains how the six-sigma methodology is used to calculate the threshold values for the significant parameters in this dissertation.

Let us consider the DoS attack for which network parameters, PD (Packet Drop), Queue Length (QL) and Energy Consumption (EC) are identified as significant parameters through the classification trees methodology described in Step 2. The training

data sourced from the MANET is used to compute average μ and standard deviation σ for these attack sensitive network parameters. The LCL and UCL of these parameters for the uncertain and vulnerable state are then calculated using the equations 3.7 to 3.10 and are shown in the Table 3.1.

Table 3.1 UCL and LCL values of significant parameters

Parameter	μ	σ	UCL_{vs}	UCL_{us}	LCL_{us}	LCL_{vs}
Packet Drop (PD)	29.5294	171.4791	208.6336	119.0815	-60.0227	-149.5747
Queue Length (QL)	154.307	960.6954	1157.721	656.0145	-347.3991	-849.1059
Energy Consumption (EC) Joules	0.6852	1.2532	1.9941	1.3397	0.8953	-0.6237

As shown in Table 3.1, LCL values for PD, QL and EC are negative and UCL values are positive. Since the negative values do not make sense for PD, QL and EC, values greater than or equal to UCL_{vs} correspond to the vulnerable state. The values between UCL_{vs} and UCL_{us} correspond to the uncertain state. Values less than or equal to UCL_{us} correspond to the normal state. Hence for PD, values greater than or equal to 208 will represent the vulnerable state. Values of PD between 119 and 208 will represent the uncertain state and values of PD less than or equal to 119 will represent the normal state. For QL, values greater than or equal to 1157 will represent the vulnerable state. Values of QL between 656 and 1157 will represent the uncertain state and values of QL less than or equal to 656 will represent the normal state. For EC, values greater than or equal to 1.9941 joules will represent the vulnerable state. Values of EC between 1.9941 and 1.3397 will represent the uncertain state and values of EC less than or equal to 1.3397 will represent the normal state. These threshold values are used to obtain the fuzzy relationships of the network parameters with the membership functions as explained in Step 5.

Step 4 - Methodology to set the TI thresholds from the Training Data:

We have set the TI metric to be a number between 1 and 10, where 10 represents the highest threat (Most Vulnerable State) and 1 represents the lowest threat (Most Normal State) to the node. For the value of TI to make sense, we need to classify which number indicates which state of the node – normal state, uncertain state, vulnerable state. So we need to identify uncertain state threshold (P_1) and vulnerable state threshold (P_2) for TI from this range of 1 to 10, such that values of TI less than the uncertain state threshold (P_1) indicate a node is in normal state, TI values greater than the vulnerable state threshold (P_2) indicate a node is in vulnerable state and TI values between the uncertain state threshold (P_1) and the vulnerable state threshold (P_2) indicate a node is in the uncertain state.

Once obtained, these TI thresholds are used for the following purposes in the proposed model: First purpose is to assign outputs (Y_j) in the fuzzy rule(explained in Step 5) and the second purpose is to identify the node state as being normal, uncertain or vulnerable by comparing the computed TI with these thresholds.

Since P_1 and P_2 are the integers values between 1 and 10 and P_2 should be greater than P_1, 45 combinations of P_1 and P_2 are possible. The following brute force search algorithm is applied on the training sample space S to search for the pair (P_1, P_2) that gives the correct classification of TI.

Algorithm:

The algorithm for the TI threshold training is illustrated in the Figure 3.9.

1. Let there be N records in the training sample space S, where each record consists of the following elements: {x_1, x_2,…x_m, Outcome Label}. Here x_1, x_2,…x_m represent the values of the significant parameters and 'Outcome Label' represents 'Normal', 'Uncertain' and 'Vulnerable' to indicate the true state.
2. For each of the possible pair (P_1, P_2) where $P_2 > P_1$ and $1<=P_1<=10$ and $1<=P_2<=10$

Figure 3.9 TI Threshold Training Algorithm

Based on the experimentation explained in Chapter 5, the TI threshold training algorithm resulted in the pair (4, 7) having the highest count. Hence, the TI threshold for the uncertain state of the network, P_1, is chosen to be 4, and the TI threshold for vulnerable state of the network, P_2, is chosen to be 7.

Step 5 - TI Evaluation using Fuzzy logic based methodology:

TI is evaluated using fuzzy logic and continuously measured values of the significant network parameters. Fuzzy logic is one of the heuristic approaches for threat

evaluation. One of the active areas of fuzzy logic applications is fuzzy expert control system. Fuzzy expert control systems are systems that use common sense rules and natural language statement instead of hard-boundary. Fuzzy control systems have several important characteristics that suit intrusion detection well. They are summarized as follows:

- Fuzzy systems can readily combine inputs from widely varying sources.
- It is easy, flexible, fast and accurate in design and implementation.
- Fuzzy logic is a very natural approach to represent human thinking [85].
- It is suitable for both linear and nonlinear systems.
- It allows for imprecise mathematical models and measuring sensors
- It is more robust, accurate and flexible than classical controllers [102]. Fuzzy logic may radically affect computer security as a relatively new paradigm.
- It can be used in trusted system analysis and design, in measuring the security of the systems, and in representing the imprecise human world of policies and inference [103].

Fuzzy intrusion detection system is an effective and flexible system compared to the existing distributed intrusion detection systems. Fuzzy logic enables efficiency, scalability, effectiveness, robustness, and adaptability to changing methods of attacks [103]. In addition, fuzzy logic is an excellent complement to other data analysis techniques. For example, outputs (predicted values) of data mining can be used as inputs to improve the predictive accuracy of fuzzy logic. Fuzzy logic can be integrated with the intrusion detection system very well [104]. Using the fuzzy logic as the detection mechanism, produces an effective detection mechanism that minimizes false alarm rates,

and helps to tolerate the legitimate variations in behavior that occur when the system users perform tasks that they do not ordinarily do.

A good anomaly-based intrusion detection system should be able to perform correlation analysis of multiple metrics (input network parameters), anomalous behavior, and the threat value evaluation. Such a correlation can be addressed by a set of fuzzy variables and rules, which if properly constructed, can help the system formulate an accurate threat evaluation, while allowing flexible detection for legitimate variation in behaviors. The proposed fuzzy representation enables the proposed protocol to do this correlation accurately and effectively. The fuzzy concepts used to evaluate Threat Index are described as below.

Mamdani Fuzzy Rule Based Model:

Mamdani fuzzy-rule based systems constitute a linguistic description in both the antecedent parts and the consequent parts. Each rule is a description of a condition-action statement that may be clearly interpreted by the users. To describe a mapping from input $U_1 \times U_2 \times \ldots \times U_n$ (where $\times$ is the Cartesian product) to output W, the linguistic rule structure of Mamdani models is as follows: R_i: IF x_1 is A_{i1} and … and x_n is A_{in} THEN y is C_i, $i = 1, \ldots, L$. Where, L is the number of fuzzy rules, $x_j \in U_j$, $j = 1, 2, \ldots, n$, are the input variables, y is the output variable, and A_{ij} and C_i are linguistic variables or fuzzy sets for x_j and y respectively. A_{ij} and C_i are characterized by membership functions $\mu_{A_{ij}}(x_j)$ and $\mu_{C_i}(y)$, respectively. Inputs are of the form: x_1 is A'_1, x_2 is $A'_2,\ldots$, x_r is A'_n. where A'_1, $A'_2,\ldots$, A'_n are fuzzy subsets of U_1, $U_2,\ldots$ U_n, which are the universe of disclose (or the domain of interest) of inputs. The snapshot of sample Mamdani based fuzzy model simulated in MATLAB is attached in Appendix F.

Fuzzification:

The first step in the design of a fuzzy logic evaluation system is the *fuzzification* of the input and output linguistic variables.

Fuzzy Rule Base:

The second step in the design of a fuzzy logic evaluation system is to convert the designer's knowledge of the process into a set of if-then rules that relate input to output. Generically, it can be stated in the following form: If X is *x* then Y is *y Where* X and Y are linguistic variables and *x* and *y* are their linguistic values. The rule base defines a set of imprecise dependencies between the two linguistic variables. This is the module that determines the rule. The rules in the rule base are verified for the following mathematical properties:

- Completeness
- Consistency
- Continuity and
- Interaction.

Defuzzification:

The third and last step in the design of a fuzzy logic control system is to defuzzify the results of the rule base to produce a crisp control action. This provides the mathematical basis to proactively respond to failures and to control at much higher speeds. Consider the example of a simple fuzzy system with two fuzzy rules as shown in the following Figure 3.10.

Input x_1 =10 Input x_2=1

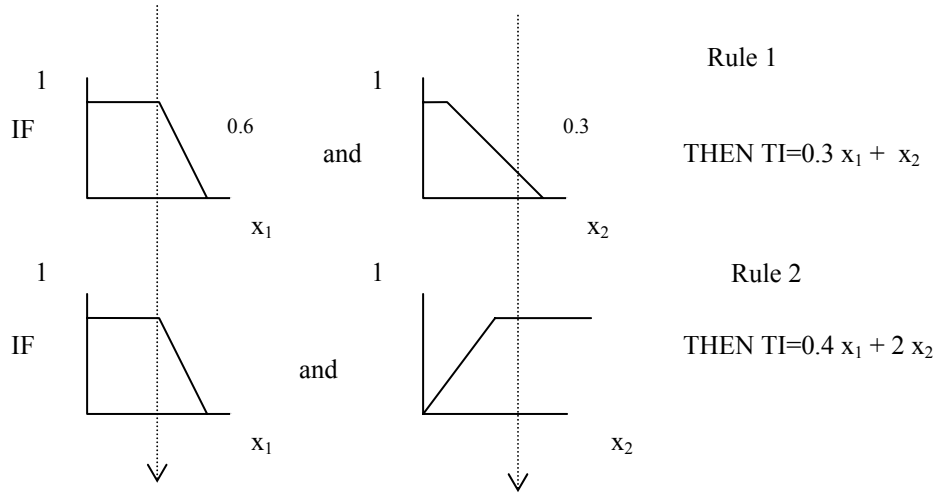


Figure 3.10 Example for Defuzzification

Here, the first rule is "IF x_1 is small and x_2 is small, then TI=0.3 x_1+ x_2 " and second rule is "IF x_1 is large and x_2 is large, then TI = 0.4 x_1 + 2 x_2" where x_1 and x_2 are the input metrics and TI is the output Threat Index. For the given input vector (x_1, x_2) = (10, 1), membership values are calculated for each rule. The first rule has membership values, 0.6 and 0.3, for the given input values. For the min operator, as per Mamdani model, the rule strength for the first rule is min (0.6, 0.3) = 0.3. Similarly the outputs and rule strengths are calculated for each rule. The output for each rule for the input metric vector (10, 1) is 4, and 6 respectively and the rule strength for the second rule is min (1.0, 0.6) = 0.6. Therefore, the final system output is

$TI = y^* = \sum w_i y_i / \sum w_i = (0.3 * 4 + 0.6 * 6) / (0.3 + 0.6) = 5.33$ (approx)

Design of Antecedent Parts and Design of Fuzzy Inference System:

Once the vulnerability metrics and their normal, uncertain and vulnerable state values are designed, the membership function and the fuzzy rules can be designed. A sample set of fuzzy rules is shown below.

If Vulnerability metric is *low* then TI state is *normal* (rule 1)

If Vulnerability metric is *medium* then TI state is *uncertain* (rule 2)

If Vulnerability metric is *high* then TI state is *vulnerable* (rule 3)

Here the vulnerability metrics are antecedent parts and antecedent parts need to be designed to partition an input space as shown in the following Figure 3.11.

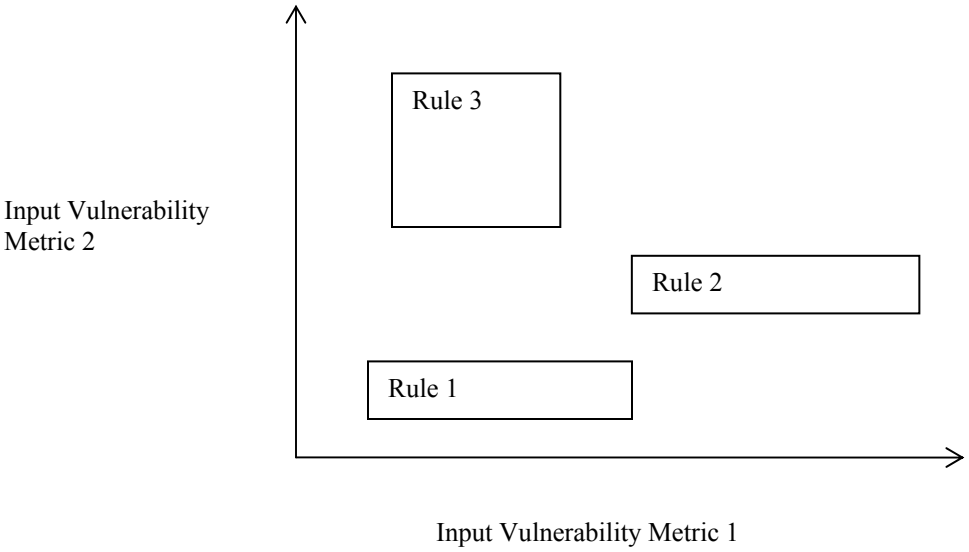


Figure 3.11 Fuzzy Rules and Vulnerability Metrics

Fuzzy systems allow overlapping rule areas to shift from one control rule to another. The degree of this overlapping is defined by membership functions. The overlapping can be optimized by proper fuzzy training. Such optimization takes care of the multivariate input analysis effectively.

Optimization of the Membership Functions Using Fuzzy Training:

For a fuzzy variable, x with universe of discourse [0, U_x] and three fuzzy sets *NS*, *US*, and *VS*, the membership functions are as shown in Figure 3.12. In this figure, ns_x and vs_x are initialized with the threshold values computed in Step 3. us_x is initialized to be the average of ns_x and vs_x. The membership function optimization problem is the determination of the optimized ns_x, us_x and vs_x points for the fuzzy variable, x.

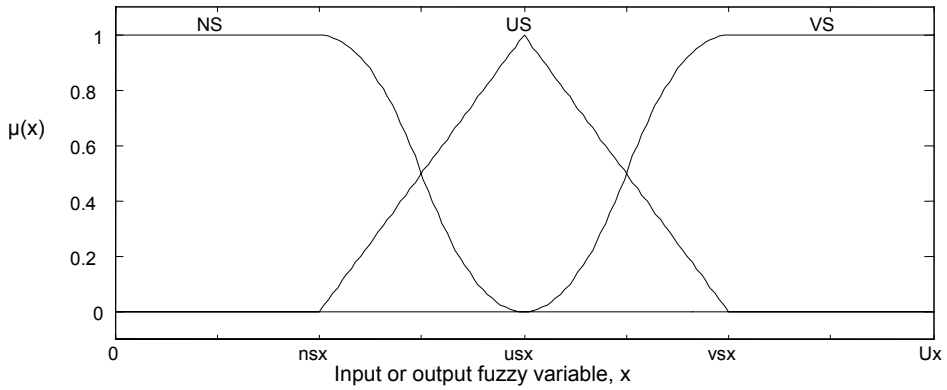


Figure 3.12 Membership Functions for a Fuzzy Variable

The membership function optimization methodology works by increasing or decreasing the membership value of each fuzzy set based on the multivariate input training data. The concept here is to increase/decrease the membership values by modifying the slope. This is achieved by multiplying vs_x with decrease_ratio in order to shift vs_x value to the left; ns_x is multiplied with increase_ratio in order to shift ns_x to the right, and for us_x, it is shifted either left or right depending on the input values.

Figure 3.13 shows the membership functions for fuzzy variable, x after vs_x is shifted to left by multiplying vs_x with decrease_ratio

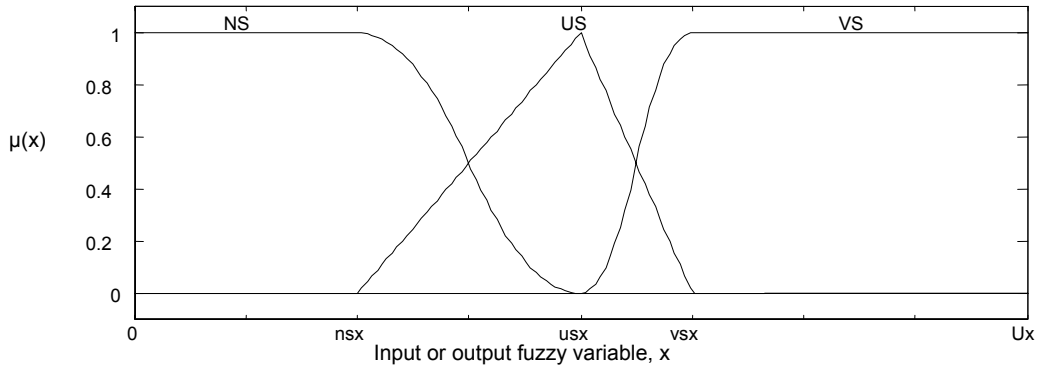


Figure 3.13 Membership Functions for a Fuzzy Variable by Shifting vs_x to Left

Figure 3.14 shows the membership functions for fuzzy variable, x after ns_x is shifted to right by multiplying ns_x with increase_ratio.

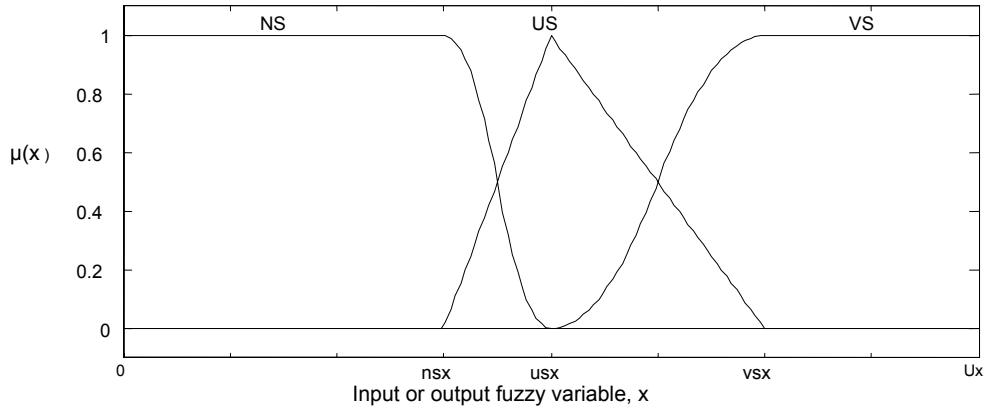


Figure 3.14 Membership Functions for a Fuzzy Variable by Shifting ns_x to Right

Due to the triangle rule, value of us_x must always be greater than ns_x and lower than vs_x. The triangle rule is given by: $ns_x < us_x < vs_x$. This rule is taken care by checking the result of multiplication and making sure the above rule is preserved before assigning the new values to ns_x, us_x and vs_x.

In our membership function optimization methodology, the increase_ratio and decrease_ratio were selected to be 1.03 and 0.97 respectively. This is due to the reason that the selected training data is not continuous, and so we cannot choose values less than 3% and keep increasing or decreasing it. Also, the decrease/increase ratio cannot be in a very large range say (say 60% or 40%). This is because the range between ns_x and us_x is set to 50% and the range between us_x and vs_x is set to 50% as us_x is initialized to be the average of ns_x and vs_x. The use of 3% for increment/decrement ratio also means that the membership function optimization methodology needs around 15 steps to make $ns_x = us_x$ (the maximum value for ns_x using triangle rule), and around 15 steps to make $vs_x = us_x$ (the minimum value for vs_x using triangle rule). To summarize, in recommending values

for increment/decrement ratio, it is necessary to ensure that a single variation of the triangle points (ns_x, us_x, and vs_x) does not result in large variation in the membership functions. In our case it takes a maximum of 15 intervals to increase from ns_x to us_x or to decrease from vs_x to us_x.

Also, in our fuzzy membership function optimization methodology, it is not necessary to optimize all input variables at once. For example, it is not necessary that if one wants to optimize EC membership functions, one should optimize PD membership functions also at the same time. Only if the value of the input variable is 20% lesser than ns_x or 20% greater than vs_x, the optimization of its membership functions is considered. This is due to the reason that as per our testing performed by varying percentages to the variables in the training data, values less than 10% generated too narrow values for membership function optimization and values greater than 30% generated too far values for membership function optimization. MATLAB code which implements our fuzzy training methodology to generate optimized FIS from the training data are attached in the Appendix C. Thus, normalization of TI (security level) from the arbitrary fuzzy sets is performed with the help of multivariate fuzzy membership function optimization algorithm. This algorithm identifies the optimized membership function points of all the input significant parameters by training them together. These trained and optimized membership function points normalize the input parameters though they are in multi dimensional space and hence generate the normalized output TI.

Design of consequent parts:

They can be designed to express the output as constant values, say TI = 1 to 3 for normal state, TI = 4 to 6 for uncertain state and TI = 7 to 10 for vulnerable state.

Formal Analysis:

$$\text{Let} \quad X = \{x_1, x_2 \ldots x_n\} \tag{3.11}$$

represent a sample space of the attack sensitive network parameters. For a sample space of objects defined as $X = \{x_i\}$, the fuzzy set A in X is a set of ordered pairs defined as:

$$A = \{(x_i, \mu_j(x_i)), x \epsilon X\} \tag{3.12}$$

The above set of ordered pairs in the area of fuzzy logic can be formally represented as:

$$A = \{x_1/\mu_j(x_1) + x_2/\mu_j(x_2) + x_3/\mu_j(x_3) + \ldots + x_n/\mu_j(x_n)\} \tag{3.13}$$

In the above equation 3.13, '+' and '/' are purely syntactic symbols and do not denote arithmetic operations. In the above expressions, μ_j represents the grade of membership of x_i. Each attack sensitive network parameter has a grade of membership corresponding to its normal, uncertain and vulnerable threshold ranges. The grade of membership indicates the level of participation of the metric in a particular threshold range. The grades of membership are defined as μ_{NS} for the normal threshold range, μ_{US} for the uncertain threshold range and μ_{VS} for the vulnerable threshold range. The values of membership function $\mu_j (x_i)$ are real numbers in the interval [0, 1]. The membership function values can be computed using many different schemes. In this model, we have used trapezoidal and triangular membership function for fuzzy variables.

Fuzzy system in our model uses linguistic variables to describe input and output to perform a fuzzy operation on the inputs for generating the output. Since this model is based on a Mamdani type of fuzzy controller [103], it uses composition based inference mechanism, which combines all rules into an aggregated system output and determines the final non-fuzzy control value. This model uses a Centroid method with *min* operator

for defuzzification, where the final system output, Threat Index (TI), using the fuzzy logic is expressed as [103]: $TI = \frac{\sum_{j=1}^{m} w_j y_j}{\sum_{j=1}^{m} w_j}$ (3.14)

Here, y_j indicates the output value associated with the particular rule j in the fuzzy set; The output y_j takes its values from the threshold values of TI for normal, uncertain and vulnerable states. Here "m" indicates the number of rules and w_j indicates the rule strength for rule j in the fuzzy set.

Rule strength illustrates how active or reliable a rule is in the fuzzy set. Rule strength is calculated as:

$$w_j = \min(\mu_j(x_i)) \quad (3.15)$$

where i Є {1, 2,…n}, and n is number of network parameters for each rule. If there are k membership values possible for the network parameters and if n is the number of network parameters, then $m = k^n$ rules are possible and $1 <= j <= m$.

Since all the significant parameters that are selected have equal importance (as explained in Step 2), they are equally weighted in the TI calculation.

Example to illustrate fuzzy logic methodology for TI evaluation to detect threat:

The network parameters, Packet Drop (PD), Queue Length (QL) and Energy Consumption (EC) are identified as the significant parameters for a DoS attack (explained in Step 2) The threshold ranges for the normal state (NS), uncertain state(US) and vulnerable state(VS) of these network parameters were identified in Step3 and are listed again below for reference.

For the Packet Drop(PD) parameter, PD<=119 represents normal state, 119<PD<208 represents uncertain state and PD>=208 represents vulnerable state. For the

Queue Length (QL) parameter, QL<=656 represents normal state, 656<QL<1157 represents uncertain state and QL>=1157 represents vulnerable state. For the Energy Consumption (EC) parameter, EC<=1.33 joules represents normal state, 1.33<EC<1.99 joules represents uncertain state and EC>=1.99 joules represent vulnerable state.

The next step is to formulate the fuzzy relation of these three attack sensitive network parameters with the grade of membership. Figure 3.15 below represents the fuzzy relation of PD with the membership functions.

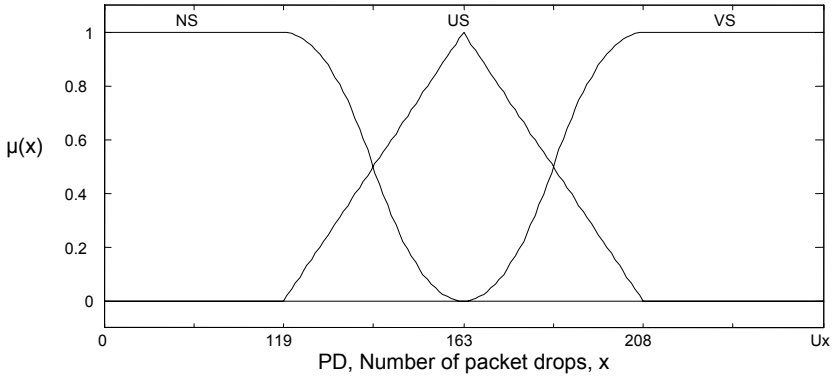


Figure 3.15 Fuzzy Model for Packet Drop Metric

The Figure 3.15 shows two trapezoidal and one triangular shaped fuzzy set. We have used the triangular membership functions shown in Figures 3.12 through 3.17 due to the reason that the parametric and functional descriptions of these membership functions are efficient. In these membership functions, the designer needs only to define three parameters; nsx, vsx and usx. Here nsx, usx, and vsx are the normal state, uncertain state and vulnerable state threshold values. It has been proven that triangular MFs can approximate any other membership function [127]. This function is specified by three parameters (a, b, c) as follows:

$$triangle(x_{in};a,b,c) = \begin{cases} (x_{in}-a)/(b-a) & for\ a \le x_{in} \le b \\ (c-x_{in})/(c-b) & for\ b \le x_{in} \le c \\ 0 & elsewhere \end{cases} \quad (3.16)$$

where a = nsx, b = vsx, c = vsx and xin is the input to the fuzzy system of type fuzzy variable x.

The first trapezoid represents the fuzzy set μ_{NS}(PD) = {1.0/>=0;<=119, 0.0/163}. This fuzzy set indicates that membership function μ_{NS} of PD is 1 for the value of PD from 0 to 119, and the membership function μ_{NS} of PD is 0 when the value of PD is 163. The second fuzzy set is a triangle and represents the fuzzy set μ_{US} (PD) = {0.0/119, 1.0/163, 0.0/208}. The third fuzzy set is a trapezoid and represents that μ_{VS}(PD) = {0.0/163, 1.0/>=208}.

The fuzzy relation of QL with the membership functions is represented in Figure 3.16.

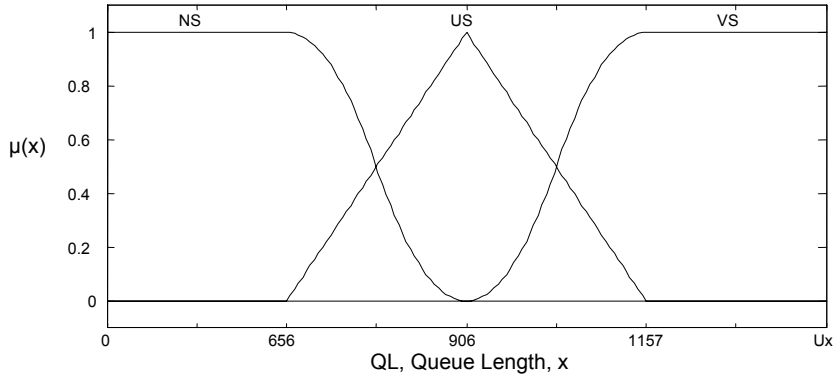


Figure 3.16 Fuzzy Model for Queue Length Metric

The Figure 3.16 shows two trapezoidal and one triangular shaped fuzzy set. The first trapezoid represents the fuzzy set μ_{NS}(QL) = {1.0/>=0;<=656, 0.0/906}. This fuzzy set indicates that membership function μ_{NS} of QL is 1 for the value of QL from 0 to 656, and the membership function μ_{NS} of QL is 0 when the value of QL is 906. The second fuzzy set is a triangle and represents the fuzzy set μ_{US} (QL) = {0.0/656, 1.0/906,

0.0/1157}. The third fuzzy set is a trapezoid and represents that μ_{VS}(QL) = {0.0/906, 1.0/>=1157}.

The fuzzy relation of EC with the membership functions is represented in Figure 3.17.

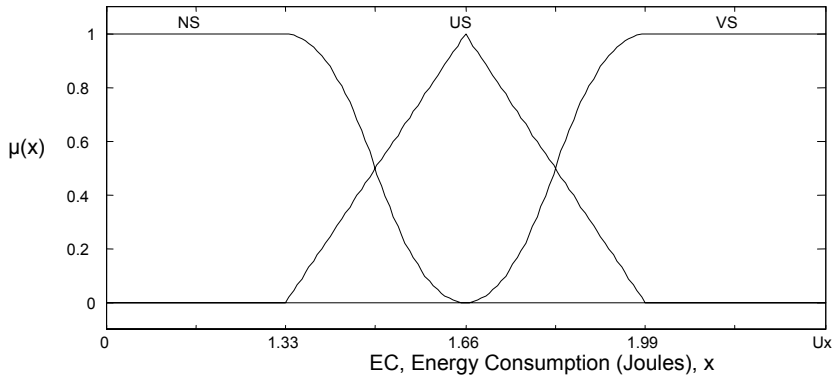


Figure 3.17 Fuzzy Model for Energy Consumption Metric

The Figure 3.17 shows two trapezoidal and one triangular shaped fuzzy set. The first trapezoid represents the fuzzy set μ_{NS}(EC) = {1.0/>=0;<=1.33, 0.0/1.66}. This fuzzy set indicates that membership function μ_{NS} of EC is 1 for the value of EC from 0 to 1.33 joules, and the membership function μ_{NS} of EC is 0 when the value of EC is 1.66 joules. The second fuzzy set is a triangle and represents the fuzzy set μ_{US} (EC) = {0.0/1.33, 1.0/1.66, 0.0/1.99}. The third fuzzy set is a trapezoid and represents that μ_{VS}(EC) = {0.0/1.66, 1.0/>=1.99}.

Next we assign values to the output parameter y_j. As explained in Step4, the uncertain state and vulnerable state thresholds for TI are 4 and 7 respectively. These thresholds are used to assign the values to the output y_j. The implication relation "VS PD" → "VS TI", "VS QL" → "VS TI", "VS EC" → "VS TI", "NS PD → "NS TI" and so on is used in arriving at the output for each rule.

The next step is to formulate all the combination of rules possible for this system. Since the system has 3 network parameters and 3 grades of membership for each parameter, i.e. n=3 and k=3 as per the notations used in the description, m =k^n= 3^3 = 27; So 27 combinations of rules are possible. The rules and their associated output are as below:

Rule 1: If PD is *NS* and QL is *NS* and EC is *NS*; threat index of the rule y_1 is *NS*
Rule 2: If PD is *NS* and QL is *NS* and EC is *US*; threat index of the rule y_2 is *NS*
Rule 3: If PD is *NS* and QL is *NS* and EC is *VS*; threat index of the rule y_3 is *NS*
Rule 4: If PD is *NS* and QL is *US* and EC is *NS*; threat index of the rule y_4 is *NS*
Rule 5: If PD is *NS* and QL is *US* and EC is *US*; threat index of the rule y_5 is *US*
Rule 6: If PD is *NS* and QL is *US* and EC is *VS*; threat index of the rule y_6 is *US*
Rule 7: If PD is *NS* and QL is *VS* and EC is *NS*; threat index of the rule y_7 is *NS*
Rule 8: If PD is *NS* and QL is *VS* and EC is *US*; threat index of the rule y_8 is *US*
Rule 9: If PD is *NS* and QL is *VS* and EC is *VS*; threat index of the rule y_9 is *VS*
Rule 10: If PD is *US* and QL is *NS* and EC is *NS*; threat index of the rule y_{10} is *NS*
Rule 11: If PD is *US* and QL is *NS* and EC is *US*; threat index of the rule y_{11} is *US*
Rule 12: If PD is *US* and QL is *NS* and EC is *VS*; threat index of the rule y_{12} is *US*
Rule 13: If PD is *US* and QL is *US* and EC is *NS*; threat index of the rule y_{13} is *US*
Rule 14: If PD is *US* and QL is *US* and EC is *US*; threat index of the rule y_{14} is *US*
Rule 15: If PD is *US* and QL is *US* and EC is *VS*; threat index of the rule y_{15} is *US*
Rule 16: If PD is *US* and QL is *VS* and EC is *NS*; threat index of the rule y_{16} is *US*
Rule 17: If PD is *US* and QL is *VS* and EC is *US*; threat index of the rule y_{17} is *US*
Rule 18: If PD is *US* and QL is *VS* and EC is *VS*; threat index of the rule y_{18} is *VS*
Rule 19: If PD is *VS* and QL is *NS* and EC is *NS*; threat index of the rule y_{19} is *NS*
Rule 20: If PD is *VS* and QL is *NS* and EC is *US*; threat index of the rule y_{20} is *US*
Rule 21: If PD is *VS* and QL is *NS* and EC is *VS*; threat index of the rule y_{21} is *VS*
Rule 22: If PD is *VS* and QL is *US* and EC is *NS*; threat index of the rule y_{22} is *US*
Rule 23: If PD is *VS* and QL is *US* and EC is *US*; threat index of the rule y_{23} is *US*
Rule 24: If PD is *VS* and QL is *US* and EC is *VS*; threat index of the rule y_{24} is *VS*
Rule 25: If PD is *VS* and QL is *VS* and EC is *NS*; threat index of the rule y_{25} is *VS*
Rule 26: If PD is *VS* and QL is *VS* and EC is *US*; threat index of the rule y_{26} is *VS*
Rule 27: If PD is *VS* and QL is *VS* and EC is *VS*; threat index of the rule y_{27} is *VS*

Now, let us consider that at some point in time, network parameter PD takes a value of 174, parameter QL takes a value of 843 and the parameter EC takes the value of 1.8 Joules. With the framework developed above, we can calculate the TI and determine

the node vulnerability for this scenario by applying equation 3.14. The rule strength for this scenario can be calculated using the membership functions shown in Figure 3.15, 3.16 and 3.17. Table 3.2 shows the values obtained using these fuzzy relations.

Table 3.2 Calculation of rule strength

Rule Number (j)	$\mu_{j\,(PD)}$	$\mu_{j\,(QL)}$	$\mu_{j(EC)}$	Rule Strength, w_j , $\min(\mu_{i(PD)}\mu_{j(QL)}\ \mu_{i(EC)})$	Output, y_j	$w_j y_j$
1	0	0.25	0	0	1	0
2	0	0.25	0.4	0	1	0
3	0	0.25	0.6	0	1	0
4	0	0.75	0	0	1	0
5	0	0.75	0.4	0	4	0
6	0	0.75	0.6	0	4	0
7	0	0	0	0	1	0
8	0	0	0.4	0	4	0
9	0	0	0.6	0	7	0
10	0.75	0.25	0	0	1	0
11	0.75	0.25	0.4	0.25	4	1
12	0.75	0.25	0.6	0.25	4	1
13	0.75	0.75	0	0	4	0
14	0.75	0.75	0.4	0.4	4	1.6
15	0.75	0.75	0.6	0.6	4	2.4
16	0.75	0	0	0	4	0
17	0.75	0	0.4	0	4	0
18	0.75	0	0.6	0	7	0
19	0.25	0.25	0	0	1	0
20	0.25	0.25	0.4	0.25	4	1
21	0.25	0.25	0.6	0.25	7	1.75
22	0.25	0.75	0	0	4	0
23	0.25	0.75	0.4	0.25	4	1
24	0.25	0.75	0.6	0.25	7	1.75
25	0.25	0	0	0	7	0
26	0.25	0	0.4	0	7	0
27	0.25	0	0.6	0	7	0

Here, $\sum_{j=1}^{m} w_j y_j$ = 11.5 and $\sum_{j=1}^{m} w_j$ = 2.5 and hence, from equation 3.14, TI = 4.6. Comparing the TI calculated with the TI thresholds (explained in Step 4), this value of 4.6 for the TI indicates that the node is in uncertain state (US) and is not under attack. Thus, the Threat Index calculated by applying fuzzy logic indicates the state of a node and can be used to detect intrusions/threat.

3.5 Intrusion Detection Algorithm for the Model

The detection framework mechanism explained in Section 3.4 can be summarized as an algorithm shown in the Figure 3.18.

1. Identify the significant parameters, X= $\{x_{i;}\ 1<i<n\}$, using the CART data mining technique.
2. Calculate the threshold values for the significant parameters, X identified in step 1 using six-sigma methodology.
3. Set the threshold ranges for the Threat Index, TI using TI threshold training algorithm
4. From the MANET, continuously measure the values of these parameters identified in step1, on all the links where the node whose TI is to be evaluated is the destination node. For each parameter, compute their average.
5. For each node in the MANET, calculate Threat Index (TI) using the average parameter values obtained in 4 above.
6. Compare the calculated TI with the TI threshold to detect if the network is under attack.
7. For each node that is under threat (abnormal TI) invoke intruder identification and response algorithm (explained in Chapter 4).

Figure 3.18 Intrusion Detection Algorithm Used in IDRMAN

3.6 Example to Illustrate the Detection Framework Mechanism

This section illustrates the detection framework methodology with an example. Let us consider the MANET shown in Figure 3.19. Let the node N_1 be the destination node and $M_{1,j}$ be the neighboring source nodes to N_1.

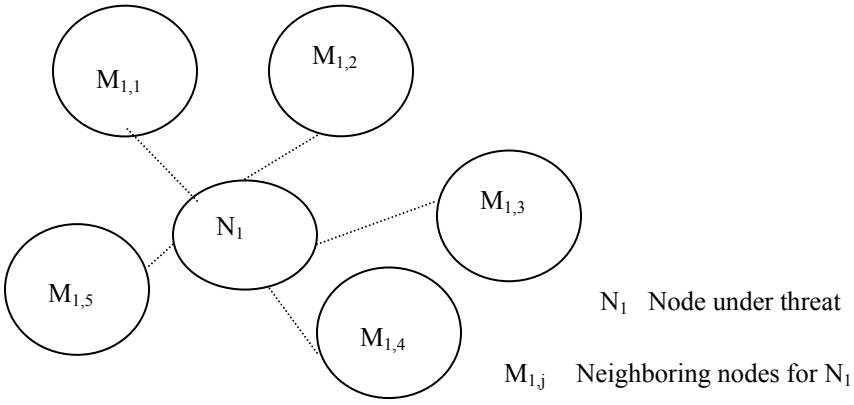


Figure 3.19 MANET Node Under Threat with Neighboring Nodes

Let us assume that node $M_{1,2}$ is malicious and creates a DoS attack at 200 ms on N_1 by bombarding packets and generating traffic on the link between $M_{1,2}$ and N_1. Using CART (Step 2 of Section 3.4), the significant parameters for DoS attack were found to be PD, QL and EC. At 200ms, let the values of the significant parameters for DoS attack on all links to N_1 (where $N_1$is the destination node) be as shown in the Table 3.3.

Table 3.3 Values of significant parameters during attack

Values of parameters at 200 ms	**M_{11} to N_1**	**M_{12} to N_1**	**M_{13} to N_1**	**M_{14} to N_1**	**M_{15} to N_1**	**Average**
PD	155	2000	20	20	20	443
QL	120	12000	120	120	120	2496
EC (Joules)	1.300045	3.920767	2.327486	2.363686	2.611764	2.5047496

From the above table, it is seen that at 200ms, the average values of the significant parameters for Node N_1 are, 443 for PD, 2496 for QL and 2.5 for EC.

These average values are then fed to the fuzzy logic based TI framework to compute TI. Using Table 3.1 for the thresholds of the significant parameters (explained in Step 3 of Section 3.4) , and the TI thresholds obtained in Step 4 of Section 3.4, the TI evaluated by the fuzzy logic based detection framework (explained in Step 5 of Section 3.4) turns out to be 8.14. This is greater than the vulnerable TI threshold of 7 and hence indicates that the node N_1 is under attack.

3.7 Mathematical Analysis to demonstrate that TI is a good metric

In this section, we mathematically analyze the accuracy of the detection metric. We use probability techniques to derive the mean squared error that results when the Threat Index metric is used in estimating a node's vulnerability.

Since identification of the vulnerable state is the most important aspect to recognize an attack, the vulnerable state threshold P_2 plays a vital role in determining the accuracy of the detection technique. So we only consider the probability distributions with respect to P_2 for our analysis.

The Figure 3.20 shows the probability distributions of TI for normal and vulnerable truth values. The probability distribution is obtained by comparing the Truth state of the node with the threat state indicated by the TI computed using fuzzy logic. Table 3.4 and Table 3.5 show the experimentally obtained values for $P_N(TI)$ and $P_V(TI)$ respectively from which the probability distribution of Figure 3.20 is obtained.

Table 3.4 $P_N(TI)$ values obtained from the experimental results

TI	$P_N(TI)$
1	0.00
2	0.00
3	0.77
4	0.23
5	0.00
6	0.00
7	0.00
8	0.00
9	0.00
10	0.00

Table 3.5 $P_V(TI)$ values obtained from the experimental results

TI	$P_V(TI)$
1	0.00
2	0.00
3	0.00
4	0.05
4.4	0.05
6.4	0.10
7	0.00
8	0.80
9	0.00
10	0.00

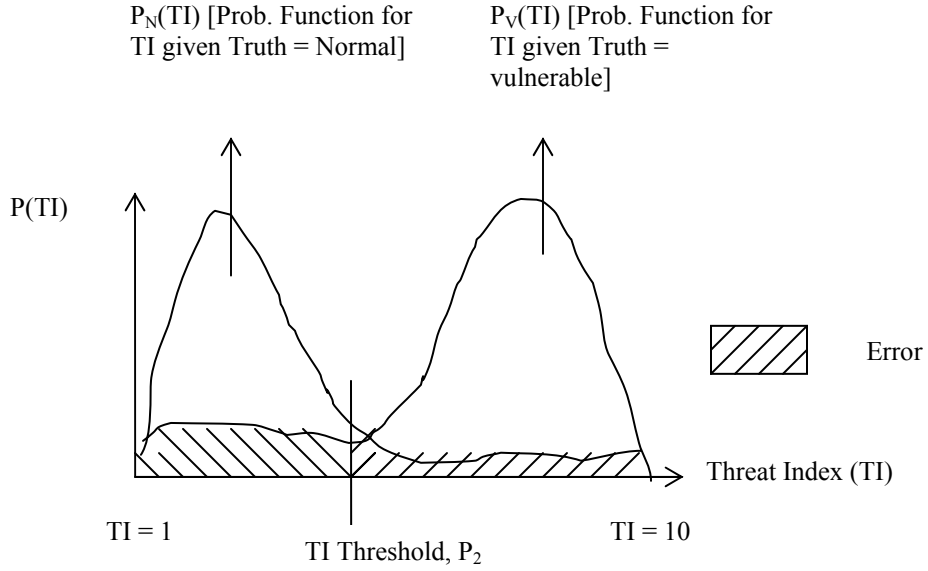


Figure 3.20 Probability distribution of TI for normal and vulnerable truth values

The curve $P_V(TI)$ represents the probability distribution of TI for vulnerable nodes and the curve $P_N(TI)$ represents the probability distribution of TI for normal nodes. As shown in Figure 3.20, for the probability distribution $P_N(TI)$, the shaded area after the TI threshold (P_2) is the error region. This is because, though in reality it belongs to the normal area, it will be perceived as vulnerable due to the TI threshold P_2. Similarly, for the probability distribution $P_V(TI)$ the shaded area before the TI threshold P_2 will be the error region.

The shaded region in the Figure 3.20 represents the total error, Z, due to both the distributions. By the law of total probability [114, 115],

$$P(Z) = P(Z/N).P(N) + P(Z/V).P(V) \text{ where } N \cup V = \Omega \text{ and } N \cap V = \Theta \qquad (3.16)$$

Here the symbols, $\cup$ and $\cap$ represent the union and intersection set operation. Symbols, Ω and Θ indicate the universal set and the null set respectively. P(N) and P(V) represent the probability of occurrence of normal node and vulnerable node in the MANET. P(Z/V) represents the probability of error given that the node is vulnerable and P(Z/N) represents the probability of error given that the node is normal.

Proposition: Threat Index (TI) computed using fuzzy logic based on significant network parameters and threshold values captures the node vulnerability, such that the mean squared error given by $E[(TI-Truth)^2]$ is approximately equal to zero for all values of Truth, where Truth is the unknown reality to be estimated using TI.

Proof: Threat Index estimates the truth, which could be that a node is under attack, or that a node is in the normal state without an attack. As explained in the previous sections, P_2 represents the vulnerable state threshold value of the Threat Index. If a node's TI is >= P_2 and <=10, then the node is categorized to be under attack, i.e. in the vulnerable

state, and if TI < P_2, the node is not under attack. Since TI is discrete, by applying equation 3.16, Mean Square Error (MSE) is given by

$$\text{MSE} = \left[\sum_{i=1}^{i<=P_2} (P_2 - i)^2 P_V(i) \right].P(V) + \left[\sum_{i>=P_2}^{i=10} (i - P_2)^2 P_N(i) \right].P(N) \qquad (3.17)$$

In equation 3.17, $P_N(i)$ and $P_V(i)$ represent the experimentally obtained values for the probability of detection of normal node and vulnerable node respectively when the MANET is simulated normal without any attacks and when the MANET is simulated to be under attack by malicious nodes. Here 'i' represents the TI and takes values from 1 to P_2 for the error in the distribution of vulnerable nodes and values from P_2 to 10 for the error in the distribution of normal nodes. Since in reality only a small percentage of the nodes in the MANET are malicious, the probabilities P(V) and P(N) are assumed to be 0.1 and 0.9 respectively [106-111]. The threshold P_2, as explained in Section 3.4, takes a value of 7. Substituting the values of P_2, P(V), P(N), $P_V(i)$ and $P_N(i)$ in equation 4, the value of the MSE obtained is 0.075, which is approximately equal to 0. Thus the Threat Index computed using fuzzy logic has the mean squared error approximately equal to 0 for all values of Truth.□

CHAPTER IV

IDRMAN SECURITY MODEL – RESPONSE FRAMEWORK

4.1 Overview of the Chapter

This chapter explains the response and protection framework in the IDRMAN model. The response and protection mechanism gets triggered when an attack is detected by the detection framework (explained in Chapter 3). The significant parameters identified by the CART methodology and the thresholds for these significant parameters identified by the six sigma technique in the detection framework are used by the response and protection framework as well. When the detection framework detects an attack, each neighboring source node to the node under threat is examined by the response framework, and different action plans are initiated based on the identification of the nature of the neighboring nodes.

This chapter is organized as follows: Section 4.2 describes the general architecture of the response framework. Sections 4.3 through 4.5 explain the IDRMAN response mechanism. Section 4.6 provides mathematical analysis of the effectiveness of the response mechanism. Section 4.7 highlights the applications of IDRMAN.

4.2 Architecture of IDRMAN Response Framework

Figure 4.1 shows the architecture of the response framework. The main modules of the response model are:

1. Significant parameters monitoring module

2. Reputation management mechanism module

3. Response action execution module

Monitoring significant parameters is the starting point of the response framework architecture. “Monitor Significant Parameter” block in the response framework monitors significant parameters in each network node in a distributed and cooperative manner and feeds the collected observation to the “Reputation Management Mechanism” block. “Reputation Management Mechanism” block then updates the node’s reputation rating counter. The term ‘Reputation Management’ in the architecture refers to assigning counters and flag to the nodes based on their behavior, which is achieved by monitoring significant parameters in a node level at a given instance. The reputation counter is updated by comparing the values of the significant parameters against an expected norm.

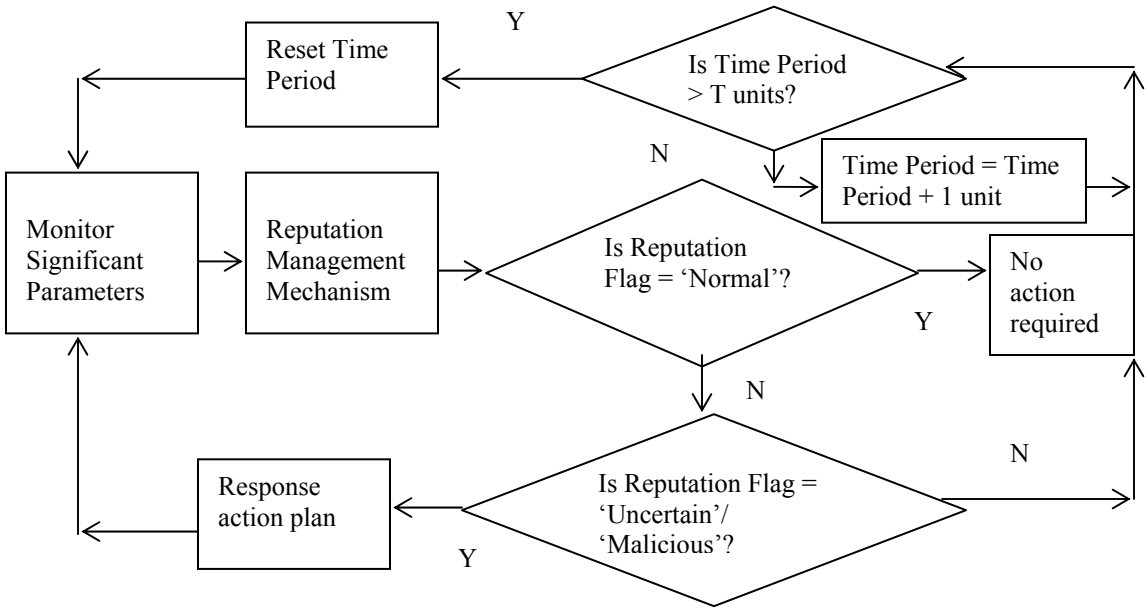


Figure 4.1 Architecture of the IDRMAN Response Framework

Based on the reputation counter that is evaluated, the reputation flag is asserted as “Normal”, “Uncertain” or “Malicious”. If the reputation flag is “Normal”, no action is

necessary. If the reputation flag is “Uncertain” or “Malicious”, response action plans are executed to protect the MANET. The response action plans isolate, disconnect, block or automatically deny future connections to malicious nodes.

Figure 4.1 shows a time period check logic between “No action required” block and “monitor significant parameters” block. If there is no time period check between “No action required” block and “monitor significant parameters” block, the response framework would create an unnecessary overhead. This is due to the reason that continuous monitoring is expensive in terms of resources needed. Hence, a timed periodic check is applied in the framework.

Thus, this framework is based on monitoring the network parameters and performing response action plan on the basis of assertion of reputation flag. Such a framework allows for response to selfish and malicious node attacks. Those malicious and selfish nodes with a specific IP address that generate abnormal metric parameter values are identified and gradually isolated.

4.3 The IDRMAN Response Framework Mechanism

The IDRMAN response and protection framework gets invoked when the Threat Index computed by the detection framework (explained in Chapter 3) is in the vulnerable state. The response and protection framework identifies the intruder and responds to the attack with the response action plan.

Counters and flag are associated with each neighboring source node to the node under attack to implement the response algorithm. Each node has three types of counters – normal, uncertain and abnormal counter; the flag at each node can take three different values – normal, uncertain and malicious; the nature of the neighboring node is indicated

by the value that its flag takes. For each node which is a neighbor to the node under attack, the response and protection mechanism compares the value of the significant attack sensitive network parameters with the uncertain state and vulnerable state threshold values and updates the counters. The normal counter is incremented when the value of a significant parameter is in normal state, the uncertain counter is incremented when a significant parameter value is in uncertain state and the abnormal counter is incremented when a significant parameter value is in vulnerable state. Since all the significant parameters that are selected have equal importance (as explained in Step 2 of Section 3.4 in Chapter 3), they have equal weight in the intruder identification and response mechanism. After the counters are incremented, the flag for the node is asserted using the following logic: If the value of the normal counter is greater than sum of the uncertain and abnormal counters, "normal" flag is asserted. If the value of the uncertain counter is greater than the sum of normal and abnormal counters, "uncertain" flag is asserted. If the value of the abnormal counter is greater than the sum of normal and uncertain counters, "malicious" flag is asserted. If all the counter values are equal, the "uncertain" flag is asserted. Based on the flag that is asserted, different response action plans are triggered. The response action plans are explained as follows:

Response Action Plans:

Response actions are required when the flag of a neighboring source node is "uncertain" or "malicious. The following action plans could be used in the response framework.

Action Plan 1: If a neighboring source node to a node under threat is flagged as "normal", no action is needed since the node is neither malicious nor selfish. For this

level, the following could be executed [91].

- Basic computer security policy like basic encryption, authentication, authorization and so forth.
- Specially designed core software for proactive security.
- Set the maximum concurrent connections allowed per user.
- Powerful firewall capable of filtering a wide range of traffic based on extensive set of historical traffic characteristics. Provision of ability to add the detected traffic characteristics to a database.
- Set the bandwidth at a lower acceptable level than the possible maximum.

Action Plan 2: If a neighboring source node to a node under threat is flagged as "uncertain", necessary precautions are needed to prevent further damage. In this case the following action plans could be executed:

- This plan executes moderate response action like automatic node re-authentication.
- Verify the correct execution of the packet forwarding function.
- Automatic modification of the routing table information to the original state, in order to bring the system to original state.
- Automatic modification of the propagation limits of the ad hoc nodes, in order to perform the packet forwarding function.
- Reduce the maximum concurrent connections allowed per user.
- Block specific IP address.
- Automatically deny future connections to this address.
- Ability to drop idle connections.
- Increase the bandwidth.

- Filter “redundant” data packets or using routing information (filter spoofed packets traveling unexpected routes from their specified addresses) [91].

Action Plan 3: If a neighboring source node to a node under threat is flagged as “malicious”, action plan 3 is fired instantly to protect the system. Actions that could be executed for this plan include:

- Drastic action like cutting off the node and restoration of the links. This is achieved by allowing nodes in MANET to observe several types of abnormal behavior makes it possible for the nodes to route around the misbehaved nodes and isolate them or delete the path containing malicious nodes.
- Immediately close the connection (Server will not wait to receive any data). If one requires a message to be sent back to user (agent), the redirection feature should be used instead of deny feature, to redirect specific document to specific users (agent).

Thus, with the help of the intruder identification mechanism and action plans, malicious nodes that create threat to the components in the network will be removed from the network. Response could be achieved by tracking the addresses of the nodes that generate abnormal values for the selected metrics. Those nodes with the specific address can then be disconnected or blocked or automatically denied future connections from accessing the network. This is achieved by means of modifying the routing protocol that controls the nodes participating in the network.

4.4 Intruder Identification and Response Algorithm for IDRMAN

Intruder Identification and Response algorithm is shown in Figure 4.2

1. Let $N_1, N_2, \ldots N_k$ be the nodes which are detected to be under threat based on Intrusion Detection Algorithm.

2. Let $x_1, x_2, \ldots x_n$ be the significant metric parameters which have been identified using CART explained in Section 3.4 of Chapter 3.

3. For each node under threat (N_i), where 1< i<k and 'k' being the number of nodes under threat

    3.1. For each adjacent source node, ($M_{i,z}$), where 1<z<m and 'm' being the number of adjacent source nodes to the node under threat

        3.1.1 Initialize its {abnormal, uncertain, normal}$counter_{iz}$

        3.1.2 For each significant parameter (x_j), where 1<j<n and 'n' being the number of significant parameters

            i. Measure the parameter value ($Val(x_{izj})$) for the node ($M_{i,z}$).

            ii. If $Val(x_{izj})$ is in {abnormal state, uncertain state, normal state} increment the {abnormal, uncertain, normal}$counter_{iz}$ of source node ($M_{i,z}$) respectively.

    3.2. For each adjacent source node ($M_{i,z}$),

        3.2.1. if (its $abnormalcounter_{iz}$ is greater than sum of $normalcounter_{iz}$ and $uncertaincounter_{iz}$)
            {set the flag for node ($M_{i,z}$) to malicious}
        else
            if (its $uncertaincounter_{iz}$ is > sum of $abnormalcounter_{iz}$ and $normalcounter_{iz}$)
                {set the flag for node ($M_{i,z}$) to uncertain}
            else
                {set the flag for node ($M_{i,z}$) to normal}
            end-if
        end-if

    3.2.2. if (flag of node $M_{i,z}$ is malicious)
        {Execute Action Plan 3: this plan executes drastic action like cutting off the node and restoration of the links.}
    else
        if(flag of node $M_{i,z}$ is uncertain)
            {Execute Action plan 2: this plan executes moderate response action like automatic re-authentication of the node.}
        else
            {Execute Action Plan 1: No action is needed since the node is neither malicious nor selfish. In order to avoid unnecessary overhead, apply time period check before further collaborative monitoring, for this action plan}
        end-if
    end-if

Figure 4.2 Intruder Identification and Response Algorithm

4.5 Example to Illustrate the Response Framework Mechanism

Let us consider the MANET example of Section 3.6, where our detection mechanism identified node N_1 to be under attack at 200ms. Since the MANET is detected to be under attack, the intrusion response framework gets invoked. Table 4.1 is a combination of Table 3.1 and 3.3. Here the threshold values: UCL_{vs} and UCL_{us} are included for reference, and the parameter values on each link is classified into normal state (NS), uncertain state (US) and vulnerable state (VS) by comparing against the threshold references.

Table 4.1 Threshold values and values of significant parameters during attack

Parameter	UCL_{vs}	UCL_{us}	M_{11} to N_1	M_{12} to N_1	M_{13} to N_1	M_{14} to N_1	M_{15} to N_1
(PD)	208.6336	119.081	155 / US	2000 /VS	20/NS	20/NS	20/NS
(QL)	1157.721	656.014	120 / NS	12000/ VS	120/NS	120/NS	120 / NS
(EC)Joules	1.9941	1.3397	1.30 /NS	3.92/VS	2.33/VS	2.36/VS	2.61/ VS

From Table 4.1, we find that on the link where the neighboring source node is M_{11} and the destination is N_1, QL and EC are in normal state and PD is in uncertain state. So the normal counter is incremented twice and takes a value of 2. Uncertain counter is incremented once and takes a value of 1. Since none of the significant parameters is in vulnerable state, the abnormal counter value is not incremented and remains 0. Since the value of the normal counter is greater than the sum of uncertain and abnormal counters, the "normal" flag is asserted for the link's source M_{11}.

On the link where the source is M_{12} and destination is N_1 PD, QL and EC are in vulnerable state. So the abnormal counter is incremented thrice and takes a value of 3. The normal and uncertain counters are not incremented and they remain at 0. Since the value of the abnormal counter is greater than the sum of uncertain and normal counters, the "malicious" flag is asserted for this link's source M_{12}.

On the link where the source is M_{13} and destination is $N_{1,}$ PD and QL are in normal state, and so the normal counter is incremented twice and takes a value of 2. Since EC is in vulnerable state, abnormal counter is incremented once and hence take a value of 1. Since none of the significant parameters is in uncertain state, the uncertain counter value is not incremented and remains 0. Since the value of the normal counter is greater than the sum of uncertain and abnormal counters, the "normal" flag is asserted for this link's source M_{13}.

On the link where the source is M_{14} and destination is $N_{1,}$ PD and QL are in normal state, and so the normal counter is incremented twice and takes a value of 2. Since EC is in vulnerable state, the abnormal counter is incremented once and results in a value of 1. None of the significant parameters is in uncertain state and so the uncertain counter is not incremented and it remains at 0. Since the value of the normal counter is greater than the sum of abnormal and uncertain counters, the "normal" flag is asserted for this link's source M_{14}.

On the link where the source is M_{15} and destination is N_1, PD and QL are in normal state, and so the normal counter is incremented twice and takes a value of 2. As EC is in vulnerable state the abnormal counter is incremented once and hence results in a value of 1. None of the significant parameters is in uncertain state and so the uncertain counter is not incremented and it remains 0. Since the value of the normal counter is greater than the sum of abnormal and uncertain counters, the "normal" flag is asserted for this link's source M_{15}. Using all the above information, Table 4.2 is formed and is shown as follows.

Table 4.2 Illustration of response framework for IDRMAN

Node Under Threat	Neighboring Nodes	Normal Counter	Uncertain Counter	Abnormal Counter	Flag	Action Plan
N_1	$M_{1,1}$	2	1	0	Normal	Action Plan 1
	$M_{1,2}$	0	0	3	Malicious	Action Plan 3
	$M_{1,3}$	2	0	1	Normal	Action Plan 1
	$M_{1,4}$	2	0	1	Normal	Action Plan 1
	$M_{1,5}$	2	0	1	Normal	Action Plan 1

Each row in Table 4.2 represents the counter, flag of the neighboring source node for the node under threat and the type of the action plan that needs to be taken against the neighboring node. The agent implementation of the routing protocol in mobile ad hoc network has the view of the Table 4.2 and takes appropriate response action based on the counters and the flag value. In Table 4.2, sum of abnormal, uncertain and normal counters for every neighboring node to a node under threat will be equal to the number of significant parameters. This is due to the reason that for every neighboring node to a node under threat, the effect of each significant parameter is analyzed and categorized into three possible states, and the normal, uncertain or abnormal and counters are incremented based on these states.

4.6 Mathematical analysis of the IDRMAN Response Framework

In this section, we mathematically analyze the response model and compute the mean square error, which is the summation of false positives and false negatives in identifying an intruder for a node under threat, of the response framework. False-positive represents the number of incorrectly isolated neighboring nodes for the node under threat and false-negative represents the number of not isolated neighboring nodes which should have been isolated for the node under threat.

Since 'Yes' and 'No' are the only possible outcomes in identifying the true nature of the intruder, where 'Yes' represents the correct Identification and 'No' represents the wrong Identification, the probability of correct identification of the N neighboring nodes can be obtained by applying the binomial distribution:

$$\mathrm{P(X=k)} = C_k^N \times p^k \times (1-p)^{N-k}. \tag{4.1}$$

Here k represents the number of nodes that are correctly estimated using the intruder identification and response framework, and can take any value from 1 to N; p represents the probability of correct identification. The Mean Square Error (MSE) is given by the sum of the variance and the square of the bias i.e.,

$$\mathrm{MSE} = \mathrm{variance} + \mathrm{bias}^2 \tag{4.2}$$

where bias is defined as the distance between the estimator's mean and the parameter's (truth value's) mean [112]. For the binomial probability distribution, variance is given by:

$$\mathrm{variance} = N \times p \times (1-p) \tag{4.3}$$

where p is the probability of the success and N is the number of trials [113].

Proposition:

The flag M_{iz} of the intruder identification and response framework indicates if the neighboring node to the node under attack (i) is malicious or not, such that, if the mean squared error (MSE) defined by E $[(M_{iz} - \text{Truth})^2]$) is computed on each neighboring node for a node under threat (i), then the sum of the MSE on all the neighboring nodes $(\sum_{z=1}^{N} E[(M_{iz} - \text{Truth})^2])$ for a node under threat is approximately equal to $0.09N$, where N is the number of neighboring nodes. Truth is the unknown reality (i.e. the state of the neighboring node) which is to be estimated using M_{iz}.

Proof:

Let M_{iz} represents the flag of the z^{th} neighbor to the node i that is under threat. Let M_{iz} takes a value of '1' and '0' as given by the following definition.

$$M_{iz} = \begin{cases} 1 \text{ if } abnormalcounter_{iz} > normalcounter_{iz} + uncertaincounter_{iz} \\ 0; \text{ otherwise} \end{cases} \quad (4.4)$$

In the above equation, $abnormalcounter_{iz}$ is the abnormal counter value of the z^{th} neighbor to the node i that is under threat; $normalcounter_{iz}$ is the normal counter value of the z^{th} neighbor to the node i that is under threat, and $uncertaincounter_{iz}$ is the value of the uncertain counter of the z^{th} neighbor to the node i that is under threat.

From the experimental results obtained by simulating the intruder detection and response algorithm for DoS attacks on MANET nodes, the probability p of correctly identifying a neighboring node as an intruder node or a normal node is 0.9. Hence, applying equation 4.3, variance= $N \times 0.09$, where N is the number of neighboring nodes for the node under threat. Also based on the sample simulation data, $bias^2$ is estimated to be 0.03. Thus, applying equation 4.2, $MSE = 0.09N + 0.03$. □

4.7 Application of IDRMAN for MANET Attacks

The intrusion detection and response of the IDRMAN can be applied to protect MANET from several types of attacks. IDRMAN can be applied for both active attacks like Denial of Service, unauthorized access and passive attacks like Masquerade. Following are the possible MANET attacks where IDRMAN can be applied:

(i) Denial of Service: A node could spam other nodes causing resource constraints by repeatedly sending messages to another node. This may place undue burden on the message handling routines of the recipient. Nodes can also intentionally distribute false or useless information to prevent other nodes from completing their tasks correctly or in a timely manner. In another kind of DoS attack, the local host could also ignore a node's query or retrieval request by turning away the request. Nodes on other platforms waiting for the results from a non-responsive agent in the malicious host platform could cause deadlock problems. In yet another possible DoS attack, a node might consume excess amount of host resources so that the host cannot service other nodes properly. The network parameters like Queue length, Energy Consumption and Packet Drop were identified as the parameters that are sensitive to DoS attacks through the experimentation. By observing the threshold values for DoS attack sensitive parameters, DoS attacks could be identified and responded.

(ii) Unauthorized Access: In these attacks, a node would invoke other node's resources by accessing or modifying its code or data, which could change the behavior of the host from trusted to harmful one. In another possible kind of authorization attack, remote users, processes, and agents may request resources from the host, for which they are not authorized [34]. Reijo Savola has proposed metrics for MANET authentication and

authorization attacks based on the reasoning that in MANET, cryptographic strength has tight cross-relationships with critical control information like trust management information [117]. Some of those metrics are:

Key length metric: The security of symmetric cryptosystem is a function of the length of the key. Since, there are 2^N possibilities to break a key of length N bits by means of brute force attack, every extra key bit generally doubles the number of possible keys and therefore increases the effort required for successful brute force attack against most symmetric algorithms. .

Algorithm Strength Metric: Jorstad and Landgrave use algorithm strength as metric to express the overall measurement of a cryptographic algorithm's strength [117].

Rounds Metric: Rounds are important to the strength of some ciphers. For example, an eight round version of an algorithm like DES is not secure. In general, more rounds lead to more security up to a point.

Attack steps metric: It is defined as the number of steps required to perform the best known attack.

Attack time metric: It is defined as the time that might be required to perform the fastest known attack.

By observing the threshold values for these attack sensitive parameters, unauthorized attack could be identified and responded.

(iii) Masquerade: In this type of attack, a node posing as host could deceive nodes and it harms both the node that is being deceived and the node whose identity has been assumed, especially in societies where reputation is valued and used as a means to establish trust. A node from a system external to MANET can also masquerade as

another node and request services and resources from the MANET for which it is not authorized. The network parameters like percentage of connections to different hosts, protocol type and number of source packets were identified by experimentation as the parameters that are sensitive to Masquerade [84]. By observing the threshold values for these attack sensitive parameters, masquerade could be identified and responded.

Thus our IDRMAN security model could be applied to any of the above attacks by identifying critical network system parameters and their threshold values that are affected by various types of attacks. Though we conducted experiments to identify significant parameters for both active and passive attacks like Denial of Service, Masquerade and Unauthorized access, we performed simulation experiments to detect the attack, identify the intruder and respond to the attacks only for Denial of Service attacks in MANET. By continuously monitoring the threshold values of the attack sensitive network parameters for fixed or mobile hosts in a network, we could measure the relative change in these parameter values and detect the type of attack accurately and protect the system through an effective response.

CHAPTER V

SIMULATION EXPERIMENTATION AND RESULTS

5.1 Overview of the Chapter

This chapter explains the simulation and experimentation carried out to demonstrate the validity and performance of the IDRMAN model described in Chapters 3 and 4. This Chapter is organized as follows:

- Section 5.2 describes the experiments that were conducted to identify the significant network parameters that would help detect an attack and identify the intruder. These experiments were carried out using data mining methodology with the CART tool and the MANET training dataset created through simulation. This section also presents the validation experiment conducted with the help of CART and DARPA benchmark dataset to validate the significant parameters identified using the MANET dataset. This section relates to the Step 2 of the model described in Section 3.4.
- Section 5.3 describes the experiments that were conducted to identify the threshold values of the selected network parameters. These experiments were carried out using six sigma methodology on the MANET training dataset created through simulation. This section relates to Step 3 of the model described in Section 3.4.

- Section 5.4 presents the experimentation to calculate the TI thresholds and also validate these thresholds. These experiments were carried out using the fuzzy tool, MATLAB [101].
- Sections 5.5 through 5.8 describe the simulation experiments of the IDRMAN model conducted using NS2 and the experimental results.
  - Section 5.5 describes the simulation environment for implementing the mobile ad hoc network.
  - Section 5.6 describes the scenario for the simulation experimentation.
  - Section 5.7 describes the experimental results when DSDV is used as the routing protocol. It presents the results of the detection of attack using fuzzy logic based threat index calculation and the results of the intruder identification and response.
  - Section 5.8 presents the results of the threat detection and intruder identification and response when AODV is used as the routing protocol for the model simulation.
- Section 5.9 describes the experiments that were conducted to demonstrate the efficiency and effectiveness of the IDRMAN detection framework compared to related MANET IDA.
- Section 5.10 describes the experiments that were conducted to demonstrate the scalability aspect of the detection framework in the IDRMAN model. The purpose of this experimentation is to illustrate that the IDA scheme used in the IDRMAN performs well even when more mobile nodes are added into the MANET.

5.2 Significant Parameters Identification Experimentation

Experimentation using MANET dataset:

To identify the significant parameters for MANET, we generated training dataset consisting of 200 records by simulating the ad hoc network with DoS attacks using NS2. The MANET simulation environment where the training data is sourced from, is explained in detail in Section 5.5. We then applied CART to the simulated training dataset to identify the significant parameters for DoS attacks [94]. The significant parameters were identified using the variable importance table generated in CART. Figure 5.1 shows the variable importance table generated by CART. Step 2 in Section 3.4 explains in detail with an example, the concept of variable importance identification from a dataset using classification trees. As shown in Figure 5.1, Queue Length, Energy Consumption and Packet drop were identified as the significant parameters through the variable importance table since their variable importance is 100%. All experiments were conducted on Intel Pentium 3 machine with 1 Ghz processor and 512 MB RAM.

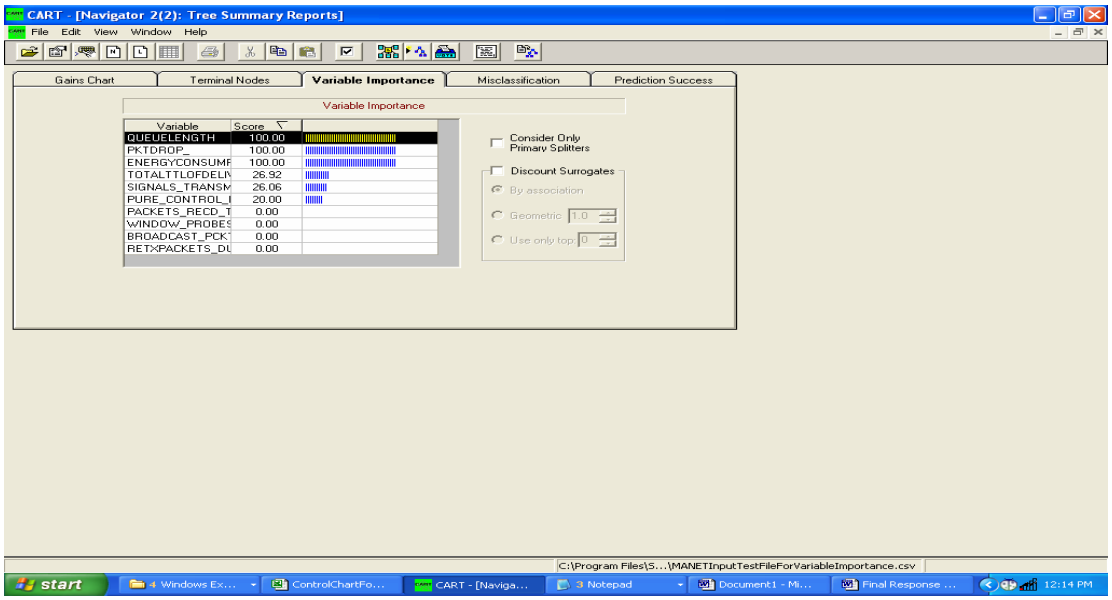


Figure 5.1 Snapshot of Identification of Significant Parameters for MANET Dataset Through Variable Importance Tool in CART

Validation Experimentation using DARPA dataset:

In order to validate the significant network parameters that were identified by applying CART on the MANET training dataset for DoS attack, we performed experiments using the high dimensional DARPA validation benchmark dataset and data mining methodology as described in Chapter 3. The Table 5.1 shows the significant network parameters that were identified for specific attacks like DoS, masquerade and unauthorized access attacks. The significant network parameters were identified using the variable importance table generated by CART. A 'Y' in the column of Table 5.1 indicates that the parameter is affected by the attack and hence it needs to be used to measure the vulnerability of the system. An 'N' in the column of Table 5.1 indicates that the parameter is not affected by the attack. It can be seen that the network parameters identified for DoS attacks through the MANET dataset are detected through the benchmark DARPA dataset as well. 'Queue Length' network parameter identified in the MANET dataset is referred as 'count' in the DARPA dataset [100]. Packet Drop parameter in the MANET dataset is actually a difference between 'Src Bytes' and 'Dst Bytes' identified through the DARPA dataset. 'Energy Consumption' parameter being unique to MANET is not identified as attack sensitive network parameter in DARPA dataset, which is based on wired network.

Unlike DoS and masquerading attacks, there appear to be no sequential patterns for unauthorized access attacks. This is because DoS and masquerading attacks involve many connections to some host(s) in a very short period of time, but the unauthorized access attack patterns are embedded in the data portions of packets, and normally involve only a single connection. Hence, it is unlikely that they can have any unique frequent

traffic patterns. So, we used domain knowledge to construct a set of "content" features to indicate whether the connection contents suggest suspicious behavior. In Table 5.1, except for categorical parameters like protocol and service type, all the parameters are of continuous type. In Table 5.1, the "same host" network parameter examines only the connections in the past two seconds that have the same destination host as the current connection. The "same service" network parameters examine only the destination host connections in the past two seconds that provide same service for the current connection. The "same host" and "same service" network parameters are time-based continuous traffic features. 'SYN' error represents the error due to TCP SYN flood attack. A TCP SYN flood sends erroneous TCP requests to the target system, which cannot complete the connection request. 'REJ' error indicates that the destination node has not received the packets correctly.

Table 5.1 Identification of significant network parameters through DARPA dataset

Network Parameter	DoS	Masquerade	Unauthorized Access
1. Service (Type of network service e.g., http, telnet, etc)	Y	Y	Y
2. Protocol_type (Type of protocol, e.g. tcp, udp, etc)	Y	Y	N
3. Src_bytes (No of data bytes sent from source to destination)	Y	Y	Y
4. Dst_bytes (No of data bytes received at destination from source)	Y	N	Y
5. Count (No of connections to the 'same host' in the past two seconds)	Y	N	N
6. Dst_Host Count (No of connections to the 'same host' using a window of 100 connections instead of a time window)	N	Y	Y
7. Dst_Host Same Source Port Rate (No of connections to the same source port of host)	N	Y	Y
8. Dst_Host Service Count (No of host connections providing the 'same service' for the current connection in the past two seconds)	N	N	Y
9. Dst_Host_Srv_Diff_Host Rate (% of connections to different hosts)	N	Y	N
10. Dst_Host_Srv_ Rerror_Rate (% of host connections that have ``REJ" errors for providing the 'same service')	Y	N	N
11.Dst_Host_Serror Rate (% of host connections that have ``SYN" errors)	Y	N	N
12.Dst_Host_Rerror Rate (% of host connections that have ``REJ" errors)	Y	N	N

5.3 Threshold Identification Experimentation

To identify the thresholds of the significant parameters for MANET, we generated a training dataset consisting of 2000 records by simulating the MANET under normal conditions using NS2. The MANET simulation environment, where the training data is sourced from, is explained in detail in Section 5.5. The values of the significant metrics for the MANET (simulated with DSDV and AODV as routing protocol) from the training dataset were used to calculate upper control limits for vulnerable state (UCL_{vs}) and uncertain state (UCL_{us}) as explained in Chapter 3 and they are shown in Tables 5.2 and 5.3. Since LCL values for the significant parameters are negative and it does not make sense for these parameters, they are not considered and hence not shown in Tables 5.2 and 5.3. Thus only UCL values are used to set the uncertain state and vulnerable state threshold values.

Table 5.2 US and VS threshold values of the metric parameters for DSDV

Metric Parameter	Average	σ (Sigma)	UCL_{vs}	UCL_{us}
Queue Length, QL	545.7273	2181.261	2823.98	1684.853
Energy Consumption, EC (Joules)	0.7702712	1.3813058	2.2053	1.4824
Packet Drop, PD	124.42424	395.4895724	537.4996	330.9619

Table 5.3 US and VS threshold values of the metric parameters for AODV

Metric Parameter	Average	σ (Sigma)	UCL_{vs}	UCL_{us}
Queue Length, QL	154.3077	960.695	1157.72	656.014
Energy Consumption, EC (Joules)	0.6852	1.2532	1.9941	1.3397
Packet Drop, PD	29.5294	171.4791	208.6336	119.0815

These thresholds are not validated for all types of MANETs. For instance, multi hop MANET could have a different threshold values for Queue Length, Packet Drop and Energy Consumption network parameters compared to single hop MANET. However, they can be obtained and validated by performing experiments on the training dataset obtained from simulation of the particular MANET type subjected to attacks.

5.4 Experimentation to Identify and Validate TI Thresholds

As explained in Chapter 3, Threat Index (TI) is an objective measure to identify whether a node is under threat. This section presents the experiment to identify and validate the TI thresholds for the uncertain and vulnerable states. This experiment was carried out with the Fuzzy Inference System (FIS) created using MATLAB [101] and the algorithm shown in Figure 3.9. Another purpose of this section is to validate through experimentation, the relationship between TI and attack sensitive network parameters.

The training data samples with about 150 normal, abnormal and uncertain set of values of significant attack sensitive network parameters were used as the input to the TI detection framework created using MATLAB FIS. The MANET simulation environment where the training data is sourced from, is explained in detail in Section 5.5. TI was then calculated by varying the TI threshold pair (P_1, P_2), where $P_2 > P_1$ and $1<=P_1<=10$ and $1<=P_2<=10$. For each TI threshold pair (P_1, P_2), if the final defuzzified TI was greater than or equal to P_2 and the outcome label was 'Vulnerable', or if the calculated TI was less than or equal to P_1 and the outcome label was 'Normal', or if the TI was between P_1 and P_2 and the outcome label was 'Uncertain', a counter was incremented for that threshold pair. The threshold pair (4, 7) had the highest counter value and hence was chosen as the TI threshold for uncertain and vulnerable state. MATLAB code to implement the TI threshold identification algorithm is attached in Appendix C.

Validation of the Vulnerable Threshold:

Since the vulnerable threshold of TI is more important than the uncertain threshold, the vulnerable threshold (P_2) value of 7 is validated by the methodology below. This validation methodology uses the concepts from Proposition 1 which is explained in

Chapter 3. P_2 value of 7 is validated by checking, if the Mean Square Error (MSE) (explained in proposition 1) computed using P_2 is a minimum.

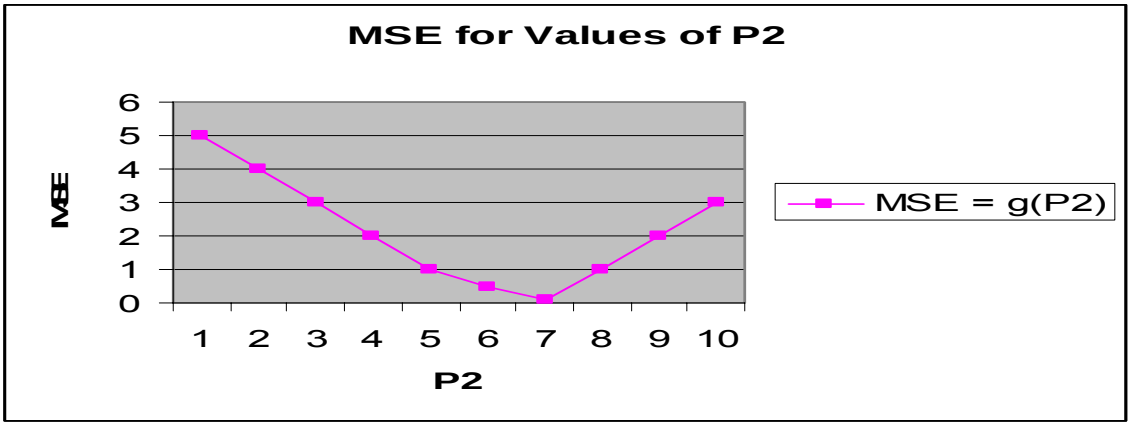


Figure 5.2 MSE for Various Values of P_2

As explained in Proposition 1,

$$\text{MSE} = g(P_2) = \left[\sum_{i=1}^{i=P_2}(P_2 - i)^2 P_V(i)\right].P(V) + \left[\sum_{i=P_2}^{i=10}(i - P_2)^2 P_N(i)\right].P(N) \quad (5.1)$$

Where, $P_V(i)$ represents the probability distribution of TI for vulnerable nodes, $P_N(i)$ represents the probability function of TI for normal nodes, P(N) and P(V) represent the probability of occurrence of normal node and vulnerable node in the MANET.

Our objective is to find the value of P_2 between 1 and 10 such that $g(P_2)$ is minimum. To find this, the value of P_2 was varied from 1 to 10 in intervals of 0.1 and $g(P_2)$ was calculated; The calculation was done using a software program coded using VB attached in Appendix E. The values of $P_V(i)$, $P_N(i)$, P(N) and P(V) are obtained from Section 3.7. The value of P_2 that resulted in the minimum value for $g(P_2)$ (ie. minimum MSE of 0.075) was 6.7. This is very close to the P_2 value of 7 obtained by experimentation using the algorithm shown in Figure 3.9. Hence our choice of P_2 stands validated.

Validation of the Relationship Between TI state and the State of the Parameters:

The basic relationship between TI and the significant parameters is that TI for a node will be in vulnerable state if the values of the parameters are in vulnerable state. Similarly TI for a node will be normal or uncertain if the values of the parameters are normal or uncertain state respectively. Also, as explained earlier in this section, we have higher TI values to represent the abnormal state for the node.

In order to validate this relationship between TI and the attack sensitive network parameters, we input the testing records (which are obtained from the simulated MANET as explained in Section 5.5) that had significant parameters in the vulnerable state to the TI detection framework created using MATLAB FIS to calculate TI. All the TI turned out to be between 7 and 10 meaning the state of these records is vulnerable. The validation experiment was then repeated by inputting the testing records (which are obtained from the simulated network) that had significant parameters in the normal state. All the TI turned out to be between 1 and 4 meaning the state of these records is normal. The simulation results are presented in Figures 5.11 through 5.17, They show that values of TI are in vulnerable state (greater than or equal to 7), when the values of the attack sensitive network parameters are in vulnerable state and the values of TI are in normal state (less than or equal to 4), when the significant variables are in normal state. These results thus validate the relationship between TI and the attack sensitive network parameters.

5.5 Simulation Environment for MANET

Figure 5.3 shows the simulated MANET. Nodes (denoted by Ni) are connected within the mobile ad hoc network environment and mobile agents (denoted by Mi) are dispatched by a source node to a destination node for service purposes [92]. The mobile agents serve as a communication agent between the source and the destination node. The simulation of the MANET was carried out using NS2.

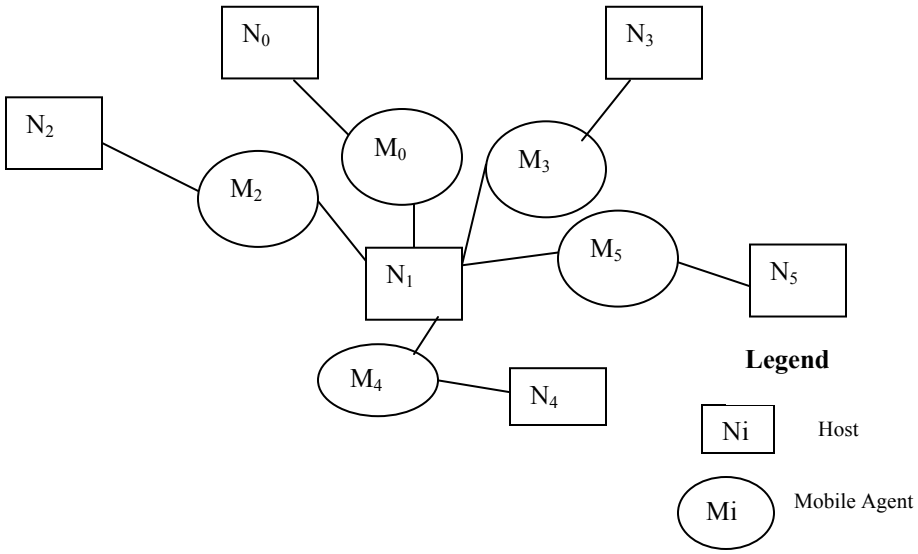


Figure 5.3 Simulated MANET

The description of the NS2 package, its input and output parameters and its use in simulation is explained with details in [95]. The parameters for the MANET to be simulated were specified using the OTcl configuration script. The routing protocol used in NS2 for mobile ad hoc networks was the proactive routing protocol - Destination Sequenced Distance Vector (DSDV). The experiments were also repeated using the reactive routing protocol - Ad hoc On Demand Distance Vector (AODV). The network parameters considered for analysis were: packet drop rate, energy consumption and queue

length based on the identification of significant parameters experiments using the classification trees methodology. The measured values of these three network parameters (metrics) were used in the detection and response framework, to perform intrusion detection, and intruder identification as explained in Chapters 3 and 4.

In order to study the feasibility and performance of the IDRMAN model and to validate our detection and response algorithm, we carried out extensive simulation experiments using various MANET parameters. In our simulation, the channel type was set to wireless channel type and TwoRayGround model was used as the propagation model. The Distributed Coordination Function (DCF) of IEEE 802.11 for wireless LANs was used as the MAC layer protocol. An unslotted carrier sense multiple access (CSMA) technique with collision avoidance (CSMA/CA) was used to transmit the data packets. The radio model was modeled as a shared-media radio with all nodes having the same channel capacity of 2 Mb/s and a transmission range of 250 m.

In the simulation, six mobile nodes were set to move in a 1000 meter x 500 meter rectangular region. Each node was set to move independently with the same average speed. The mobility model we used was the Random Waypoint (RW) model. In RW model, a node randomly selects a destination from the physical terrain. It moves in the direction of the destination in a speed uniformly chosen between the minimal and the maximal speed. After it reaches its destination, the nodes stay there for a pause time and then moves again to a newly selected destination. In our simulation, the minimal speed was 3 m/s, and the maximal speed was 15 m/s. The pause time, which affects the relative speeds of the mobile nodes, was varied. Simulations were run for 1000 simulated seconds.

Constant Bit Rate (CBR) traffic sources were used. The source-destination pairs were spread randomly over the network to generate CBR traffic. The size of all data packets was set to 512 bytes. The OTcl script used to implement the simulation environment is attached in the Appendix A. The hardware used for the simulation was Pentium 3, 512 MB machine. Operating system used for the simulation was Redhat 9.0, Linux Kernel 2.6.

Using this MANET simulation environment, we generated normal and vulnerable (intrusion) data for training and testing purposes explained in Section 5.2 through 5.4. We further generated a different set of normal and vulnerable data to evaluate the performance of IDRMAN under different node mobility and density scenarios to compare its performance with the related models, explained in Section 5.7 through 5.10.

5.6 Scenario for Simulation Experiments

In our simulation experiments, we considered the Denial of Service (DoS) attack on a host node by a set of mobile agent based nodes in MANET. As explained in Chapter 3, these attacks by mobile agents based mobile nodes cause the network to be loaded excessively, thus causing enormous retransmissions, which consumes excessive amount of host resources and hence the host cannot service genuine agents properly. In this case as shown in Figure 5.3, the mobile agents M_0, M_2, M_3, M_4, and M_5 dispatched by host N_0, N_2, N_3, N_4, and N_5 respectively need to get serviced by N_1. However, due to DoS attack by agent M_2, the host N_1 could not service the genuine agents M_0, M_3, M_4, and M_5. In our experiments, the nodes as well as agents were simulated to be mobile. We specified mobility for the nodes using the OTcl script [95]. OTcl script is the interpretor in the NS2 package that is used to specify the simulation parameters.

The basic steps of the simulation experimentation were as follows: Agent M_2 dispatched by node N_2 was configured to send heavy traffic to Node N_1, while all other nodes received normal traffic from their agents. TI evaluation framework implemented using C++ program then calculates TI for each node in the network using the optimized FIS to detect if a node is under attack. The optimized FIS is generated by applying a fuzzy training algorithm on a training dataset that has normal, uncertain and vulnerable state label. This training algorithm generates the optimized FIS with the optimized membership functions that are used by the TI evaluation framework. The TI evaluation C++ source code and MATLAB code which implements our fuzzy training algorithm to generate optimized FIS from the training data are attached in the Appendix B and C respectively.

Node N_1 was detected to be under attack as the calculated TI had a value greater than the vulnerable TI threshold. So its neighboring nodes N_0, N_2, N_3, N_4, and N_5 were subjected to intruder identification and response algorithm. The intruder was identified to be N_2 and it was isolated from the network by disabling its send and receive function in the routing protocol. This was implemented by modifying the C++ code in NS2. The values of the network metric parameters were observed before and after the attack and also after the response. A portion of the observed data of the significant parameters with and without response is attached in Appendix D. After the response, the TI was again calculated and it was found to be in the normal range. The Sections 5.7 and 5.8 give the experimental results for this experiment performed using DSDV and AODV protocols.

5.7 Experimental Results for DSDV

Figure 5.4 shows the values of the evaluated TI without the response and with the response applied for the DoS attack at node N_1 when DSDV is used as the protocol.As seen in Figure 5.4, TI of node N_1 increases due to DoS attacks. Due to subsequent intruder identification and protection measures applied through the response plan, TI for the node N_1, decreases within milliseconds after it is detected and reaches the normal state [86, 93].

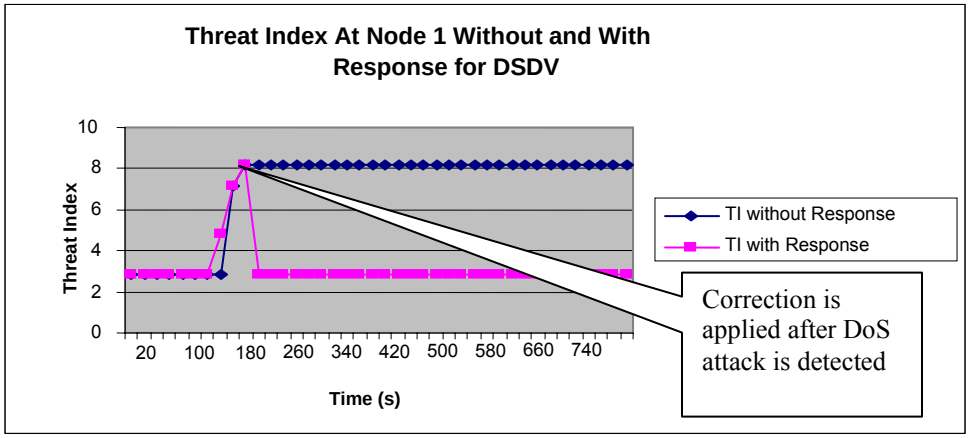


Figure 5.4 Plot of the Evaluated TI for DSDV Experiment

The Table 5.4 shows the values of the normal, uncertain and abnormal counters for each node that is neighbor to node under threat (N_1). The agent implementation (using C++) for the DSDV routing protocol in mobile ad hoc network has the view of the Table 5.4. The values are recorded every 100 ms of the simulation. As explained in Chapter 4, if the value of the parameter exceeds the vulnerable threshold level, abnormal counter is incremented. If the parameter exceeds the uncertain threshold level, uncertain counter is incremented. Otherwise, normal counter is incremented. For malicious node N_2, at around 100 to 200 ms, the abnormal counter reaches the value of 3, thus resulting in the response plan 3 to be executed, to isolate the node. For every other neighboring node at

all the time during the simulation, the normal counter is higher than the sum of vulnerable and uncertain counter, thus resulting in their flag to be normal. The values for the parameters shown in Figures 5.5 through 5.10 validate the value of the counters and the flag in Table 5.4 for each node that is a neighbor to the node under threat (N_1).

Table 5.4 Values of counters used for response action in DSDV experimentation

Node Under Threat	Neighboring Nodes	Time (ms)	Normal Counter	Uncertain Counter	Abnormal Counter	Flag	Action Plan
N_1	N_0	100	2	1	0	Normal	Resp Plan 1
		200	2	1	0	Normal	Resp Plan 1
		300	2	1	0	Normal	Resp Plan 1
		400	2	1	0	Normal	Resp Plan 1
		500	2	1	0	Normal	Resp Plan 1
		600	2	1	0	Normal	Resp Plan 1
	N_2	100	2	0	1	Normal	Resp Plan 1
		200	0	0	3	Malicious	Resp Plan 3
	N_3	100	2	1	0	Normal	Resp Plan 1
		200	2	1	0	Normal	Resp Plan 1
		300	2	1	0	Normal	Resp Plan 1
		400	2	1	0	Normal	Resp Plan 1
		500	2	1	0	Normal	Resp Plan 1
		600	2	1	0	Normal	Resp Plan 1
	N_4	100	2	1	0	Normal	Resp Plan 1
		200	2	1	0	Normal	Resp Plan 1
		300	2	1	0	Normal	Resp Plan 1
		400	2	1	0	Normal	Resp Plan 1
		500	2	1	0	Normal	Resp Plan 1
		600	2	1	0	Normal	Resp Plan 1
	N_5	100	2	1	0	Normal	Resp Plan 1
		200	2	1	0	Normal	Resp Plan 1
		300	2	1	0	Normal	Resp Plan 1
		400	2	1	0	Normal	Resp Plan 1
		500	2	1	0	Normal	Resp Plan 1
		600	2	1	0	Normal	Resp Plan 1

Figures 5.5 through 5.10 give the plot of values of the significant attack sensitive network parameters without the response and with the response applied, for each neighboring node to the node under threat N_1. Figures 5.5 and 5.6 represent control chart for the queue length metric during the DoS attack and after application of response

respectively. As shown in the Figure 5.5, queue length metric for the link between source node N_2 and host is significantly above the vulnerable state threshold (2800) when no response is applied during the attack. After response is applied, the queue length metric for the link between node N_2 and host N_1 is within the normal state threshold control limit as seen in Figure 5.6, since it is cut off from the network.

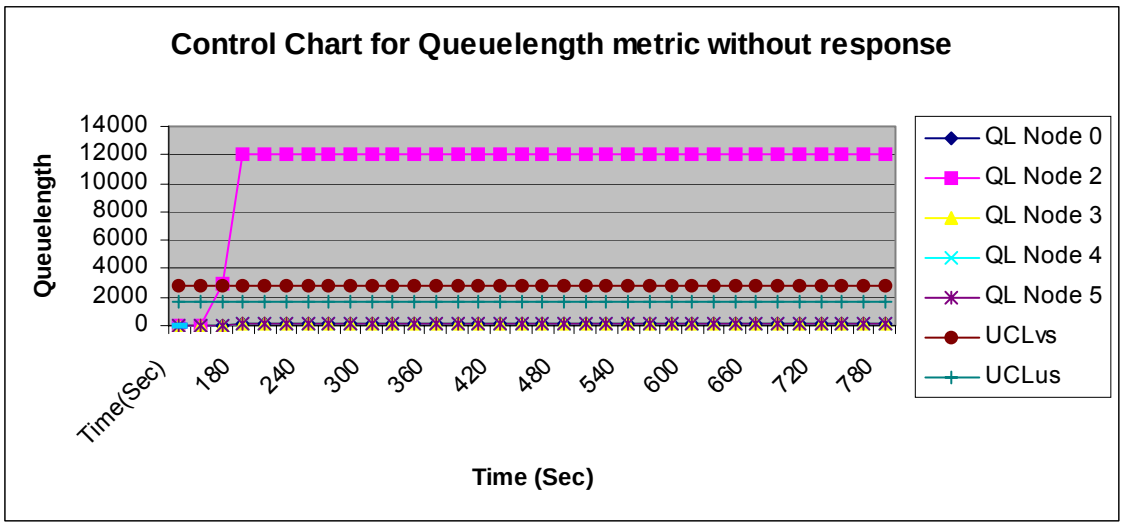


Figure 5.5 Control Chart of Queue Length Metric Without Response for DSDV

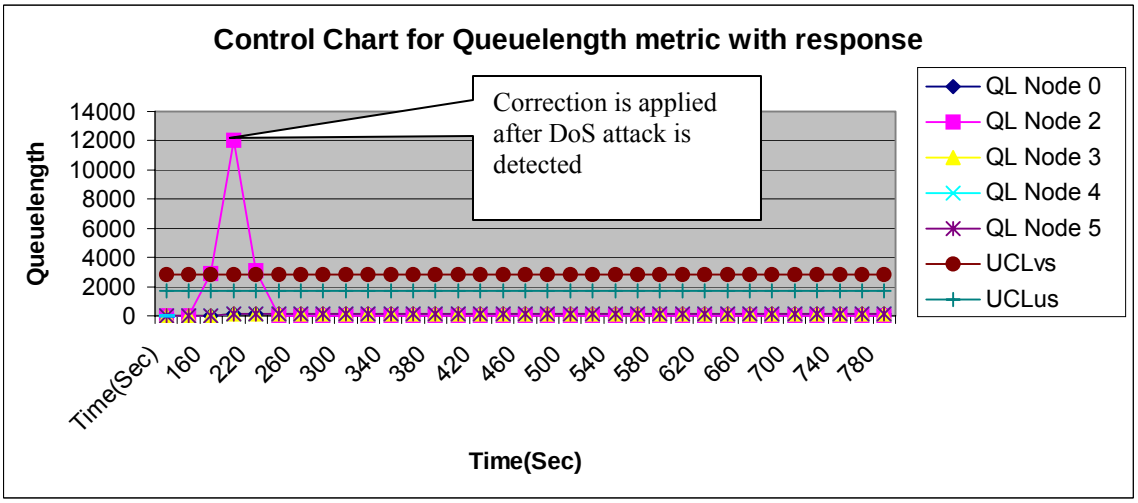


Figure 5.6 Control Chart of Queue Length Metric With Response for DSDV

Figures 5.7 and 5.8 represent the control chart for the packet drop metric during the DoS attack and after the application of response respectively. As shown in Figure 5.7,

the packet drop metric for the link between source node N_2 and host N_1 is significantly above the vulnerable state threshold (537) when no response is applied during the attack. As shown in Figure 5.8, once the malicious node N_2 is identified and isolated from the network, there are no packet drops associated with the malicious node N_2 and the host N_1.

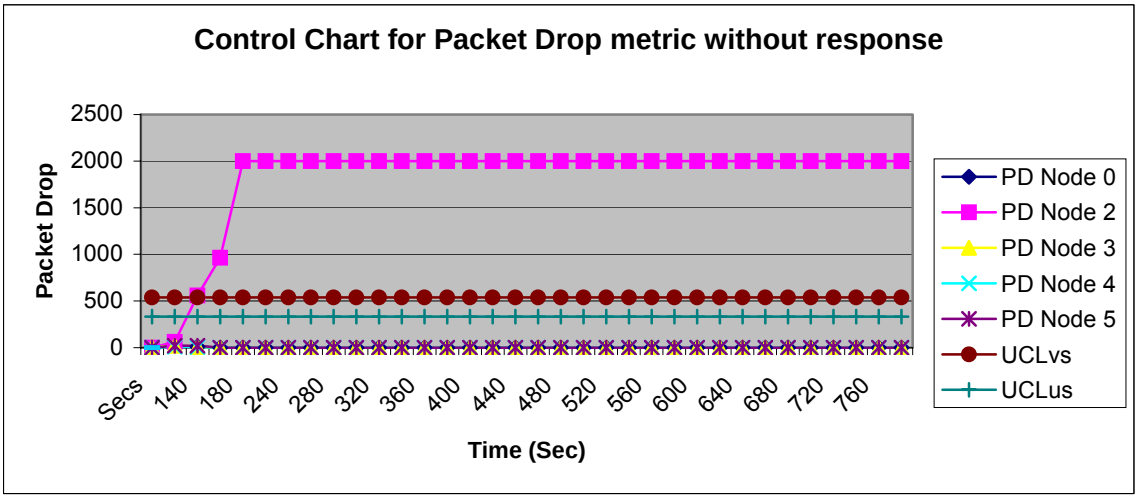


Figure 5.7 Control Chart of Packet Drop Metric Without Response for DSDV

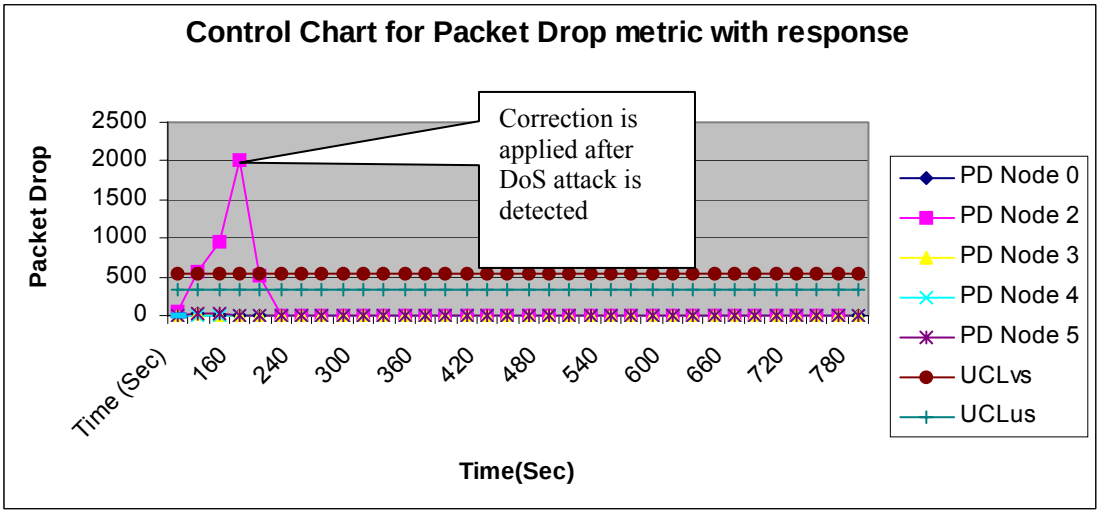


Figure 5.8 Control Chart of Packet Drop Metric With Response for DSDV

Figures 5.9 and 5.10 represent the control chart for the energy consumption metric during the DoS attack and after the response respectively. As shown in Figure 5.9, the energy consumption metric for source node N_2 is significantly above the vulnerable state

threshold (2 joules) when no response is applied during the attack. After response is applied, the energy consumption metric for the source node N_2 is within the normal state threshold control limit as seen in Figure 5.10.

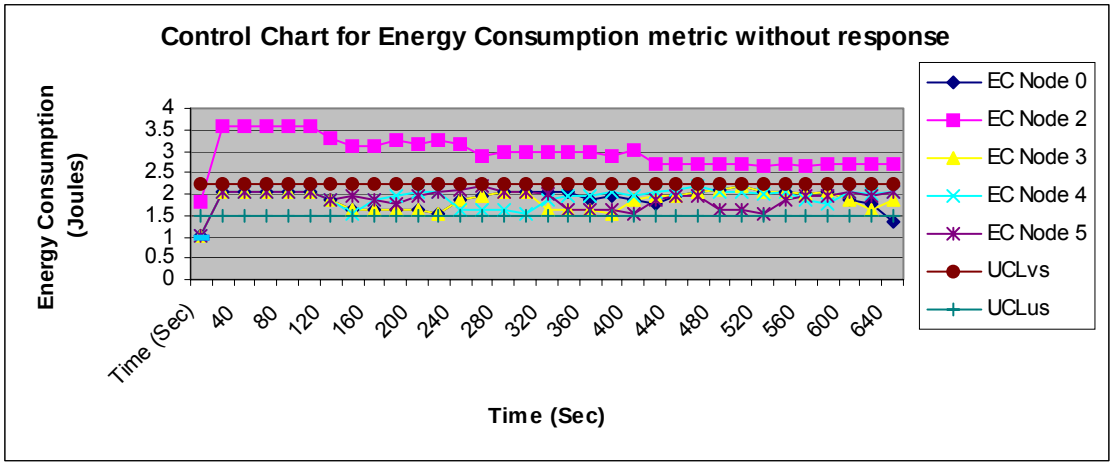


Figure 5.9 Control chart of energy consumption metric without response for DSDV

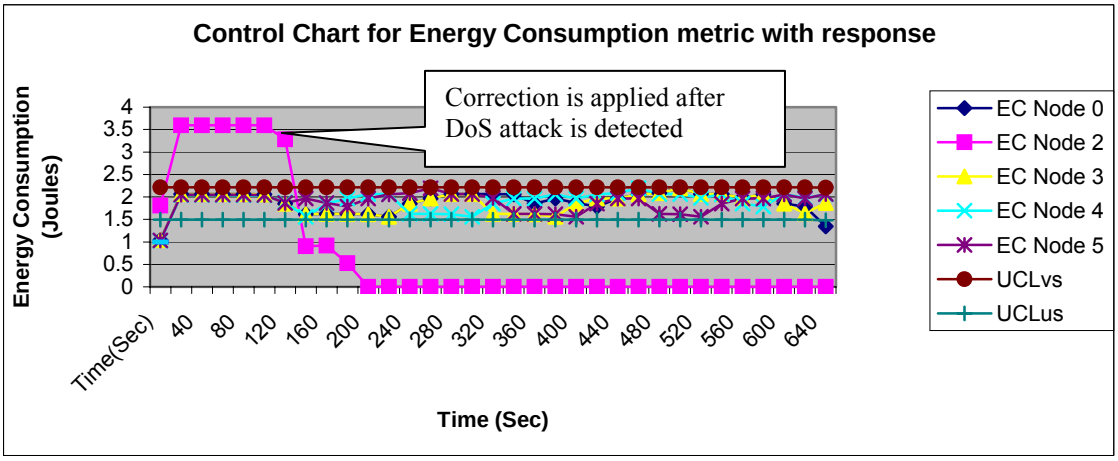


Figure 5.10.Control Chart of Energy Consumption metric with response for DSDV

5.8 Experimental Results for AODV

Figure 5.11 shows the results of the evaluated TI without the response and with the response applied for the DoS attack at node N_1 when AODV is used as the protocol. As seen in Figure 5.11, TI of node N_1 increases due to DoS attacks. Due to subsequent intruder identification and protection measures applied through response plan, TI for the node N_1, decreases within milliseconds, and reaches the normal state.

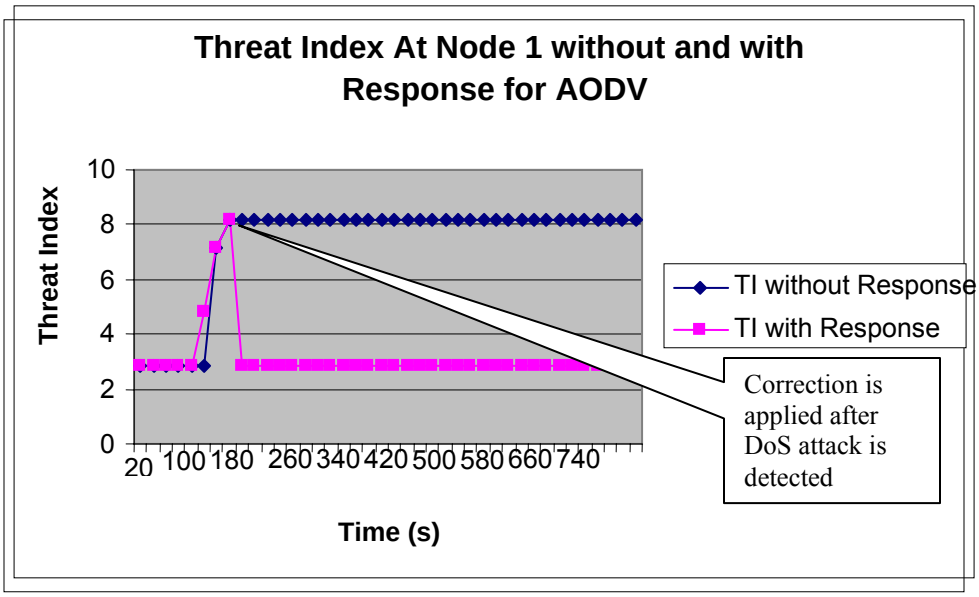


Figure 5.11 Plot of the Evaluated TI for AODV Experiment

Table 5.5 shows the values of the normal, uncertain and abnormal counters of each node that is a neighbor to the node under threat (N_1). The agent implementation (using C++) for the AODV routing protocol in mobile ad hoc network has the view of the Table 5.5. The values are recorded every 100 ms of the simulation. As explained in Chapter 4, if the value of the parameter exceeds the vulnerable threshold level, abnormal counter is incremented. If the parameter exceeds the uncertain threshold level, uncertain counter is incremented. Otherwise, normal counter is incremented. For malicious node N_2, at around 100 to 200 ms, the abnormal counter reaches the value of 3, thus resulting

in the response plan 3 (explained in Chapter 4) to be executed which isolates the node. For every other neighboring node at all the time during the simulation, the normal counter is higher than the sum of vulnerable and uncertain counter, thus resulting in their flag to be normal. The values of the parameters shown in Figures 5.12 through 5.17 justify the value of the counters and flag in Table 5.5 for each node that is a neighbor to N_1, the node under threat.

Table 5.5 Values of counters used for response action in AODV experimentation

Node Under Threat	Neighboring Nodes	Time (ms)	Normal Counter	Uncertain Counter	Abnormal Counter	Flag	Action Plan
N_1	N_0	100	2	1	0	Normal	Resp Plan 1
		200	2	1	0	Normal	Resp Plan 1
		300	2	1	0	Normal	Resp Plan 1
		400	2	1	0	Normal	Resp Plan 1
		500	2	1	0	Normal	Resp Plan 1
		600	2	1	0	Normal	Resp Plan 1
	N_2	100	2	0	1	Normal	Resp Plan 1
		200	0	0	3	Malicious	Resp Plan 3
	N_3	100	2	1	0	Normal	Resp Plan 1
		200	2	1	0	Normal	Resp Plan 1
		300	2	1	0	Normal	Resp Plan 1
		400	2	1	0	Normal	Resp Plan 1
		500	2	1	0	Normal	Resp Plan 1
		600	2	1	0	Normal	Resp Plan 1
	N_4	100	2	1	0	Normal	Resp Plan 1
		200	2	1	0	Normal	Resp Plan 1
		300	2	1	0	Normal	Resp Plan 1
		400	2	1	0	Normal	Resp Plan 1
		500	2	1	0	Normal	Resp Plan 1
		600	2	1	0	Normal	Resp Plan 1
	N_5	100	2	1	0	Normal	Resp Plan 1
		200	2	1	0	Normal	Resp Plan 1
		300	2	1	0	Normal	Resp Plan 1
		400	2	1	0	Normal	Resp Plan 1
		500	2	1	0	Normal	Resp Plan 1
		600	2	1	0	Normal	Resp Plan 1

Figures 5.12 through 5.17 give the plot of the values of the significant attack sensitive parameters without the response and with the response applied, for each

neighboring node to the node under threat N_1. Figures 5.12 and 5.13 represent the control chart for the queue length metric during the DoS attack and after the application of the response respectively. As seen in the Figure 5.12, queue length metric for the link between source node N_2 and host N_1 is significantly above the vulnerable state threshold (1157) when no response is applied during the attack. After the response is applied, the queue length metric for the link between node N_2 and the host N_1 is within the normal state threshold control limit as seen in Figure 5.13, since it is cut off from the network.

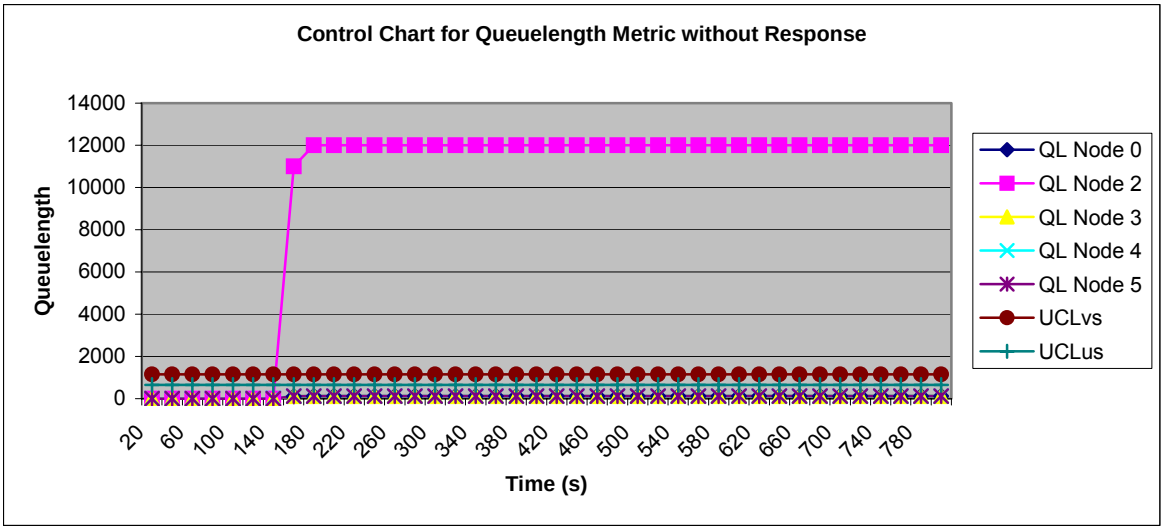


Figure 5.12 Control Chart of Queue Length Metric Without Response for AODV

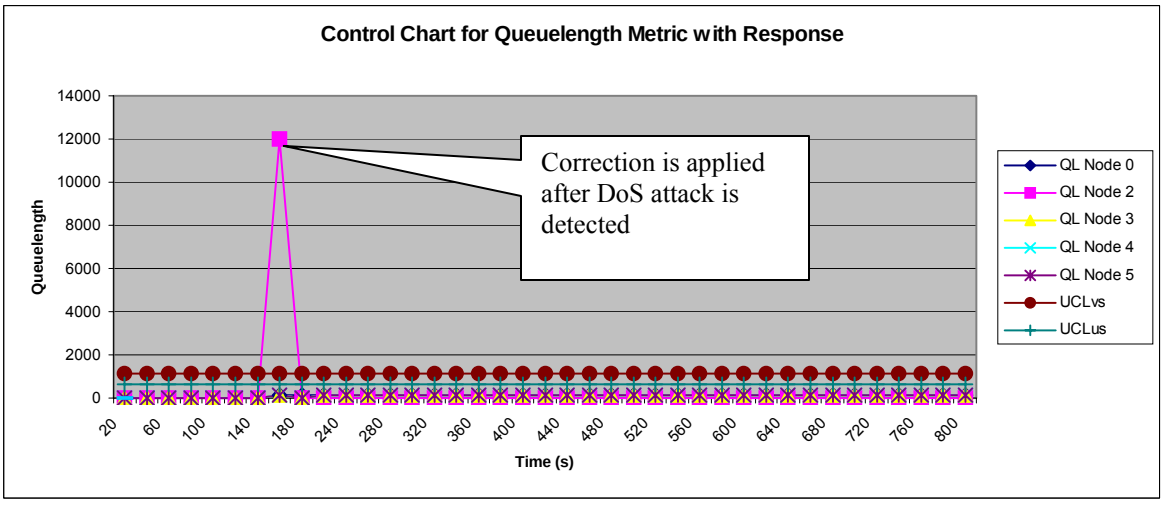


Figure 5.13 Control Chart of Queue Length Metric With Response for AODV

Figures 5.14 and 5.15 represent the control chart for the packet drop metric during the DoS attack and after the application of response respectively. As shown in Figure 5.14, the packet drop metric for the link between source node N_2 and host is significantly above the vulnerable state threshold (208) when no response is applied during the attack. As shown in Figure 5.15, once the malicious node N_2 is identified and isolated from the network, there are no packet drops associated with the malicious node N_2 and host N_1.

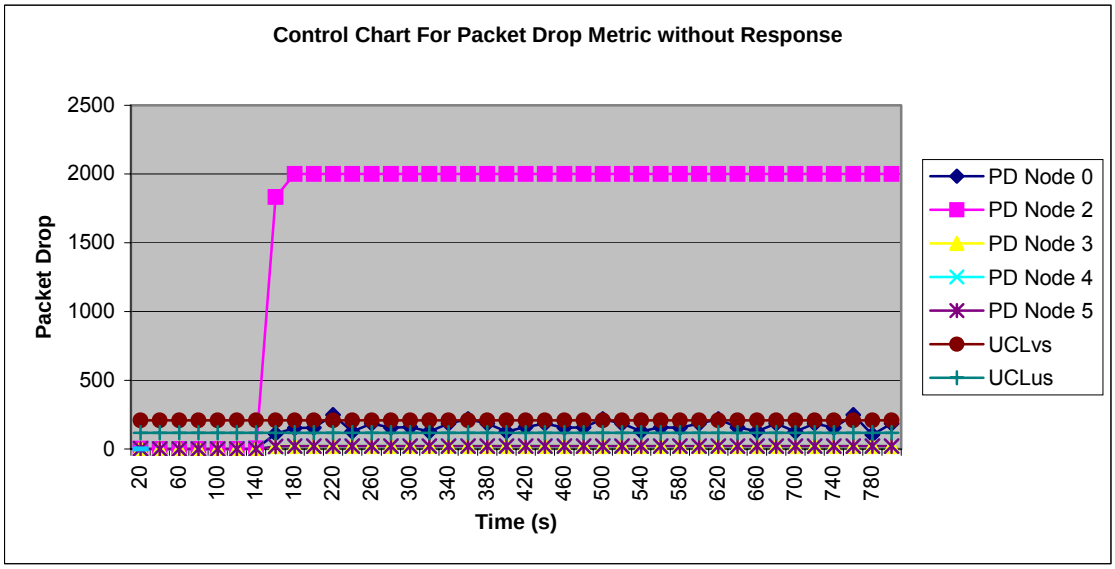


Figure 5.14 Control Chart of Packet Drop Metric Without Response for AODV

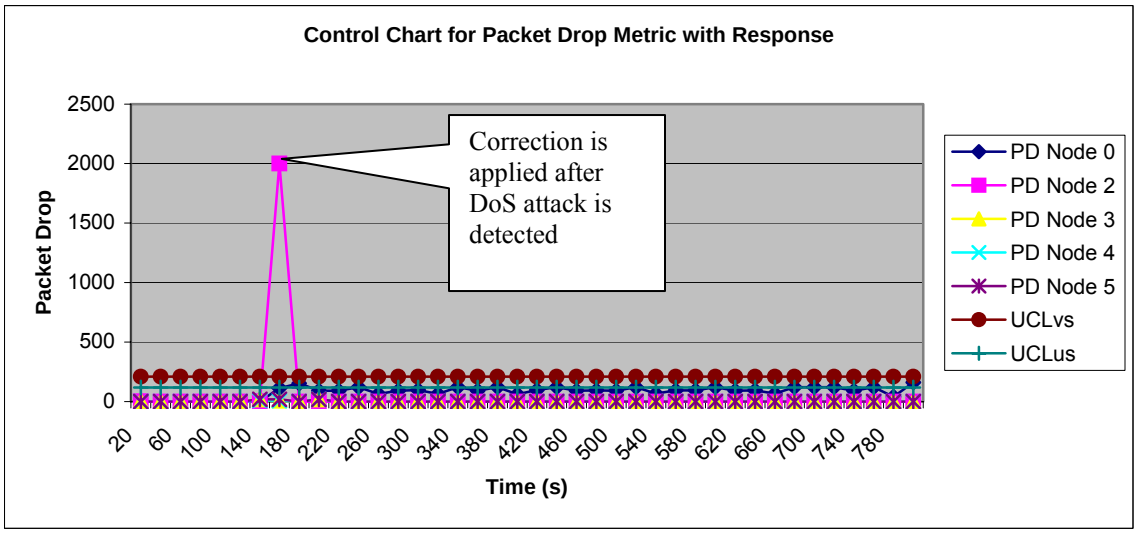


Figure 5.15 Control Chart of Packet Drop Metric With Response for AODV

Figures 5.16 and 5.17 represent the control chart for the energy consumption metric during the DoS attack and after the response respectively. As shown in Figure 5.16, the energy consumption metric for source node N_2 is significantly above the vulnerable state threshold (2 joules) when no response is applied during the attack. After response is applied, the energy consumption metric for the source node N_2 is within the normal state threshold control limit as seen in Figure 5.17.

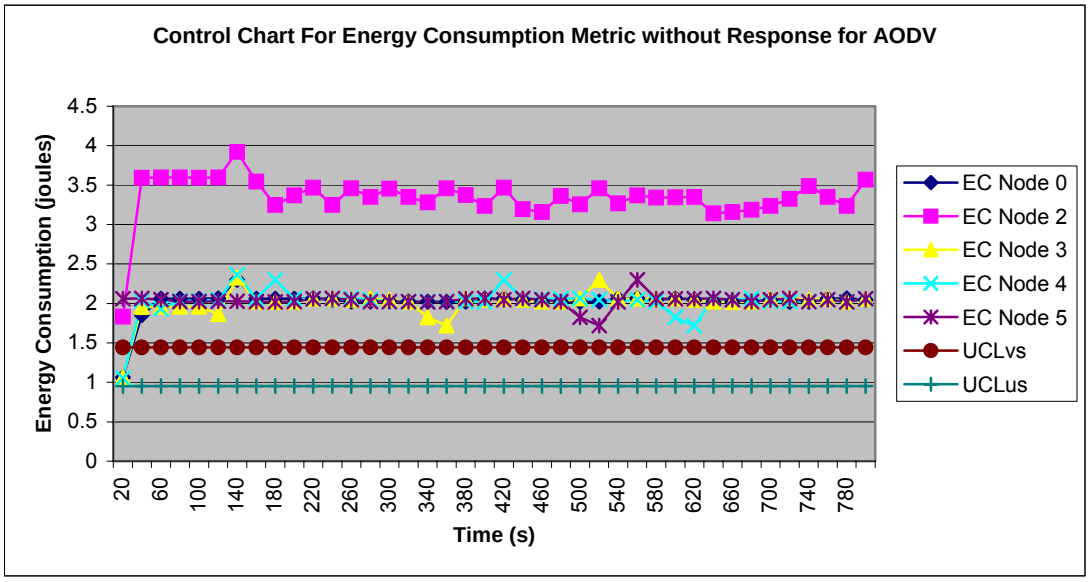


Figure 5.16 Control chart of energy consumption metric without response for AODV

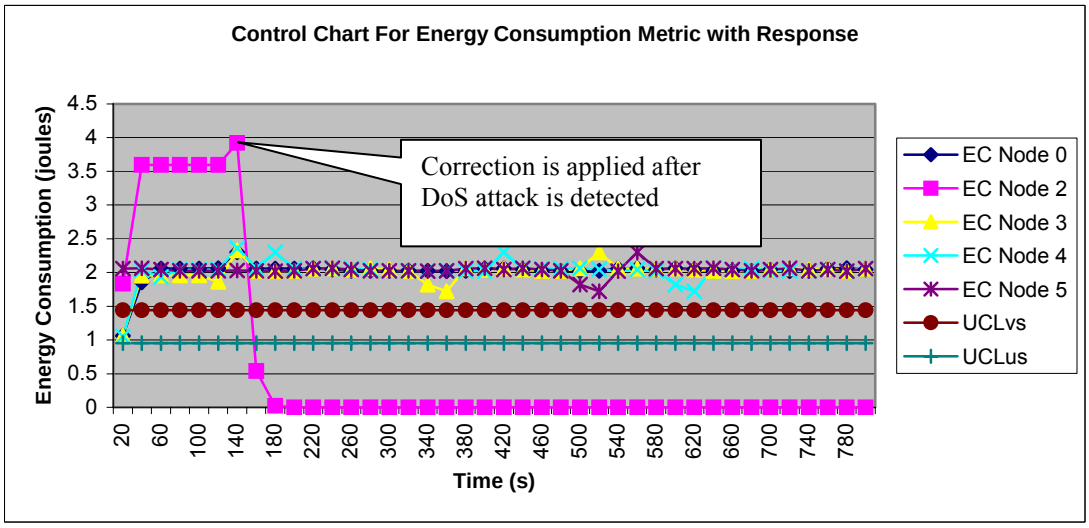


Figure 5.17 Control chart of energy consumption metric with response for AODV

Thus the vulnerability of the node is detected from the TI values and upon detecting that a node is under threat, the neighboring nodes are subjected to the response and protection mechanism, which identifies the intruder and isolates it. This stabilizes the TI of the node under threat and hence the entire network.

As illustrated in Tables 5.2 and 5.3, the values of the metric parameters for AODV and DSDV are different. Specifically, average values for EC, QL and PD are lesser for AODV. This is due to the reason that under stressful load, AODV tends to perform better in routing than DSDV. In general, AODV tends to generate less routing load than DSDV due to its on-demand mechanism of route discovery and route maintenance. However, as shown in Figures 5.4 through 5.17, without response and with response results for DSDV and AODV are similar. This illustrates that the proposed model is protocol independent and can be applied to any routing protocol.

5.9 Related Model Performance Evaluation Experimentation

This section explains the performance evaluation experimentation and comparison of results of the proposed model with related schemes to illustrate the effectiveness of the detection framework used in our IDRMAN model. This section demonstrates that the detection framework used in the IDRMAN model performs better compared to related MANET IDA schemes.

The performance of the proposed IDRMAN detection framework is compared with similar MANET IDA models like Cooperative Intrusion Detection System for Ad hoc Networks (CIDSAN) and Integration of Mobility and Intrusion Detection for Wireless Ad hoc Networks (MIDWAN) [58, 64]. These related models are explained in detail with results in Chapter 2. MIDWAN and CIDSAN are chosen for the comparison

because they are the latest anomaly detection based Intrusion Detection Model for MANET referred in the literature. CIDSAN is an improvement over their previous pioneer work [18]. It is also referred by MIDWAN as the most current anomaly detection based related MANET IDA at the time of the publication of their work.

The Simulation environment and scenario for this experiment is same as the one explained in Section 5.5 and Section 5.6. However, the number of nodes used is around 30 mobile nodes with a minimum speed of 5m/s and the maximum speed of 40 m/s. Also the number of malicious nodes is set to 5. AODV is used as the routing protocol.

Metrics for Performance Evaluation:

The following metrics were chosen for comparing the performance of IDRMAN IDA approach with MIDWAN and CIDSAN.

Detection rate at varied mobility speed: It is defined as the percentage or fraction that a mobile node under threat is correctly detected. This is chosen since accuracy is one of the most important characteristics of an IDA. A high detection rate is desirable for a good MANET IDA.

False Positive rate at varied mobility speed: It is defined as the percentage or fraction that a mobile node has been falsely classified to be under threat. This parameter represents sensitivity to the noisy training data. A good IDA should adapt better even to the unseen data, even if the data representation has some irrelevant information.

Experimental Results:

The accuracy feature of the related IDA is shown in the following Figure 5.18. It shows the detection rate metric results of IDRMAN IDA at varied mobility speed compared to the results of MIDWAN and CINSAN.

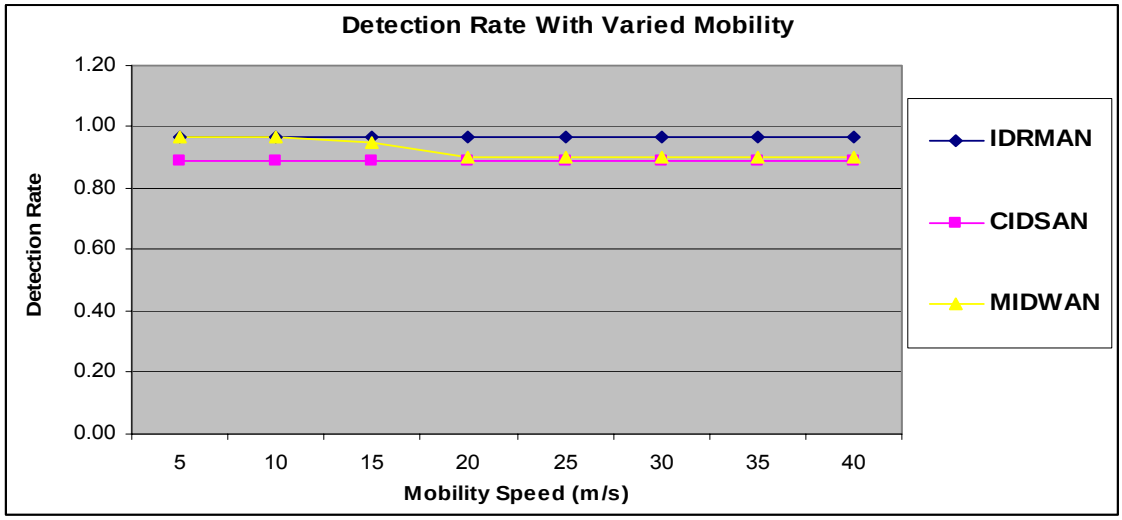


Figure 5.18 Detection Rate at Varied Mobility Speed for Related IDA Evaluation

For detection rate metric at varied node mobility speed, IDRMAN IDA approach works better at various node mobility speeds compared to CIDSAN and MIDWAN. For our IDRMAN IDA approach, the detection rate of 97% is consistent at varied mobility speed. MIDWAN has detection rate of about 97% until 10 m/s. As speed increases the detection rate slightly falls to 90% at 40 m/s. Whereas CIDSAN's detection rate is consistent at about 89% from 5 m/s to 40 m/s.

The noise tolerance feature of the IDA is shown in the following Figure 5.19. It shows the results of IDRMAN IDA at varied mobility speed compared to the results for MIDWAN and CIDSAN approaches for the false positive rate metric.

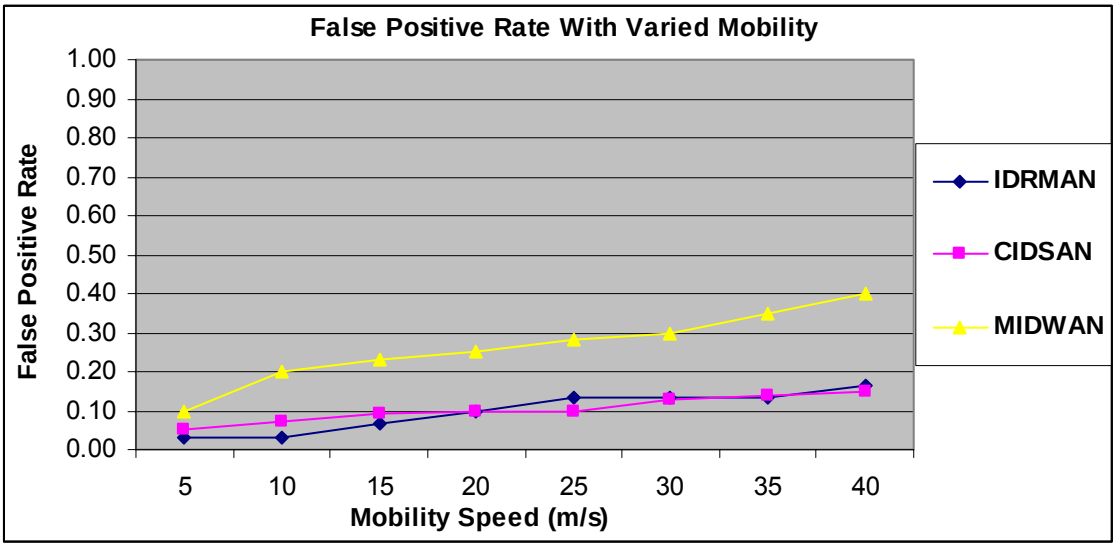


Figure 5.19 False Positive Rate at varied Mobility Speed for Related IDA Evaluation

For false positive rate metric at varied node mobility speed, IDRMAN IDA approach works better at different mobility speeds compared to CIDSAN and MIDWAN. IDRMAN has false positive rate of about 3% until 10 m/s. As speed increases the false positive rate slightly increases to 13% at 25 m/s and then to 17% at 40m/s node speed. While CIDSAN has false positive rate of about 5% until10 m/s and as speed increases, the false positive rate slightly increases to 10% at 25 m/s and then to 15% at 40 m/s. Whereas MIDWAN's false positive rate is about 10% at 5 m/s but it gets significantly increase to about 25% at about 20m/s and increase further at mobility levels higher than 20m/s.

Thus, the performance of IDRMAN detection framework is superior for both metrics, Detection rate at varied mobility speed and false positive rate at varied mobility speed. These two metrics are the most important factors for evaluating a detection scheme. This hence demonstrates that the IDRMAN detection framework performs better compared to the related MANET IDA.

5.10 Scalability Performance Evaluation Experimentation for IDRMAN

This section explains the performance evaluation of the IDRMAN detection framework when more mobile nodes are added into the network at varied mobility speed. The results of this section demonstrate the effectiveness of IDRMAN detection framework from the scalability perspective with respect to the size of the mobile network.

Simulation environment and scenario for this experiment is same as the one explained in Section 5.5 and Section 5.6. However, the number of mobile nodes is varied from 5 to 100 in steps of 10 with the minimum speed of 3m/s and the maximum speed of 15 m/s. Also the number of malicious nodes is also varied. AODV is used as the routing protocol.

Metrics for Performance Evaluation with respect to scalability:

The following metrics were chosen for evaluating the performance of IDRMAN detection framework from the scalability perspective.

Detection rate: It is defined as the percentage or fraction that a mobile node under threat is correctly detected. This is chosen since accuracy is one of the most important characteristics of an IDA. A high detection rate for a large sized MANET is desirable for a scalable MANET IDA.

False Positive rate: It is defined as the percentage or fraction that a mobile node has been falsely classified to be under threat. This parameter represents sensitivity to the noisy training data. A good scalable IDA with large number of mobile nodes should adapt better even to the unseen data, even if the data representation has some irrelevant information.

Total Processing Time: It is defined as the total time the system takes to analyze parameters and detect that the node is under threat. This is an important metric since effective intrusion detection should happen quickly even when there are a large number of mobile nodes, so that the response can be applied before significant damage occurs to the MANET.

Experimental Results:

The accuracy feature of the scalable IDA is shown in the following Figure 5.20. It shows the results of IDRMAN IDA when more mobile nodes are added into MANET at varying mobile speed.

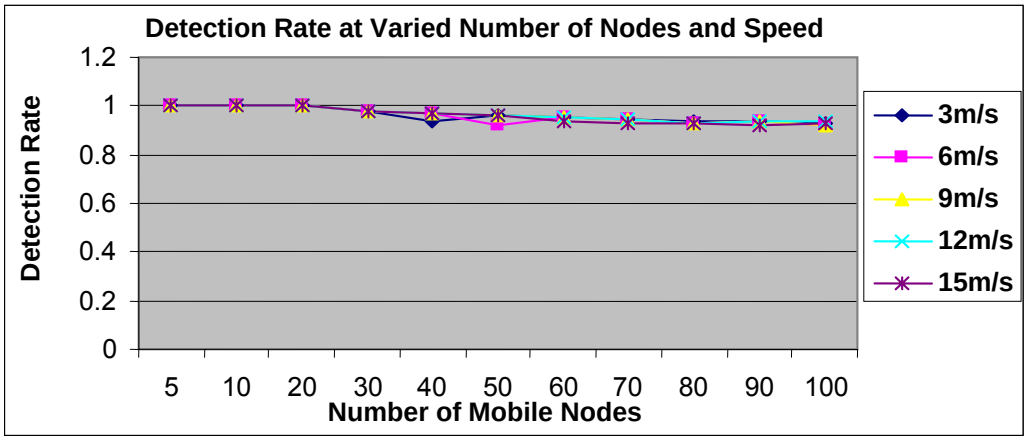


Figure 5.20 Detection Rate Metric Results for Varied Number of Mobile Nodes

For detection rate metric, IDRMAN IDA approach works slightly better when there are less mobile nodes compared to higher number of mobile nodes. The detection rate is close to 100% when the number of mobile nodes is less than 20. But as the number of mobile nodes increases, the detection rate falls to 98% when the number of nodes is 30, then to 95 % when the number of nodes is 60 and finally to 93% when the number of nodes is 100. Also the detection rate is slightly better when the node mobility speed is less.

The noise tolerance feature of the IDA for varying MANET size is shown in the following Figure 5.21. It shows the results for the false positive rate metric of the IDRMAN IDA when more mobile nodes are added into the MANET at varied speed.

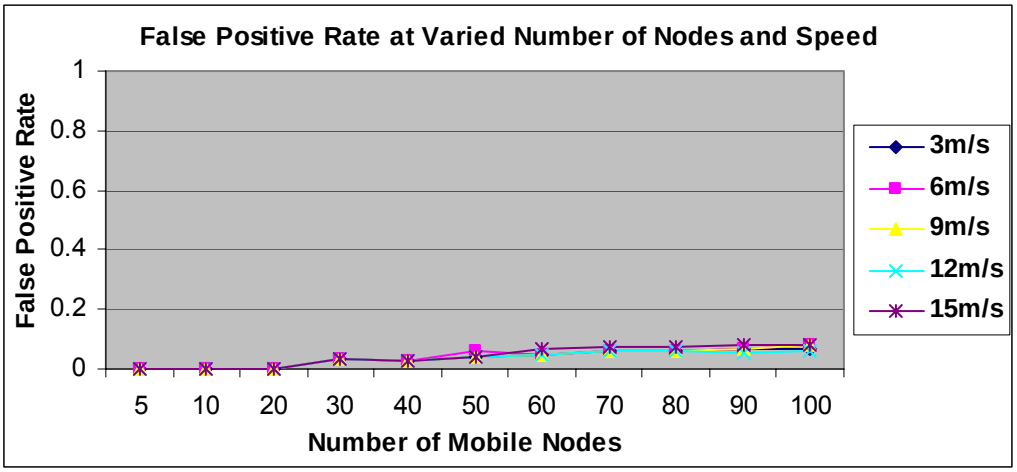


Figure 5.21 False Positive Rate Metric Results for Varied Number of Mobile Nodes

For false positive rate metric, IDRMAN IDA approach works slightly better when there are less mobile nodes compared to higher number of mobile nodes. The false positive rate is close to 0% when the number of mobile nodes is less than 20. But as the number of mobile nodes increases, the false positive rate increases to 3% when the number of nodes are 30, then to 5 % when the number of nodes are 60 and finally to about 7% when the number of mobile nodes are 100. Also the false positive rate is slightly smaller when the node mobility speed is less.

Figure 5.22 shows the processing time metric of IDRMAN IDA when more mobile nodes are added into the MANET at varying speed. It shows the results for the IDRMAN IDA model when the number of mobile nodes is increased from 5 to 100 in steps of 10.

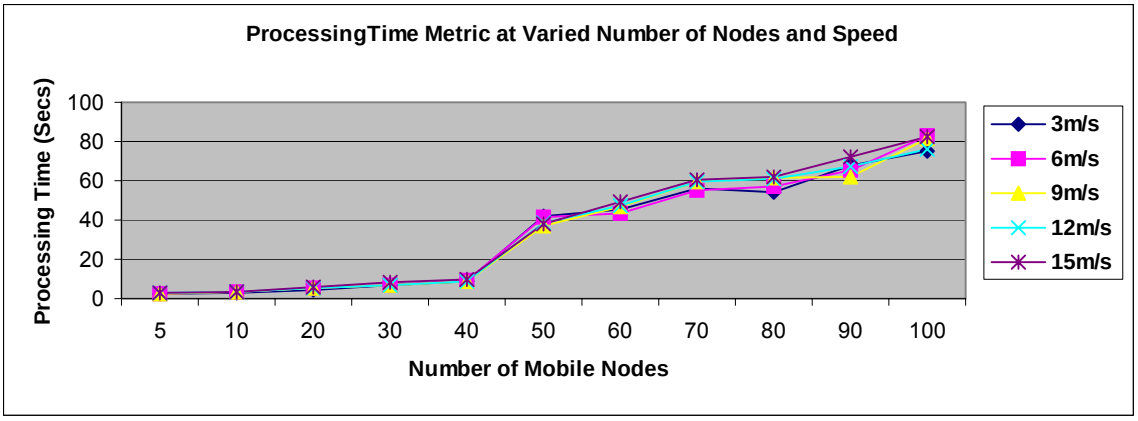


Figure 5.22 Processing Time Metric Results for Varied Number of Mobile Nodes

For Processing Time Metric, IDRMAN IDA approach performs better when there are less mobile nodes compared to higher number of mobile nodes. The processing time is less than 10 seconds when the number of mobile nodes is less than 40. But as the number of mobile nodes increases, the processing time increases to 50 seconds when the number of nodes are 60, then to 60 seconds when the number of nodes are 70 and finally to about 80 seconds when the number of mobile nodes increase to 100. Also the processing time is slightly less when the node mobility speed is less.

Thus, from the metrics used to evaluate the scalability aspect of the IDRMAN IDA, the performance of IDRMAN detection framework is superior when the number of mobile nodes is less than 40 and the performance falls only slightly as more mobile nodes are added to the MANET. This demonstrates that the IDRMAN detection framework is scalable and can perform well even when more mobile nodes are added into the network.

CHAPTER VI

CONCLUSION

This chapter summarizes the research work, reviews the contributions and suggests possible future work.

6.1 Summary of the Research

This research designed and developed the intrusion detection and response model for mobile ad hoc networks (IDRMAN). This model consists of the intrusion detection framework and the intrusion response framework. These two frameworks complement each other to make a complete intrusion detection based security model for MANETs. The functionality and effectiveness of this model was validated by applying this model for simulated Denial of Service attacks(DoS) in MANET.

The detection framework of the IDRMAN uses CART based data mining methodology to identify the parameters that are significant for a particular attack. It then uses the six sigma methodology to set the thresholds for the significant parameters identified. The detection framework then quantifies an attack using a metric called Threat Index (TI) by applying fuzzy logic on the measured values of the significant parameters.

The response framework of the IDRMAN is invoked when the detection framework identifies an attack. The response framework has an intruder identification component and an intrusion response component. The intruder identification component uses the significant parameters and their thresholds in a reputation management mechanism to identify the intruder and flag its status. The response action plan is then

executed by the response framework based on the intruder status indicated by the reputation management mechanism.

The model was simulated for selected Denial of Service attacks with DSDV and AODV as the routing protocols. The results indicate that the threat index evaluation based intrusion detection framework and the intruder identification based intrusion response framework can be used to detect and respond to an attack effectively and is protocol independent.

Extensive simulations were performed to evaluate the performance of proposed model. Simulation results demonstrate that the proposed fuzzy logic based threat detection algorithm has better false positive ratio and detection ratio at varied mobility rates compared to related models. Also, the false positive ratio and detection ratio performance by the model is not affected when we add more mobile nodes into the network. This indicates that the proposed model is scalable with respect to the size of the MANET.

In our literature review, we studied several existing mobile ad hoc network security schemes and IDA schemes with respect to MANET. We also analyzed the security attacks and issues concerning the mobile ad hoc network. Our analysis as described in Chapter 2 shows that the potential threats faced by MANET come in the form of authentication, denial of service, selfish node behavior, or routing attack. We classified the contributions by various authors and the different types of approaches taken to provide security based on the type of security attacks and attempted a comparative study of related requirement features for a secured system. Such classification enhances the understanding of the proposed security schemes in the mobile ad hoc networks.

6.2 Review of Contributions

To recap, the main research contributions are as follows:

- We have proposed an effective Intrusion Detection for MANET through our metric called Threat Index. Threat Index is computed using the fuzzy logic based intrusion detection framework and it detects whether a node is under threat or not.
- Through our response framework in the model we have provided the mechanism for attack control and protection of MANET mobile nodes under threat by identifying the intruder and subjecting appropriate response plan.
- We have also proposed the methodology to identify the significant attack sensitive parameters through machine learning based decision trees concept; we have also proposed the methodology to set their threshold values to differentiate normal, uncertain and vulnerable states.
- Our detection and response framework provides a protocol independent infrastructure for protecting the MANET from active attacks by measuring critical parameters in the underlying MANET infrastructure. Proposed model continuously monitors the online network data and efficiently detects the attacks independent of the protocol used in MANET.

6.3 Recommendations for Future Work

Since not many research efforts have been devoted to MANET IDA, especially the intruder identification, intrusion response and control, this dissertation provides the leading effort in constructing a viable MANET intrusion detection, intrusion response and control model. As a very new, hot and promising research area there are several interesting and important future directions explained as follows.

- Focusing on DSDV and AODV as the routing protocol, and DoS attacks as the threat model, we have designed and developed IDRMAN model to the full. Further work can be performed to extend this model to other passive attacks like unauthorized access, probing, selfishness and non-repudiation attacks in mobile ad hoc networks and active routing attacks.
- Further research could also be devoted to apply the proposed model to secure the integrated wired and wireless networks like cellular networks/sensor networks.
- Having demonstrated the viability of intrusion detection and response approach in providing security for mobile ad hoc networks in a simulation environment, it would be valuable to evaluate the model in a wireless MANET test bed in order to bridge the gap between simulation and actual MANET deployment.

REFERENCES

[1] J. Hubaux, L. Buttyan, and S. Capkun, "The quest for security in mobile ad hoc networks," in *Proceedings of the MobiHoc Conference*, 2001, pp. 146-155.

[2] F. Stajano and R. Anderson, "The resurrecting duckling: Security issues for ad hoc wireless networks," in *Proceedings of International workshop on Security Protocols*, 1999, pp. 172-194.

[3] P. Vinayakray-Jani, "Security within ad hoc networks," presented at First PAMPAS Workshop, London, UK, 2002, pp. 66-67.

[4] K. Wrona, "Distributed security: Ad hoc networks & beyond," presented at First PAMPAS Workshop, London, UK, 2002, pp. 70-71.

[5] L. Buttyan and J. Hubaux, "Report on a working session on security," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 7, no. 1, pp. 74-94, January 2003.

[6] P. Michiardi and R. Molva, "Simulation-based analysis of security exposures in mobile ad hoc networks," in *Proceedings of European Wireless Conference*, 2002, pp. 287-292.

[7] C. G. Harrison, D. M. Chess, and A. Kershenbaum, "Mobile agents: Are they a good idea?," IBM Research Report, 1995.

[8] S. Park, A. Ganz, and Z. Ganz, "Security protocol for IEEE 802.11 wireless area network," *Mobile Networks and Applications*, vol. 3, no. 4, pp. 237-246, September 1998.

[9] F. Stajano, *Security for Ubiquitous Computing*. NY: John Wiley and Sons, 2002.

[10] W. Lee and W. Fan, "Mining system audit data: Opportunities and challenges," *ACM SIGMOD Record,* vol. 30, no. 4, pp. 35-44, December 2001.

[11] S. J. Stolfo et al., "Data mining based intrusion detectors: An overview of the Columbia IDS project," *ACM SIGMOD Record*, vol. 30, no. 4, pp. 5-14, December 2001.

[12] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P. Tan, "Data mining for network intrusion detection," in *Proceedings of NSF Workshop on Next Generation Data Mining*, 2002, pp. 25-36.

[13] C. Warrender, S. Forrest, and B. Pearlmutter, "Detecting intrusions using system calls," in *Proceedings of IEEE symposium on Security and Privacy*, 1999, pp. 133-145.

[14] S. Kumar and E. H. Spafford, "A software architecture to support misuse intrusion detection," in *Proceedings of National Information Security Conference*, 1995, pp. 194-204.

[15] E. Eskin, "Anomaly detection over noisy data using learned probability distributions," in *Proceedings of International Conference on Machine Learning*, 2000, pp. 255-262.

[16] K. IIgun, R. A. Kemmerer, and P. A. Porras, “State transition analysis: A rule based intrusion detection approach,” *IEEE transactions on Software Engineering*, vol. 21, no. 3, pp. 181-199, March 1995.

[17] W. Lee and S. Stolfo, “A framework for constructing features and models for intrusion detection systems,” *ACM Transactions on Information and System Security,* vol. 3. no. 4, pp. 227-261, November 2000.

[18] Y. Zhang, W. Lee, and Y. Huang, “Intrusion detection techniques for mobile wireless networks,” *Wireless Networks*, vol. 9 no. 5, pp. 545-556, September 2003.

[19] H. Deng, Q. Zeng, and D. P. Agrawal, "Network intrusion detection system using random projection technique,” in *Proceedings of the International Conference on Security and Management*, 2003, pp. 10-16.

[20] M. Satyanarayanan, J. J. Kistler, L. B. Mummert, M. R. Ebling, P. Kumar, and Q. Lu, “Experiences with disconnected operation in a mobile environment,” in *Proceedings of USENIX Symposium on Mobile and Location Independent Computing*, 1993, pp. 11–28.

[21] R. Heady, G. Luger, A. Maccabe, and M. Servilla, “The architecture of a network level intrusion detection system,” Computer Science Department, University of New Mexico, Technical Report, 1990.

[22] O. Kachirski and R. Guha, “Intrusion detection using mobile agents in wireless ad hoc networks,” in *Proceedings of IEEE Workshop on Knowledge Media Networking*, 2002, pp. 153-158.

[23] A. Boukerche, “Performance comparison and analysis of ad hoc routing algorithms,” in *Proceedings of IEEE International Conference on Performance, Computing, and Communications*, 2001, pp. 171-178.

[24] J. P. Anderson, *Computer Security Threat Monitoring and Surveillance*. Fort Washington: James P. Anderson Co., 1980.

[25] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222-232, February 1987.

[26] T. Sander and C. Tschudin, “Towards mobile code cryptography,” in *Proceedings of IEEE Symposium on Security and Privacy*, 1998, pp. 215-224.

[27] T. G. Brutch and P. C. Brutch, “Mutual Authentication, Confidentiality, and Key Management (MACKMAN) system for mobile computing and wireless communication,” in *Proceedings of Fourteenth Annual Computer Security Applications Conference*, 1998, pp. 308-317.

[28] B. K. Bhargava, S. B. Kamisetty, and S. K. Madria, “Fault tolerant authentication in mobile computing,” in *Proceedings of International Conference on Internet Computing*, 2000, pp. 395-402.

[29] W. Jansen and T. Karygiannis, “Mobile agent security,” *NIST Special Publication*, 1999.

[30] A. Fugetto, G. P. Pivvo, and G. Vigna, “Understanding code mobility,” *IEEE transactions on software engineering*, vol. 24, no. 5, pp. 342-361, May 1998.

[31] D. Johansen, R. Renessee, and F. B. Schneider, “An introduction to the TACOMA distributed system –version 1.0,” Dept. of Computer Science, Univ. of Tromso and Cornell Univ., Tromso, Norway, Technical Report 95-23, 1995.

[32] B. Askwith, M. Merabti, Q. Shi, and K. Whiteley, "Achieving user privacy in mobile networks," in *Proceedings of Thirteenth Annual Computer Security Applications Conference*, 1997, pp. 108-116.

[33] S. Vemulapalli, M. Halappanavar, and R. Mukkamala, "Security in distributed digital libraries: Issues and challenges," in *Proceedings of Workshop on Distributed Computing Architectures for Digital Libraries*, 2002, pp. 480–486.

[34] E. Ferrari, N. R. Adam, V. Atluri, E. Bertino, and U. Capuozzo, "An authorization system for digital libraries," *VLDB Journal*, vol. 11, no. 1, pp. 58-67, August 2002.

[35] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens, "An on-demand secure routing protocol resilient to byzantine failures," in *Proceedings of ACM Workshop on Wireless Security*, 2002, pp. 21-30.

[36] P. Papadimitratos and Z. Haas, "Secure routing for mobile ad hoc networks," in *Proceedings of SCS Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2002, pp. 27-31.

[37] P. Michiardi and R. Molva, "Prevention of denial of service attacks and selfishness in mobile ad hoc networks," Institute Eurecom, Research Report RR-02-063, 2002.

[38] S. Buchegger and J. Boudec, "Nodes bearing grudges: Towards routing security, fairness, and robustness in mobile ad hoc networks," in *Proceedings of Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, 2002, pp. 403-410.

[39] L. Buttyán and J. P. Hubaux, "Stimulating cooperation in self-organizing mobile ad hoc networks," *ACM journal for Mobile Networks (MONET)*, vol. 8, no. 5, pp. 579-592, October 2003.

[40] S. Capkun, L. Buttyan, and J. P. Hubaux, "Self organized public-key management for mobile ad hoc networks," *Transactions on Mobile Computing*, vol. 2, no. 1, pp. 52-64, January 2003.

[41] M. Steiner, G. Tsudik, and M. Waidner "Diffie-Hellman key distribution extended to group communication," in *Proceedings of ACM Conference on Computer and Communications Security*, 1996, pp. 31-37.

[42] S. Yi and R. Kravets, "Key management for heterogeneous ad hoc wireless networks," Department of Computer Science, University of Illinois, Urbana-Champaign, Technical Report UIUCDCS-R-2002-2290, 2002.

[43] P. Michiardi and R. Molva, "Core: A COllaborative REputation mechanism to enforce node cooperation in mobile ad hoc networks," in *Proceedings of Communication and Multimedia Security Conference*, 2002, pp. 107-121.

[44] S. Buchegger and J. Boudec, "Performance analysis of the CONFIDANT Protocol: Cooperation Of Nodes - Fairness In Distributed Ad hoc NeTworks," in *Proceedings of MobiHoc Conference*, 2002, pp. 226-236.

[45] G. Avoine and S. Vaudenay, "Cryptography with guardian angels: Bringing civilization to Pirates," *ACM Mobile Computing and Communications Review (MC2R)*, vol. 7, no. 1, pp. 74-94, January 2003.

[46] A. A. Ramanujam, J. Bonney, R. Hagelstrom, and K. Thurber, "Techniques for Intrusion-resistant Ad hoc Routing Algorithms (TIARA)," in *Proceedings of MILCOM Conference*, 2000, pp. 660-664.

[47] Y. Hu, D. B. Johnson, and A. Perrig, "SEAD: Secure Efficient Distance vector routing for mobile wireless ad hoc networks," in *Proceedings of Fourth IEEE Workshop on Mobile Computing Systems & Applications*, pp. 3-13, 2002.

[48] J. Brinkley and W. Trost, "Authenticated ad hoc routing at the link layer for mobile systems," *Wireless Networks*, vol. 7, no. 2, pp. 139-145, March 2001.

[49] H. Yang, X. Meng, and S. Lu, "Self-organized network layer security in mobile ad hoc networks," in *Proceedings of ACM MOBICOM Wireless Security Workshop*, 2002, pp. 11-20.

[50] H. Luo and S. Lu, "Ubiquitous and robust authentication services for ad hoc wireless networks," Dept. of Computer Science, UCLA, Technical Report TR-200030, 2000.

[51] Y. Hu, A. Perrig, and D.B. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks," in *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking*, 2002, pp. 12-23.

[52] S. Yi, P. Naldurg, and R. Kravets**, "**Security-aware ad hoc routing for wireless networks," in *Proceedings of Second ACM international symposium on Mobile ad hoc networking & computing*, 2001, pp. 299-302.

[53] S. Marti, T. J. Giuli, K. Lai, and M. Baker, "Mitigating routing misbehavior in mobile ad hoc networks," in *Proceedings of Sixth annual conference on Mobile Computing and Networking*, 2000, pp. 255-265.

[54] Zhou and Haas, "Securing ad hoc networks," *IEEE Network Journal*, vol. 13,no. 6, pp. 24-30, November 1999.

[55] S. P. Alampalayam, A. Kumar, and S. Srinivasan, "Mobile ad hoc networks security – a taxonomy," in *Proceedings of ICACT Conference*, 2005, pp. 839-844.

[56] D. J. Marchette, *Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint.* NY: Springer 2001.

[57] J. Kong, H. Lou, K. Xu, D. Gu, M. Gerla, and S. Lu, "Adaptive security for multi-layer ad hoc networks," *Special Issue of Wireless Communication and Mobile Computing*, vol 2. pp. 533-547, August 2002.

[58] B. Sun, K. Wu, and U. W. Pooch, "Integration of mobility and intrusion detection for wireless ad hoc networks," Accepted for publication in *International Journal of Communication Systems*, 2006.

[59] H. Reiser and G. Vogt, "Security requirements for management systems using mobile agents," in *Proceedings of Fifth IEEE Symposium on Computers and Communications*, 2000, pp. 160-165.

[60] J. E. Canavan, *Fundamentals of Network Security.* Boston: Artech House, 2001.

[61] W. M. Farmer, J. D. Guttman, and V. Swarup, "Security for mobile agents: Issues and requirements," in *Proceedings of Nineteenth National Information Systems Security Conference*, 1996, pp. 591-597.

[62] J. Blaek, "A design of protocol for detecting a mobile agent clone and its correctness proof using colored petrinets," Kwang-Ju Institute of Science and Technology, Korea, Technical Report TR-DIC-CSL-1998-02, 1998.

[63] O. Kachirski and R. Guha, "Effective intrusion detection using multiple sensors in wireless ad hoc networks," in *Proceedings of Thirty Sixth International Conference On System Sciences*, 2003, pp. 57-64.

[64] Y. Huang and W. Lee, "A cooperative intrusion detection system for ad hoc networks," in *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003, pp. 135-147.

[65] P. Albers and O. Camp, "Security in ad hoc networks: A general intrusion detection architecture enhancing trust based approaches," in *Proceedings of First International Workshop on Wireless Information Systems*, 2002, pp. 1-12.

[66] B. Sun, K. Wu, and U. Pooch, "Routing anomaly detection in mobile ad hoc networks," in *Proceedings of Twelfth International conference on computer communications and networks*, 2003, pp. 20-23.

[67] R. Guha, O. Kachirski, D. G. Schwartz, S. Stoecklin, and E. Yilmaz, "Case-based agents for packet-level intrusion detection in ad hoc networks," in *Proceedings of Seventeenth International Symposium on Computer and Information Sciences*, 2002, pp. 315-320.

[68] Y. Huang, W. Fan, W. Lee, and P. S. Yu, "Cross-feature analysis for detecting ad hoc routing anomalies," in *Proceedings of Twenty Third International Conference on Distributed Computing Systems*, 2003, pp.478-487.

[69] C. Tseng and P. Balasubramanyam, "A specification-based intrusion detection system for AODV," in *Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks*, 2003, pp. 125-134.

[70] R. Sowjanya and H. Shah, "Neighborhood Watch: An intrusion detection and response protocol for mobile ad hoc networks," UMBC Technical Report, 2002.

[71] R. Puttini, J. Percher, L. Me, O. Camp, and R. De Souza, "A modular architecture for distributed IDS in MANET structures," *Lecture Notes on Computer Science* vol. 2669, pp.91-113, Springer-Verlag, May 2003.

[72] P. Brutch and C. Ko, "Challenges in intrusion detection for wireless ad hoc networks," in *Proceedings of Symposium on Applications and the Internet Workshop*, 2003, pp. 368-373.

[73] B. Mukherjee, L. Heberlein, and K. Levitt, "Network intrusion detection," *IEEE Network*, vol. 8, no. 3, pp. 26-41, May 1994.

[74] R. Janakiraman, M. Waldvogel, and Q. Zhang, "Indra: A peer-to-peer approach to network intrusion detection and prevention," in *Proceedings of Twelfth IEEE International Workshops*, 2003, pp. 226-231.

[75] H. Debar and A. Wespi, "Aggregation and correlation of intrusion detection alerts," in *Proceedings of Fourth International Symposium on Recent Advances in Intrusion Detection*, 2001, pp. 85-103.

[76] R. Sekar, "Specification-based anomaly detection: A new approach for detecting network intrusions," in *Proceedings of Ninth ACM Conference on Computer and Communications Security*, 2002, pp. 265–274.

[77] Y. Okazaki, I. Sato, and S. Goto, "A new Intrusion detection method based on process profiling," in *Proceedings of Symposium on Applications and the Internet*, 2002, pp. 82-91.

[78] Aurobindo Sundaram, "An introduction to intrusion detection," *Crossroads: The ACM Student Magazine*, vol. 2, no. 4, pp. 3-7, April 1996.

[79] R. Kemmerer, "Computer Security," *Encyclopedia of Software Engineering*, NY: John Wiley and Sons, 1994.

[80] H. S. Teng, K. Chen, and S. C. Lu, "Security audit trail analysis using inductively generated predictive rules," in *Proceedings of Sixth Conf on Artificial Intelligence Applications*, 1990, pp. 24-29.

[81] T. F. Lunt, "A survey of intrusion detection techniques," *Computers and Security*, vol.12, no. 4, pp. 405-418, June 1993.

[82] E. S. Smaha, "Haystack: An intrusion detection system," in *Proceedings of Fourth Aerospace Computer Security Applications Conference*, 1988, pp. 37-44.

[83] F. W. Breyfogle, *Implementing Six Sigma: Smarter Solutions Using Statistical Methods*. NY: John Wiley and Sons, 2003.

[84] S. P. Alampalayam and A. Kumar, "Predictive security model using data mining," in *Proceedings of IEEE GlobeCom Conference*, 2004, pp. 2208-2212.

[85] A. Kumar and R. Ragade, "X-REF: An eXtended Reliability Evaluation Framework for computer systems using fuzzy logic," *Journal of Computer and Software Engineering*, vol. 2, no. 4, pp. 437-458, January 1994.

[86] S. P. Alampalayam and A. Kumar, "Adaptive security model for mobile agents in wireless networks," in *Proceedings of IEEE GlobeCom Conference*, 2003, pp. 1516-1521.

[87] A. Dobra and J. Gehrke, "Bias correction in classification tree construction," in *Proceedings of Eighteenth International Conference on Machine Learning*, 2001, pp. 90-97.

[88] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Pacific Grove: Wadsworth, 1994.

[89] S. P. Alampalayam and A. Kumar, "Security model for routing attacks in wireless networks," in *Proceedings of IEEE VTC Conference*, Fall 2003, pp. 2122-2126.

[90] S. P. Alampalayam and A. Kumar, "An adaptive and predictive security model for mobile ad hoc networks," *Kluwer Personal Communications Journal, Security Special Issue for Next Generation Wireless Networks*, vol. 29, pp. 263-281, June 2004.

[91] D. Karig and R. Lee, "Remote denial of service attacks and countermeasures," Princeton University, Department of Electrical Engineering Technical Report CE-L2001-002, 2001.

[92] Y. Yang, O. F. Rana, C. Georgousopoulos, D. W. Walker, and R. D. Williams, "Mobile agents and the SARA digital library," in *Proceedings of IEEE Advances in Digital Libraries Conference*, 2000, pp. 71-77.

[93] GloMoSim [online] available: http://pcl.cs.ucla.edu/projects/glomosim/, accessed November 2002.

[94] CART software [online] available: http://www.salford-systems.com/, accessed October 2005.

[95] NS2 [online] available: http://www.isi.edu/nsnam/ns, accessed October 2004.

[96] S. Hariri, G. Qu, T. Dharmagadda, M. Ramkishore, and C. S. Raghavendra, "A framework for network vulnerability analysis," *IEEE Security and Privacy*, vol. 1, no. 5, pp. 49-54, September 2003.

[97] C. Sample, Symantec, and I. Poynter, "Quantifying vulnerabilities in the networked environment: Methods and uses," *Jerboa Inc White Paper*, 2000.

[98] D. Simon, "Sum normal optimization of fuzzy membership functions," *International Journal of Uncertainty, Fuzziness, and Knowledge-Based Systems*,

vol. 10, pp. 363-384, August 2002.
[99] E. Natsheh, A. Jantan, S. Khatun, and S.Subramaniam, "Fuzzy reasoning approach for local connectivity management in mobile ad hoc networks," *International Journal of Business Data Communications and Networking,* vol. 2, no. 3, pp.1-18, July 2006.
[100] UCI KDD Archive[online] available: http://kdd.ics.uci.edu, accessed on November 2005.
[101] MATLAB [online] available: www.mathworks.com , accessed on October 2006.
[102] D. Driankov, H. Hellendon, and M. Reinfrank, *An Introduction to Fuzzy Control.* Berlin: Springer-Verlag, 1993.
[103] H. Hosmer, "Security is fuzzy: Applying the fuzzy logic paradigm to the multipolicy paradigm," in *Proceedings of the 1992-1993 Workshop on New Security Paradigms*, 1993, pp.175-184.
[104] J. Luo and S. Bridges, "Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection," *International Journal of Intelligent Systems*, vol. 15, no. 8, pp. 687-703, June 2000.
[105] Viinikka and H. Debar, "Monitoring IDS background noise using EWMA control charts and alert information," in *Proceedings of the Seventh International Symposium on Recent Advances in Intrusion Detection (RAID)*, Vol. 3224 of Lecture Notes in Computer Science. Springer-Verlag, 2004, pp. 166-187.
[106] V. Pappas, B. Zhang, A. Terzis, and L. Zhang, "Fault-tolerant data delivery for multicast overlay networks," in *Proceedings of the Twenty fourth IEEE International Conference on Distributed Computing Systems (ICDCS'04),* 2004, pp. 670-679.
[107] Y. Huang and W. Lee, "Attack analysis and detection for ad hoc routing protocols," in *Proceedings of the Seventh International Symposium on Recent Advances in Intrusion Detection (RAID'04)*, 2004, pp. 125-145.
[108] M. Just, E. Kranakis, and T. Wan, "Resisting malicious packet dropping in wireless ad hoc networks," in *Proceedings of the Second Annual Conference on Ad hoc Networks and Wireless Networks*, 2003, pp.151-163.
[109] D. Liu, P. Ning, and W. Du, "Detecting malicious beacon nodes for secure location discovery in wireless sensor networks," in *Proceedings of the Twenty fifth International Conference on Distributed Computing Systems*, 2005, pp. 609-619.
[110] G. Swamynathan, Ben Y. Zhao, and K. Almeroth, "Decoupling service and feedback trust in a peer-to-peer reputation system," in *Proceedings of the International Workshop on Applications and Economics of Peer-to-Peer Systems (AEPP)*, 2005, pp. 82-90.
[111] O. Ocakoglu, B. Bayoglu, A. Levi, O. Ercetin, and E. Savas. "A probabilistic routing disruption attack on DSR and its analysis," in *Proceedings of the Third Annual Mediterranean Ad Hoc Networking Workshop*, 2004, pp. 300-306.
[112] M. Degroot, *Probability and Statistics*. MA:Addison Wesley, 1986.
[113] D. Wackerly, W. Mendenhall III, and R. L. Scheaffer, *Mathematical Statistics with Applications.* Boston: Duxbury Press, 1995.
[114] Agresti and Franklin, *Statistics: The Art and Science of Learning from Data*. NJ:Prentice Hall, 2006.

[115] R.V. Hogg and J. Ledolter, *Applied Statistics for Engineers and Physical Scientists*. NJ:Prentice Hall, 1991.
[116] V. N. Vapnik, *Statistical Learning Theory*. NY: John Wiley and Sons, 1998.
[117] R. Savola, "Architecture for self-estimation of security level in ad hoc network nodes," in *Proceedings of the Third Australian Information Security Management Conference,* 2005, pp. 88-94.
[118] S. Bhargava and D. P. Agrawal, "Security enhancements in AODV protocol for wireless ad hoc networks," in *Proceedings of IEEE Vehicular Technology Conference*, 2001, pp. 2143-2147.
[119] M. M. Islam, R. Pose, and C. Kopp, "An intrusion detection system for suburban ad-hoc networks," in *Proceedings of IEEE Tencon Conference*, 2005, pp. 41-46.
[120] G. Vigna, S. Gwalani, K. Srinivasan, E. Belding-Royer, and R. Kemmerer, "An intrusion detection tool for AODV-based ad hoc wireless networks," in *Proceedings of the Twentieth ACSA Conference*, 2004, pp. 16-27.
[121] R. Puttini, J. Percher, L. Me, and R. Sousa, "A fully distributed IDS for MANET," in *Proceedings of IEEE Symposium on Computers and Communications*, 2004, pp. 331-338.
[122] B. Lu and U. W. Pooch, "Cooperative security-enforcement routing in mobile ad hoc networks," in *Proceedings of the Fourth IEEE International Conference on Mobile and Wireless Communications Network*, 2002, pp.157–161.
[123] D. Sterne, P. Balasubramanyam, D. Carman, B. Wilson, R. Talpade, C. Ko, R. Balupari, C. Y. Tseng, T. Bowen, K. Levitt, and J. Rowe, "A general cooperative intrusion detection architecture for MANETs," in *Proceedings of the Third IEEE International Workshop on Information Assurance*, 2005, pp. 57-70.
[124] A. Patwardhan, J. Parker, A. Joshi, M. Iorga, and T. Karygiannis, "Secure routing and intrusion detection in ad hoc networks," in *Proceedings of the Third International Conference on Pervasive Computing and Communications*, 2005, pp. 191-199.
[125] N. Stakhanova, S. Basu, and J. Wong, "Taxonomy of intrusion response systems," Computer Science, Iowa State University, Technical Report 06-05, 2006.
[126] V. Ganti, J. E. Gehrke, R. Ramakrishnan, and W.-Y. Loh, "A framework for measuring changes in data characteristics," *Journal of Computer and System Sciences*, vol. 64, no. 3, pp. 542-578, May 2002.
[127] W. Pedrycz, "Why triangular membership functions?," Fuzzy Sets and Systems, vol. 64, no. 1, pp. 21–30, 1994.

APPENDIX A

```
#======================================================================
# * RelatedPerfEval6Node.TCL to  Simulate MANET in NS2.
# * Written by: Sathish Kumar AP.
# * Date: 07/15/2006.
# ======================================================================
# Define options
# ======================================================================
set val(chan)          Channel/WirelessChannel       ;# channel type
set val(prop)          Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)         Phy/WirelessPhy                ;# network interface type
set val(mac)           Mac/802_11                     ;# MAC type
set val(ifq)           Queue/DropTail/PriQueue        ;# interface queue type
set val(ll)            LL                             ;# link layer type
set val(ant)           Antenna/OmniAntenna            ;# antenna model
set val(ifqlen)        50                              ;# max packet in ifq
set val(nn)            6                               ;# number of mobilenodes
set val(rp)            AODV                            ;# routing protocol
#   Node 1 is attacked by Node 2              #
#   Mobility for all nodes is level 1 3 m/s  #
#   ===================================================================
# Main Program
#
# Initialize Global Variables
#
set ns_                [new Simulator]
set tracefd     [open newmanet.tr w]
$ns_ trace-all $tracefd
$ns_ namtrace-all-wireless [open newmanet.nam w] 1000 500
# set up topography object
set topo       [new Topography]
$topo load_flatgrid 1000 500
#
# Create God
#
create-god $val(nn)
#
#  Create the specified number of mobilenodes [$val(nn)] and "attach" them
#  to the channel.
#  Here two nodes are created : node(0) and node(1)
# configure node
       $ns_ node-config -adhocRouting $val(rp) \
                                -llType $val(ll) \
                                -macType $val(mac) \
                                -ifqType $val(ifq) \
                                -ifqLen $val(ifqlen) \
                                -antType $val(ant) \
                                -propType $val(prop) \
                                -phyType $val(netif) \
```

```
                        -channelType $val(chan) \
                        -topoInstance $topo \
                        -agentTrace ON \
                        -routerTrace ON \
                        -macTrace ON \
                        -movementTrace OFF

        for {set i 0} {$i < $val(nn) } {incr i} {
                set node_($i) [$ns_ node]
                $node_($i) random-motion 0              ;# disable random motion
        }

#
# Provide initial (X,Y, for now Z=0) co-ordinates for mobilenodes
#
$node_(0) set X_ 5.0
$node_(0) set Y_ 2.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 39.0
$node_(1) set Y_ 38.0
$node_(1) set Z_ 0.0
$node_(2) set X_ 12.0
$node_(2) set Y_ 8.0
$node_(2) set Z_ 6.0
$node_(3) set X_ 15.0
$node_(3) set Y_ 23.0
$node_(3) set Z_ 6.0
$node_(4) set X_ 22.0
$node_(4) set Y_ 28.0
$node_(4) set Z_ 6.0
$node_(5) set X_ 32.0
$node_(5) set Y_ 38.0
$node_(5) set Z_ 6.0
# Now produce some simple node movements
# Node_(1) starts to move towards node_(0)
#
# Now produce some simple node movements
# Node_(1) starts to move towards node_(0)
#
$ns_ at 50.0 "$node_(1) setdest 25.0 20.0 12.0"
$ns_ at 10.0 "$node_(0) setdest 20.0 18.0 12.0"
$ns_ at 50.0 "$node_(2) setdest 36.5 17.5 12.0"
# Node_(1) then starts to move away from node_(0)
$ns_ at 100.0 "$node_(1) setdest 490.0 480.0 12.0"
$ns_ at 70.0 "$node_(3) setdest 190.0 480.0 12.0"
$ns_ at 80.0 "$node_(4) setdest 290.0 480.0 12.0"
$ns_ at 90.0 "$node_(5) setdest 390.0 480.0 12.0"
# Setup traffic flow between nodes
# TCP connections between node_(0) and node_(1)

set udp0 [new Agent/UDP]
$udp0 set class_ 1
set udp2 [new Agent/UDP]
$udp2 set class_ 2
#SET A TCP Connection between node_(3) and node_(4)
set tcp3 [new Agent/TCP/Newreno]
```

```
$tcp3 set class_ 3
set tcpsink [new Agent/TCPSink]
$tcpsink set class_ 4
set sink [new Agent/UDP]
set sink2 [new Agent/UDP]

$ns_ attach-agent $node_(0) $udp0
$ns_ attach-agent $node_(1) $sink
$ns_ attach-agent $node_(2) $udp2
$ns_ attach-agent $node_(3) $tcp3
$ns_ attach-agent $node_(4) $tcpsink
$ns_ attach-agent $node_(5) $tcp3

$ns_ connect $udp0 $sink
$ns_ connect $udp2 $sink2
$ns_ connect $udp10 $sink11
$ns_ connect $tcp3 $tcpsink
$ns_ connect $udp20 $sink21
set cbr0 [new Application/Traffic/CBR]
$cbr0 set packetSize_ 512
$cbr0 set interval_ .01
set ftp [new Application/FTP]
$ftp attach-agent $tcp3
$ftp set packetSize_ 512
$ftp set interval_ 1
$cbr0 attach-agent $udp0
$ns_ at 10.0 "$cbr0 start"
$ns_ at 10.0 "$ftp start"
#$ns_ at 10.0 "$cbr4 start"
#set qmon [$ns_ monitor-queue $node_(0) $node_(1) [open qmadhoc.out w] 0.1];
for {set i 0} {$i < [expr ($val(nn))]} {incr i} {
   # 20 defines the node size in nam, must adjust it according to your
   # scenario
   # The function must be called after mobility model is defined
   $ns_ initial_node_pos $node_($i) 10
}
# Tell nodes when the simulation ends
for {set i 0} {$i < $val(nn) } {incr i} {
   $ns_ at 150.0 "$node_($i) reset";
}
$ns_ at 1000.0 "$ns_ nam-end-wireless 100.0"
#$ns_ at 200.0 "$ns_ nam-end-wireless 100.0"
$ns_ at 1000.0 "$ns_ halt"
#$ns_ at 200.0 "$ns_ halt"
$ns_ at 1500.0 "stop"
#$ns_ at 300.0 "stop"
#$ns_ at 1500.01 "puts \"NS EXITING...\" ; $ns_ halt"
$ns_ at 300.01 "puts \"NS EXITING...\" ; $ns_ halt"
proc stop {} {
   global ns_ tracefd
   $ns_ flush-trace
   close $tracefd
}
puts "Starting Simulation..."
$ns_ run
```

APPENDIX B

```
/*********************************************************************
 * MAIN.cpp --- C++ code to Test TI detection framework.
 * (This file is main file)
 * Written by: Sathish Kumar AP.
 * Date: 07/15/2006.
 *********************************************************************/
#include <stdio.h>
#include "Mainfis3In.cpp"
#include "Fuzzy3Input.cpp"
#include <fstream.h>
#include <sstream>
#include <string.h>
#define MAXLINE 100
//using namespace std;
using std::string;
void main( void )
{
        float EC;   float PD;  float QL;
        double TI; string line = "";
        ifstream OpenFile("sathishinp.txt");
        printf("Reading Input File\n");
        char str1[100]; char s1[10] = ""; char s2[10] = ""; char s3[10] = "";
        int u =0;int v =0;int w =0;
        while(!OpenFile.eof())
                {
                OpenFile.getline(str1,100);
                for(int x = 0; x < 8; x++)
                {       s1[u] = str1[x]; u++;}
                u=0;
                for(int y = 9; y < 17; y++)
                {       s2[v] = str1[y]; v++;}
                v=0;
                for(int z = 18; z < 26; z++)
                {       s3[w] = str1[z]; w++;}
                w=0;
                printf("%s\n",str1); printf("%s\n",s1); printf("%s\n",s2);printf("%s\n",s3);
                EC = atof (s1);     PD = atof (s2);
                QL = atof (s3);
                printf ("EC is %f\n", EC);
                printf ("PD is %f\n", PD);
                printf ("QL is %f\n", QL);
                TI = Fuzzy_Inference("Threshold_Index.fis", EC, PD, QL);
                printf("TI = %f \n", TI);
                }
}
```

```
/*********************************************************************
 * Fuzzy3Input.cpp --- C++ code for TI fuzzy inference system.
 * Written by: Sathish Kumar A P
 * Date: 07/02/2006.
 *********************************************************************/
double Fuzzy_Inference(char *fis_file, float input1, float input2, float input3)
{
        char *data_file;
        double  **Outputmatrix;
        double fis_output;

                        FILE *Inptfile;
                        Inptfile = fopen("Inpt.txt", "w");
                        fprintf(Inptfile,"%.2f\t%.2f\t%.2f\n",input1, input2, input3);
                        fclose(Inptfile);
                        data_file = "Inpt.txt";
                        Outputmatrix = fis_system(data_file, fis_file);
                        fis_output = Outputmatrix[0][0];
                        FILE *Otptfile;
                        Otptfile = fopen("fuz_out.txt", "a");
                        fprintf(Otptfile,"%.2f\n",fis_output);
                        fclose(Otptfile);
        return(fis_output);
}
/*********************************************************************
 * MainFIS3In.cpp --- C++ code for TI fuzzy inference system.
 *********************************************************************/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include <conio.h>
#include "Macros.cpp"
#include "MF.cpp"
#include "DataMatrix3In.cpp"
#include "FISMatrix.cpp"
#include "Defuzz.cpp"
#include "BuildFisNode.cpp"
#include "Inference.cpp"
double ** fis_system(char *data_file, char *fis_file)
{
        FIS *fis;
        int i;
        int debug = 0;
        double **dataMatrix, **fisMatrix, **outputMatrix;
        int data_row_n, data_col_n, fis_row_n, fis_col_n;
        /* obtain data matrix and FIS matrix */
        dataMatrix = returnDataMatrix(data_file, &data_row_n, &data_col_n);
        fisMatrix = returnFismatrix(fis_file, &fis_row_n, &fis_col_n);
        /* build FIS data structure */
        fis = (FIS *)fisCalloc(1, sizeof(FIS));
        fisBuildFisNode(fis, fisMatrix, fis_col_n, MF_POINT_N);
        /* error checking  */
        if (data_col_n < fis->in_n) {
                printf("Given FIS is a %d-input %d-output system.\n",
                fis->in_n, fis->out_n);
```

```
            printf("Given data file does not have enough input entries.\n");
            fisFreeMatrix((void **)dataMatrix, data_row_n);
            fisFreeMatrix((void **)fisMatrix, fis_row_n);
            fisFreeFisNode(fis);
            fisError("Exiting ...");
}
/* debugging  */
if (debug)
            fisPrintData(fis);
/* create output matrix */
outputMatrix = (double **)fisCreateMatrix(data_row_n, fis->out_n, sizeof(double));
/* evaluate FIS on each input vector */
for (i = 0; i < data_row_n; i++)
            getFisOutput(dataMatrix[i], fis, outputMatrix[i]);
/* print output vector
for (i = 0; i < data_row_n; i++) {
            for (j = 0; j < fis->out_n; j++)
                        printf("outputMatrix = %.3f ", outputMatrix[i][j]);
            printf("\n");
}*//* clean up memory */
fisFreeFisNode(fis);
fisFreeMatrix((void **)dataMatrix, data_row_n);
fisFreeMatrix((void **)fisMatrix, fis_row_n);
return(outputMatrix);}
```

APPENDIX C

```
%*****************************************************************
% Matlab code to optimize Membership Functions of TI detection framework.
      % Written by: Sathish Kumar AP.
      % Date: 07/10/2006.
%*****************************************************************
       clear
       Target = 7;
       decrease_ratio = 0.97;
       increase_ratio = 1.03;
       ec_ns = 2.1;    ec_us = 2.15;         ec_vs = 2.2;
       pd_ns = 109;   pd_us = 117;         pd_vs = 126;
       ql_ns = 122;    ql_us = 134;         ql_vs = 146;
       exp_path     = 'C:\adHocSim\Generate_fis_file\Threshold_Index\Attaked_Data_3';
       results_file = 'C:\adHocSim\Generate_fis_file\Threshold_Index\Attaked_Data_3\results.txt';
       save_file    = 'C:\adHocSim\Generate_fis_file\Threshold_Index\Attaked_Data_3\MFs_points';
       addpath(exp_path);
       load pd.txt
       load ql.txt
       load ec.txt
       no_of_training_data = size(ec,1);
       fid = fopen(results_file,'a');
       for i = 1:no_of_training_data
          input_array = [ec(i) pd(i) ql(i)];
          output_TI = Fuzzy_TI_fun(input_array, ec_ns, ec_us, ec_vs, pd_ns, pd_us, pd_vs, ql_ns, ql_us,
ql_vs);
          fprintf(fid,'%.1f\n',output_TI);
          % Test NS netwotk %%%%%%%%%%%%%%%%%%%%
          if((Target <=4) & (output_TI > 4))
             if ((ec(i) > (ec_ns-(ec_ns*0.2))) & (ec(i) < (ec_vs-(ec_vs*0.2))))
                if ((ec_ns * increase_ratio) < ec_us)
                   ec_ns = ec_ns * increase_ratio;
                end
             end
             if ((pd(i) > (pd_ns-(pd_ns*0.2))) & (pd(i) < (pd_vs-(pd_vs*0.2))))
                if ((pd_ns * increase_ratio) < pd_us)
                   pd_ns = pd_ns * increase_ratio;
                end
             end
             if ((ql(i) > (ql_ns-(ql_ns*0.2))) & (ql(i) < (ql_vs-(ql_vs*0.2))))
                if ((ql_ns * increase_ratio) < ql_us)
                   ql_ns = ql_ns * increase_ratio;
                end
             end
          end
          % Test VS netwotk %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
          if((Target >=7) & (output_TI < 7))
             if ((ec(i) > (ec_ns-(ec_ns*0.2))) & (ec(i) < (ec_vs-(ec_vs*0.2))))
```

```
        if ((ec_vs * decrease_ratio) > ec_us)
            ec_vs = ec_vs * decrease_ratio;
        end
    end
    if ((pd(i) > (pd_ns-(pd_ns*0.2))) & (pd(i) < (pd_vs-(pd_vs*0.2))))
        if ((pd_vs * decrease_ratio) > pd_us)
            pd_vs = pd_vs * decrease_ratio;
        end
    end
    if ((ql(i) > (ql_ns-(ql_ns*0.2))) & (ql(i) < (ql_vs-(ql_vs*0.2))))
        if ((ql_vs * decrease_ratio) > ql_us)
            ql_vs = ql_vs * decrease_ratio;
        end
    end
  end
  %% Test US netwotk %%%%%%%%%%%%%%%%%%%%%%%%%%%
  if((Target > 4) & (Target < 7) & (output_TI < 4))
    if ((ec(i) > (ec_ns-(ec_ns*0.2))) & (ec(i) < (ec_vs-(ec_vs*0.2))))
        if ((ec_us * increase_ratio) < ec_vs)
            ec_us = ec_us * increase_ratio;
        end
    end
    if ((pd(i) > (pd_ns-(pd_ns*0.2))) & (pd(i) < (pd_vs-(pd_vs*0.2))))
        if ((pd_us * increase_ratio) < pd_vs)
            pd_us = pd_us * increase_ratio;
        end
    end
    if ((ql(i) > (ql_ns-(ql_ns*0.2))) & (ql(i) < (ql_vs-(ql_vs*0.2))))
        if ((ql_us * increase_ratio) < ql_vs)
            ql_us = ql_us * increase_ratio;
        end
    end
  end
  if((Target > 4) & (Target < 7) & (output_TI > 7))
    if ((ec(i) > (ec_ns-(ec_ns*0.2))) & (ec(i) < (ec_vs-(ec_vs*0.2))))
        if ((ec_us * decrease_ratio) > ec_ns)
            ec_us = ec_us * decrease_ratio;
        end
    end
    if ((pd(i) > (pd_ns-(pd_ns*0.2))) & (pd(i) < (pd_vs-(pd_vs*0.2))))
        if ((pd_us * decrease_ratio) > pd_ns)
            pd_us = pd_us * decrease_ratio;
        end
    end
    if ((ql(i) > (ql_ns-(ql_ns*0.2))) & (ql(i) < (ql_vs-(ql_vs*0.2))))
        if ((ql_us * decrease_ratio) > ql_ns)     ql_us = ql_us * decrease_ratio;
        end
    end
  end
  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
end % for loop
fclose(fid);
save (save_file, 'ec_ns', 'ec_us', 'ec_vs', 'pd_ns', 'pd_us', 'pd_vs', 'ql_ns', 'ql_us', 'ql_vs')
rmpath(exp_path).
```

```
%**********************************************************************
      % Matlab code  to implement Threshold Training Algorithm.
      % Written by: Sathish Kumar AP        % Date: 07/15/2006.
%**********************************************************************
  ec_ns = 2.1;
  ec_us = 2.15;
  ec_vs = 2.2;
  pd_ns = 109;
  pd_us = 117;
  pd_vs = 126;
  ql_ns = 122;
  ql_us = 134;
  ql_vs = 146;
%%%%In the above matrix w are interested in the non shaded region only where P2 > P1
for i = 1: 10
          for j = 1: 10
                    p[i] [j] = 0;
          end % end for loop
end       % end for loop
exp_path   = 'C:\adHocSim\Generate_fis_file\Threshold_Index\Attaked_Data_3';
addpath(exp_path);
load pd.txt
load ql.txt
load ec.txt
%%%%%no_of_training_data = size(ec,1);
 for k = 1:159 %no_of_training_data
          for i = 1: 10
                    for j=i+1: 10
                              p1 = i;
                              p2 = j;
                              output_TI = Fuzzy_TI_fun(input_array, ec_ns, ec_us, ec_vs, pd_ns,
                              pd_us,pd_vs, ql_ns, ql_us, ql_vs, p1, p2, (p1+p2)/2);
                              if(output_TI > p2)
                                        CalculatedState = 'VS';
                              end;
                              if(output_TI < p1)
                                        CalculatedState = 'NS';
                              end;
                              if(output_TI >= p1) and (output_TI <= p2)
                                        CalculatedState = 'US';
                              end
                              if (CalculatedState = = Target Outcome Label)
                                                  p[i][j] = p[i][j] + 1;
                              end
                    end %%%%end first for loop
          end %%% end second for loop
end %%% end third for loop
p1=0; p2=0; MaxValue = 0;
for i = 1 : 10
          for j = 1: 10
                    If (p[i][j] > MaxValue)
                              MaxValue = p[i][j]; p1=i;  p2=j;
                    end
          end % end for loop
end % end for loop
printf("P1 and P2 are '%d %d\n", p1, p2);
```

```
%**********************************************************************
    % Matlab code  to generate FIS file.
    % Written by: Sathish Kumar AP        % Date: 07/15/2006.
%**********************************************************************
        clear
        a=newfis('FuzzyTI');
        a=addvar(a,'input','EC',[0 4]);
        a=addmf(a,'input',1,'NS','zmf',[2.1 2.15]);
        a=addmf(a,'input',1,'US','trimf',[2.1 2.15 2.2]);
        a=addmf(a,'input',1,'VS','smf',[2.15 2.2]);
        a = rmmf(a,'input',1,'mf',1);
        a = rmmf(a,'input',1,'mf',1);
        a = rmmf(a,'input',1,'mf',1);
        figure('name', 'input 1: EC', 'numbertitle', 'off');
        plotmf(a,'input',1);
%-------------------
        a=addvar(a,'input','PD',[0 400]);
        a=addmf(a,'input',2,'NS','zmf',[109 117]);
        a=addmf(a,'input',2,'US','trimf',[109 117 126]);
        a=addmf(a,'input',2,'VS','smf',[117 126]);
        a = rmmf(a,'input',2,'mf',1);
        a = rmmf(a,'input',2,'mf',1);
        a = rmmf(a,'input',2,'mf',1);
        figure('name', 'input 2: PD', 'numbertitle', 'off');
        plotmf(a,'input',2);
%-------------------
        a=addvar(a,'input','QL',[0 400]);
        a=addmf(a,'input',3,'NS','zmf',[122 134]);
        a=addmf(a,'input',3,'US','trimf',[122 134 146]);
        a=addmf(a,'input',3,'VS','smf',[134 146]);
        a = rmmf(a,'input',3,'mf',1);
        a = rmmf(a,'input',3,'mf',1);
        a = rmmf(a,'input',3,'mf',1);
        figure('name', 'input 3: QL', 'numbertitle', 'off');
        plotmf(a,'input',3);
%-------------------
        a=addvar(a,'output','TI',[1 10]);
        a=addmf(a,'output',1,'NS','zmf',[4 5.5]);
        a=addmf(a,'output',1,'US','trimf',[4 5.5 7]);
        a=addmf(a,'output',1,'VS','smf',[5.5 7]);
        a = rmmf(a,'output',1,'mf',1);
        a = rmmf(a,'output',1,'mf',1);
        a = rmmf(a,'output',1,'mf',1);
        figure('name', 'output: TI', 'numbertitle', 'off');
        plotmf(a,'output',1);
%-------------------
        rule01='if EC is NS and PD is NS and QL is NS then TI is NS';
        rule02='if EC is NS and PD is NS and QL is US then TI is NS';
        rule03='if EC is NS and PD is NS and QL is VS then TI is NS';
        rule04='if EC is NS and PD is US and QL is NS then TI is NS';
        rule05='if EC is NS and PD is US and QL is US then TI is NS';
        rule06='if EC is NS and PD is US and QL is VS then TI is US';
        rule07='if EC is NS and PD is VS and QL is NS then TI is NS';
        rule08='if EC is NS and PD is VS and QL is US then TI is US';
        rule09='if EC is NS and PD is VS and QL is VS then TI is VS';
```

```
rule10='if EC is US and PD is NS and QL is NS then TI is NS';
rule11='if EC is US and PD is NS and QL is US then TI is NS';
rule12='if EC is US and PD is NS and QL is VS then TI is US';
rule13='if EC is US and PD is US and QL is NS then TI is NS';
rule14='if EC is US and PD is US and QL is US then TI is US';
rule15='if EC is US and PD is US and QL is VS then TI is VS';
rule16='if EC is US and PD is VS and QL is NS then TI is US';
rule17='if EC is US and PD is VS and QL is US then TI is VS';
rule18='if EC is US and PD is VS and QL is VS then TI is VS';
rule19='if EC is VS and PD is NS and QL is NS then TI is NS';
rule20='if EC is VS and PD is NS and QL is US then TI is US';
rule21='if EC is VS and PD is NS and QL is VS then TI is VS';
rule22='if EC is VS and PD is US and QL is NS then TI is US';
rule23='if EC is VS and PD is US and QL is US then TI is VS';
rule24='if EC is VS and PD is US and QL is VS then TI is VS';
rule25='if EC is VS and PD is VS and QL is NS then TI is VS';
rule26='if EC is VS and PD is VS and QL is US then TI is VS';
rule27='if EC is VS and PD is VS and QL is VS then TI is VS';

ruleList = [rule01; rule02; rule03; rule04; rule05; rule06; rule07; rule08; rule09;
        rule10; rule11; rule12; rule13; rule14; rule15; rule16; rule17; rule18;
  rule19; rule20; rule21; rule22; rule23; rule24; rule25; rule26; rule27 ];
a=parsrule(a,ruleList);
figure('name', 'fis', 'numbertitle', 'off');
plotfis(a);
showfis(a);
ruleview(a);
writefis(a,'C:\Threshold_Index')
```

APPENDIX D

Queue length Parameter NS2 log at node 2 without response

qlen: Second at Time 10.000000 Src 2 Dest -2
qlen: Second at Time 10.010000 Src 2 Dest -2
qlen: Second at Time 137.730000 Src 2 Dest -2
qlen: Second at Time 141.690000 Src 2 Dest -2
qlen: Second at Time 141.700000 Src 2 Dest -2
qlen: Second at Time 141.710000 Src 2 Dest -2
qlen: Second at Time 141.720000 Src 2 Dest -2
qlen: Second at Time 141.730000 Src 2 Dest -2
qlen: Second at Time 141.740000 Src 2 Dest -2
qlen: Second at Time 141.750000 Src 2 Dest -2
qlen: Second at Time 141.760000 Src 2 Dest -2
qlen: Second at Time 141.770000 Src 2 Dest -2
qlen: Second at Time 141.780000 Src 2 Dest -2
qlen: Second at Time 141.790000 Src 2 Dest -2
qlen: Second at Time 141.800000 Src 2 Dest -2
qlen: Second at Time 141.810000 Src 2 Dest -2
qlen: Second at Time 141.820000 Src 2 Dest -2
qlen: Second at Time 141.830000 Src 2 Dest -2
qlen: Second at Time 141.840000 Src 2 Dest -2
qlen: Second at Time 141.850000 Src 2 Dest -2
qlen: Second at Time 141.860000 Src 2 Dest -2
qlen: Second at Time 141.870000 Src 2 Dest -2
qlen: Second at Time 141.880000 Src 2 Dest -2
qlen: Second at Time 141.890000 Src 2 Dest –
qlen: Second at Time 175.000000 Src 2 Dest -2
qlen: Second at Time 175.010000 Src 2 Dest -2
qlen: Second at Time 175.020000 Src 2 Dest -2
qlen: Second at Time 175.030000 Src 2 Dest -2
qlen: Second at Time 175.040000 Src 2 Dest -2
qlen: Second at Time 175.050000 Src 2 Dest -2
qlen: Second at Time 175.060000 Src 2 Dest -2
qlen: Second at Time 175.070000 Src 2 Dest -2
qlen: Second at Time 175.080000 Src 2 Dest -2
qlen: Second at Time 175.090000 Src 2 Dest -2
qlen: Second at Time 175.100000 Src 2 Dest -2
qlen: Second at Time 175.110000 Src 2 Dest -2
qlen: Second at Time 175.120000 Src 2 Dest -2
qlen: Second at Time 175.130000 Src 2 Dest -2
qlen: Second at Time 175.140000 Src 2 Dest -2
qlen: Second at Time 175.150000 Src 2 Dest -2
qlen: Second at Time 175.160000 Src 2 Dest -2
qlen: Second at Time 175.170000 Src 2 Dest -2
qlen: Second at Time 175.180000 Src 2 Dest -2
qlen: Second at Time 175.190000 Src 2 Dest -2
qlen: Second at Time 175.200000 Src 2 Dest -2
qlen: Second at Time 175.210000 Src 2 Dest -2
qlen: Second at Time 175.220000 Src 2 Dest -2

qlen: Second at Time 175.230000 Src 2 Dest -2
qlen: Second at Time 175.240000 Src 2 Dest -2
qlen: Second at Time 175.250000 Src 2 Dest -2
qlen: Second at Time 175.260000 Src 2 Dest -2
qlen: Second at Time 175.270000 Src 2 Dest -2
qlen: Second at Time 175.280000 Src 2 Dest -2
qlen: Second at Time 175.290000 Src 2 Dest -2
qlen: Second at Time 175.300000 Src 2 Dest -2
qlen: Second at Time 175.310000 Src 2 Dest -2
qlen: Second at Time 175.320000 Src 2 Dest -2
qlen: Second at Time 175.330000 Src 2 Dest -2
qlen: Second at Time 175.340000 Src 2 Dest -2
qlen: Second at Time 999.850000 Src 2 Dest -2
qlen: Second at Time 999.860000 Src 2 Dest -2
qlen: Second at Time 999.870000 Src 2 Dest -2
qlen: Second at Time 999.880000 Src 2 Dest -2
qlen: Second at Time 999.890000 Src 2 Dest -2
qlen: Second at Time 999.900000 Src 2 Dest -2
qlen: Second at Time 999.910000 Src 2 Dest -2
qlen: Second at Time 999.920000 Src 2 Dest -2
qlen: Second at Time 999.930000 Src 2 Dest -2
qlen: Second at Time 999.940000 Src 2 Dest -2
qlen: Second at Time 999.950000 Src 2 Dest -2
qlen: Second at Time 999.960000 Src 2 Dest -2
qlen: Second at Time 999.970000 Src 2 Dest -2
qlen: Second at Time 999.980000 Src 2 Dest -2
qlen: Second at Time 999.990000 Src 2 Dest -2
qlen: Second at Time 1000.000000 Src 2 Dest -2

Queulength Parameter NS2 log at Node 2 With Response:

qlen: Second at Time 10.000000 Src 2 Dest -2
qlen: Second at Time 10.010000 Src 2 Dest -2
qlen: Second at Time 137.730000 Src 2 Dest -2
qlen: Second at Time 141.690000 Src 2 Dest -2
qlen: Second at Time 141.700000 Src 2 Dest -2
qlen: Second at Time 141.710000 Src 2 Dest -2
qlen: Second at Time 141.720000 Src 2 Dest -2
qlen: Second at Time 141.730000 Src 2 Dest -2
qlen: Second at Time 141.740000 Src 2 Dest -2
qlen: Second at Time 141.750000 Src 2 Dest -2
qlen: Second at Time 141.760000 Src 2 Dest -2
qlen: Second at Time 141.770000 Src 2 Dest -2
qlen: Second at Time 141.780000 Src 2 Dest -2
qlen: Second at Time 141.790000 Src 2 Dest -2
qlen: Second at Time 141.800000 Src 2 Dest -2
qlen: Second at Time 141.810000 Src 2 Dest -2
qlen: Second at Time 141.820000 Src 2 Dest -2
qlen: Second at Time 141.830000 Src 2 Dest -2
qlen: Second at Time 141.840000 Src 2 Dest -2
qlen: Second at Time 141.850000 Src 2 Dest -2
qlen: Second at Time 141.860000 Src 2 Dest -2
qlen: Second at Time 141.870000 Src 2 Dest -2
qlen: Second at Time 141.880000 Src 2 Dest -2
qlen: Second at Time 141.890000 Src 2 Dest -2
qlen: Second at Time 141.900000 Src 2 Dest -2
qlen: Second at Time 141.910000 Src 2 Dest -2

qlen: Second at Time 141.920000 Src 2 Dest -2
qlen: Second at Time 141.930000 Src 2 Dest -2
qlen: Second at Time 141.940000 Src 2 Dest -2
qlen: Second at Time 141.950000 Src 2 Dest -2
qlen: Second at Time 141.960000 Src 2 Dest -2
qlen: Second at Time 141.970000 Src 2 Dest -2
qlen: Second at Time 141.980000 Src 2 Dest -2
qlen: Second at Time 141.990000 Src 2 Dest -2
qlen: Second at Time 142.000000 Src 2 Dest -2
qlen: Second at Time 142.010000 Src 2 Dest -2
qlen: Second at Time 142.020000 Src 2 Dest -2
qlen: Second at Time 142.030000 Src 2 Dest -2
qlen: Second at Time 142.040000 Src 2 Dest -2
qlen: Second at Time 151.210000 Src 2 Dest -2
qlen: Second at Time 151.220000 Src 2 Dest -2
qlen: Second at Time 151.230000 Src 2 Dest -2
qlen: Second at Time 151.240000 Src 2 Dest -2
qlen: Second at Time 151.250000 Src 2 Dest -2
qlen: Second at Time 151.260000 Src 2 Dest -2
qlen: Second at Time 151.270000 Src 2 Dest -2
qlen: Second at Time 151.280000 Src 2 Dest -2
qlen: Second at Time 151.290000 Src 2 Dest -2
qlen: Second at Time 151.300000 Src 2 Dest -2
qlen: Second at Time 151.310000 Src 2 Dest -2
qlen: Second at Time 151.320000 Src 2 Dest -2
qlen: Second at Time 151.330000 Src 2 Dest -2
qlen: Second at Time 151.340000 Src 2 Dest -2
qlen: Second at Time 151.350000 Src 2 Dest -2
qlen: Second at Time 151.360000 Src 2 Dest -2
qlen: Second at Time 151.370000 Src 2 Dest -2
qlen: Second at Time 151.380000 Src 2 Dest -2
qlen: Second at Time 151.390000 Src 2 Dest -2
qlen: Second at Time 151.400000 Src 2 Dest -2
qlen: Second at Time 151.410000 Src 2 Dest -2
qlen: Second at Time 151.420000 Src 2 Dest -2
qlen: Second at Time 151.430000 Src 2 Dest -2
qlen: Second at Time 151.440000 Src 2 Dest -2
qlen: Second at Time 151.450000 Src 2 Dest -2
qlen: Second at Time 151.460000 Src 2 Dest -2
qlen: Second at Time 151.470000 Src 2 Dest -2
qlen: Second at Time 151.480000 Src 2 Dest -2
qlen: Second at Time 151.490000 Src 2 Dest -2
qlen: Second at Time 151.500000 Src 2 Dest -2
qlen: Second at Time 151.510000 Src 2 Dest -2
qlen: Second at Time 151.520000 Src 2 Dest -2
qlen: Second at Time 151.530000 Src 2 Dest -2
qlen: Second at Time 151.540000 Src 2 Dest -2
qlen: Second at Time 151.550000 Src 2 Dest -2
qlen: Second at Time 151.560000 Src 2 Dest -2
qlen: Second at Time 151.570000 Src 2 Dest -2
qlen: Second at Time 151.580000 Src 2 Dest -2
qlen: Second at Time 151.590000 Src 2 Dest -2
qlen: Second at Time 151.600000 Src 2 Dest -2
qlen: Second at Time 151.610000 Src 2 Dest -2
qlen: Second at Time 151.620000 Src 2 Dest -2
qlen: Second at Time 151.630000 Src 2 Dest -2

qlen: Second at Time 151.640000 Src 2 Dest -2
qlen: Second at Time 151.650000 Src 2 Dest -2
qlen: Second at Time 151.660000 Src 2 Dest -2

Packet Drop Parameter NS2 log at Node 2 Without Response:

enquedrop: Time 10.000000 Src 2 Dest -2
enquedrop: Time 10.010000 Src 2 Dest -2
drop: at time 137.729665 Src 2 Dest 1
drop: at time 137.729665 Src 2 Dest 1
drop: at time 137.729665 Src 2 Dest 1
drop: at time 137.729665 Src 2 Dest 1
enquedrop: Time 137.730000 Src 2 Dest -2
drop: at time 141.681306 Src 2 Dest 1
drop: at time 141.681306 Src 2 Dest 1
drop: at time 141.681306 Src 2 Dest 1
enquedrop: Time 141.690000 Src 2 Dest -2
enquedrop: Time 141.700000 Src 2 Dest -2
enquedrop: Time 141.710000 Src 2 Dest -2
enquedrop: Time 141.720000 Src 2 Dest -2
enquedrop: Time 141.730000 Src 2 Dest -2
enquedrop: Time 141.740000 Src 2 Dest -2
enquedrop: Time 141.750000 Src 2 Dest -2
enquedrop: Time 141.760000 Src 2 Dest -2
enquedrop: Time 141.770000 Src 2 Dest -2
enquedrop: Time 141.780000 Src 2 Dest -2
enquedrop: Time 141.790000 Src 2 Dest -2
enquedrop: Time 141.800000 Src 2 Dest -2
enquedrop: Time 175.890000 Src 2 Dest -2
enquedrop: Time 175.900000 Src 2 Dest -2
enquedrop: Time 175.910000 Src 2 Dest -2
enquedrop: Time 175.920000 Src 2 Dest -2
enquedrop: Time 175.930000 Src 2 Dest -2
enquedrop: Time 175.940000 Src 2 Dest -2
enquedrop: Time 175.950000 Src 2 Dest -2
enquedrop: Time 175.960000 Src 2 Dest -2
enquedrop: Time 175.970000 Src 2 Dest -2
enquedrop: Time 175.980000 Src 2 Dest -2
enquedrop: Time 175.990000 Src 2 Dest -2
enquedrop: Time 176.000000 Src 2 Dest -2
enquedrop: Time 176.010000 Src 2 Dest -2
enquedrop: Time 176.020000 Src 2 Dest -2
enquedrop: Time 176.030000 Src 2 Dest -2
enquedrop: Time 176.040000 Src 2 Dest -2
enquedrop: Time 176.050000 Src 2 Dest -2
enquedrop: Time 176.060000 Src 2 Dest -2
enquedrop: Time 176.070000 Src 2 Dest -2
enquedrop: Time 176.080000 Src 2 Dest -2
enquedrop: Time 176.090000 Src 2 Dest -2
enquedrop: Time 176.100000 Src 2 Dest -2
enquedrop: Time 176.110000 Src 2 Dest -2
enquedrop: Time 176.120000 Src 2 Dest -2
enquedrop: Time 176.130000 Src 2 Dest -2
enquedrop: Time 176.140000 Src 2 Dest -2
enquedrop: Time 176.150000 Src 2 Dest -2
enquedrop: Time 176.160000 Src 2 Dest -2
enquedrop: Time 176.170000 Src 2 Dest -2

enquedrop: Time 176.180000 Src 2 Dest -2
enquedrop: Time 176.190000 Src 2 Dest -2
enquedrop: Time 176.200000 Src 2 Dest -2
enquedrop: Time 999.780000 Src 2 Dest -2
enquedrop: Time 999.790000 Src 2 Dest -2
enquedrop: Time 999.800000 Src 2 Dest -2
enquedrop: Time 999.810000 Src 2 Dest -2
enquedrop: Time 999.820000 Src 2 Dest -2
enquedrop: Time 999.830000 Src 2 Dest -2
enquedrop: Time 999.840000 Src 2 Dest -2
enquedrop: Time 999.850000 Src 2 Dest -2
enquedrop: Time 999.860000 Src 2 Dest -2
enquedrop: Time 999.870000 Src 2 Dest -2
enquedrop: Time 999.880000 Src 2 Dest -2
enquedrop: Time 999.890000 Src 2 Dest -2
enquedrop: Time 999.900000 Src 2 Dest -2
enquedrop: Time 999.910000 Src 2 Dest -2
enquedrop: Time 999.920000 Src 2 Dest -2
enquedrop: Time 999.930000 Src 2 Dest -2
enquedrop: Time 999.940000 Src 2 Dest -2
enquedrop: Time 999.950000 Src 2 Dest -2
enquedrop: Time 999.960000 Src 2 Dest -2
enquedrop: Time 999.970000 Src 2 Dest -2
enquedrop: Time 999.980000 Src 2 Dest -2
enquedrop: Time 999.990000 Src 2 Dest -2
enquedrop: Time 1000.000000 Src 2 Dest -2

Packet Drop Parameter NS2 log_at Node 2 With Response:

enquedrop: Time 10.000000 Src 2 Dest -2
enquedrop: Time 10.010000 Src 2 Dest -2
drop: at time 137.729665 Src 2 Dest 1
drop: at time 137.729665 Src 2 Dest 1
drop: at time 137.729665 Src 2 Dest 1
drop: at time 141.681306 Src 2 Dest 1
drop: at time 141.681306 Src 2 Dest 1
enquedrop: Time 141.690000 Src 2 Dest -2
enquedrop: Time 141.700000 Src 2 Dest -2
enquedrop: Time 141.710000 Src 2 Dest -2
enquedrop: Time 141.720000 Src 2 Dest -2
enquedrop: Time 141.730000 Src 2 Dest -2
enquedrop: Time 141.740000 Src 2 Dest -2
enquedrop: Time 141.750000 Src 2 Dest -2
enquedrop: Time 141.760000 Src 2 Dest -2
enquedrop: Time 141.770000 Src 2 Dest -2
enquedrop: Time 141.780000 Src 2 Dest -2
enquedrop: Time 141.790000 Src 2 Dest -2
enquedrop: Time 141.800000 Src 2 Dest -2
enquedrop: Time 151.420000 Src 2 Dest -2
enquedrop: Time 151.430000 Src 2 Dest -2
enquedrop: Time 151.440000 Src 2 Dest -2
enquedrop: Time 151.450000 Src 2 Dest -2
enquedrop: Time 151.460000 Src 2 Dest -2
enquedrop: Time 151.470000 Src 2 Dest -2
enquedrop: Time 151.480000 Src 2 Dest -2
enquedrop: Time 151.490000 Src 2 Dest -2
enquedrop: Time 151.500000 Src 2 Dest -2

enquedrop: Time 151.510000 Src 2 Dest -2
enquedrop: Time 151.520000 Src 2 Dest -2
enquedrop: Time 151.530000 Src 2 Dest -2
enquedrop: Time 151.540000 Src 2 Dest -2
enquedrop: Time 151.550000 Src 2 Dest -2
enquedrop: Time 151.560000 Src 2 Dest -2
enquedrop: Time 151.570000 Src 2 Dest -2
enquedrop: Time 151.580000 Src 2 Dest -2
enquedrop: Time 151.590000 Src 2 Dest -2
enquedrop: Time 151.600000 Src 2 Dest -2
enquedrop: Time 151.610000 Src 2 Dest -2
enquedrop: Time 151.620000 Src 2 Dest -2
enquedrop: Time 151.630000 Src 2 Dest -2
enquedrop: Time 151.640000 Src 2 Dest -2
enquedrop: Time 151.650000 Src 2 Dest -2
enquedrop: Time 151.660000 Src 2 Dest -2

Energy Consumption Parameter NS2 log at node 2 Without Response

energy: at src node 2 at time 10.001870 is 99.454280
energy: at src node 2 at time 10.003240 is 99.454040
energy: at src node 2 at time 10.004151 is 99.453800
energy: at src node 2 at time 10.019865 is 99.448760
energy: at src node 2 at time 10.019865 is 99.448760
energy: at src node 2 at time 10.020000 is 99.448760
energy: at src node 2 at time 10.030000 is 99.444666
energy: at src node 2 at time 10.040000 is 99.441474
energy: at src node 2 at time 10.050000 is 99.436878
energy: at src node 2 at time 10.060000 is 99.433292
energy: at src node 2 at time 10.070000 is 99.431545
energy: at src node 2 at time 10.080000 is 99.429798
energy: at src node 2 at time 10.090000 is 99.428050
energy: at src node 2 at time 10.100000 is 99.426303
energy: at src node 2 at time 10.110000 is 99.424556
energy: at src node 2 at time 10.120000 is 99.422809
energy: at src node 2 at time 10.130000 is 99.421062
energy: at src node 2 at time 10.140000 is 99.419314
energy: at src node 2 at time 10.150000 is 99.417567
energy: at src node 2 at time 10.160000 is 99.415820
energy: at src node 2 at time 10.170000 is 99.414073
energy: at src node 2 at time 49.990000 is 92.233016
energy: at src node 2 at time 50.000000 is 92.231269
energy: at src node 2 at time 50.010000 is 92.228122
energy: at src node 2 at time 50.020000 is 92.224685
energy: at src node 2 at time 50.030000 is 92.221100
energy: at src node 2 at time 50.040000 is 92.219353
energy: at src node 2 at time 50.050000 is 92.217605
energy: at src node 2 at time 50.060000 is 92.215858
energy: at src node 2 at time 50.070000 is 92.214111
energy: at src node 2 at time 50.080000 is 92.212364
energy: at src node 2 at time 50.090000 is 92.210617
energy: at src node 2 at time 50.100000 is 92.208869
energy: at src node 2 at time 50.110000 is 92.207122
energy: at src node 2 at time 50.120000 is 92.205375
energy: at src node 2 at time 50.130000 is 92.203628
energy: at src node 2 at time 50.140000 is 92.201881
energy: at src node 2 at time 50.150000 is 92.200133

energy: at src node 2 at time 50.160000 is 92.198386
energy: at src node 2 at time 50.170000 is 92.196639
energy: at src node 2 at time 50.180000 is 92.194892
energy: at src node 2 at time 162.500980 is 75.184101
energy: at src node 2 at time 163.501100 is 75.182661
energy: at src node 2 at time 164.501460 is 75.179781
energy: at src node 2 at time 166.500980 is 75.178341
energy: at src node 2 at time 167.501180 is 75.176901
energy: at src node 2 at time 169.001280 is 75.175461
energy: at src node 2 at time 169.501040 is 75.174021
energy: at src node 2 at time 170.501420 is 75.172581
energy: at src node 2 at time 172.001340 is 75.171141
energy: at src node 2 at time 172.501420 is 75.169701
energy: at src node 2 at time 176.001100 is 75.168261
energy: at src node 2 at time 176.002890 is 75.168021
energy: at src node 2 at time 181.001240 is 75.161061
energy: at src node 2 at time 182.501380 is 75.158181
energy: at src node 2 at time 183.501260 is 75.156741
energy: at src node 2 at time 184.501500 is 75.155301
energy: at src node 2 at time 185.501320 is 75.153861
energy: at src node 2 at time 186.500940 is 75.152421
energy: at src node 2 at time 189.501080 is 75.150981
energy: at src node 2 at time 191.500940 is 75.149541
energy: at src node 2 at time 964.501160 is 74.203700
energy: at src node 2 at time 966.501320 is 74.200820
energy: at src node 2 at time 968.001240 is 74.199380
energy: at src node 2 at time 969.501460 is 74.197940
energy: at src node 2 at time 980.501060 is 74.192420
energy: at src node 2 at time 980.503130 is 74.192180
energy: at src node 2 at time 982.500940 is 74.188100
energy: at src node 2 at time 983.500920 is 74.186660
energy: at src node 2 at time 983.501850 is 74.186420
energy: at src node 2 at time 983.503950 is 74.185940
energy: at src node 2 at time 984.501280 is 74.182580
energy: at src node 2 at time 985.000980 is 74.181140
energy: at src node 2 at time 986.001360 is 74.179700
energy: at src node 2 at time 986.501040 is 74.178260
energy: at src node 2 at time 988.001040 is 74.176820
energy: at src node 2 at time 989.500940 is 74.175380
energy: at src node 2 at time 989.502810 is 74.175140
energy: at src node 2 at time 991.000920 is 74.172500
energy: at src node 2 at time 992.001320 is 74.171060
energy: at src node 2 at time 998.501060 is 74.165300

Energy Consumption Parameter NS2 log at node 2 With Response

energy: at src node 2 at time 10.001870 is 99.454280
energy: at src node 2 at time 10.003240 is 99.454040
energy: at src node 2 at time 10.004151 is 99.453800
energy: at src node 2 at time 10.019865 is 99.448760
energy: at src node 2 at time 10.019865 is 99.448760
energy: at src node 2 at time 10.020000 is 99.448760
energy: at src node 2 at time 10.030000 is 99.444666
energy: at src node 2 at time 10.040000 is 99.441474
energy: at src node 2 at time 10.050000 is 99.436878
energy: at src node 2 at time 10.060000 is 99.433292
energy: at src node 2 at time 10.070000 is 99.431545

energy: at src node 2 at time 10.080000 is 99.429798
energy: at src node 2 at time 10.090000 is 99.428050
energy: at src node 2 at time 10.100000 is 99.426303
energy: at src node 2 at time 10.110000 is 99.424556
energy: at src node 2 at time 10.120000 is 99.422809
energy: at src node 2 at time 10.130000 is 99.421062
energy: at src node 2 at time 10.140000 is 99.419314
energy: at src node 2 at time 10.150000 is 99.417567
energy: at src node 2 at time 10.160000 is 99.415820
energy: at src node 2 at time 10.170000 is 99.414073
energy: at src node 2 at time 10.180000 is 99.412326
energy: at src node 2 at time 10.190000 is 99.410578
energy: at src node 2 at time 10.200000 is 99.408831
energy: at src node 2 at time 10.210000 is 99.407084
energy: at src node 2 at time 10.220000 is 99.405337
energy: at src node 2 at time 10.230000 is 99.403590
energy: at src node 2 at time 10.240000 is 99.401842
energy: at src node 2 at time 10.250000 is 99.400095
energy: at src node 2 at time 10.260000 is 99.398348
energy: at src node 2 at time 10.270000 is 99.396601
energy: at src node 2 at time 10.280000 is 99.394854
energy: at src node 2 at time 141.620000 is 75.272189
energy: at src node 2 at time 141.623072 is 75.270441
energy: at src node 2 at time 141.630000 is 75.269477
energy: at src node 2 at time 141.633512 is 75.267729
energy: at src node 2 at time 141.640000 is 75.266765
energy: at src node 2 at time 141.643412 is 75.265017
energy: at src node 2 at time 141.650000 is 75.264053
energy: at src node 2 at time 141.653272 is 75.262305
energy: at src node 2 at time 141.660000 is 75.261806
energy: at src node 2 at time 141.665124 is 75.260059
energy: at src node 2 at time 141.670000 is 75.259757
energy: at src node 2 at time 141.674257 is 75.258009
energy: at src node 2 at time 141.680000 is 75.257813
energy: at src node 2 at time 142.001080 is 75.246437
energy: at src node 2 at time 142.002130 is 75.246197
energy: at src node 2 at time 142.005570 is 75.245237
energy: at src node 2 at time 142.007631 is 75.244757
energy: at src node 2 at time 142.502394 is 75.233444
energy: at src node 2 at time 142.506576 is 75.232661
energy: at src node 2 at time 143.000980 is 75.225789
energy: at src node 2 at time 143.001930 is 75.225549
energy: at src node 2 at time 143.002800 is 75.225309
energy: at src node 2 at time 143.500920 is 75.217793
energy: at src node 2 at time 143.502410 is 75.217553
energy: at src node 2 at time 143.503380 is 75.217313
energy: at src node 2 at time 143.504450 is 75.217073
energy: at src node 2 at time 144.500940 is 75.206901
energy: at src node 2 at time 144.502210 is 75.206661
energy: at src node 2 at time 145.001380 is 75.204261
energy: at src node 2 at time 145.500900 is 75.202821
energy: at src node 2 at time 146.500980 is 75.201381
energy: at src node 2 at time 147.501440 is 75.198501
energy: at src node 2 at time 148.001020 is 75.197061
energy: at src node 2 at time 149.001400 is 75.195621
energy: at src node 2 at time 149.501320 is 75.194181

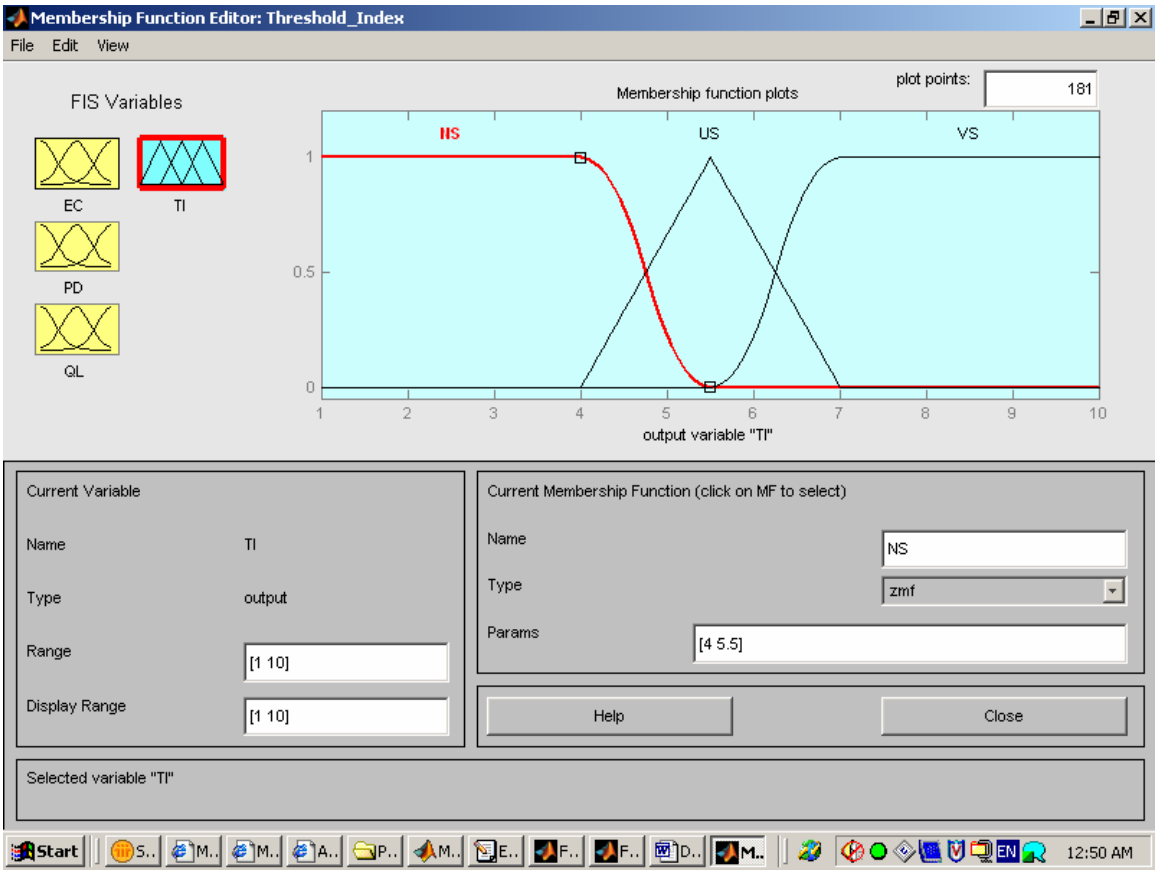
APPENDIX E

```
%**********************************************************************
      % VB code to validate VS TI threshold and MSE calculation.
      % Written by: Sathish Kumar AP        % Date: 08/10/2006.
%**********************************************************************
 Dim i As Integer
 Dim TIcrit As Double
 Dim TIcritfind As Double
 Dim Sum1 As Double
 Dim ProbA As Double
 Dim j As Integer
 Dim Sum2 As Double
 Dim ProbN As Double
 Dim Sum As Double
 Dim Oldsum As Double
 Dim Minsum As Double
 Sum1 = 0
 ProbA = 0.1
 Sum2 = 0
 ProbN = 0.9
 Sum = 0

 TIcrit = 1
 TIcritfind = 1
 Do While TIcrit <= 10
  Oldsum = Sum
  Sum = 0
  Sum1 = 0
  Sum2 = 0
  For i = 1 To TIcrit
  Sum1 = Sum1 + ((TIcrit - i) * (TIcrit - i)) * ProbAbn(i)

 Next i
 Sum1 = Sum1 * ProbA
 For j = TIcrit To 10
  Sum2 = Sum2 + ((TIcrit - j) * (TIcrit - j)) * ProbNorm(j)
 Next j
 Sum2 = Sum2 * ProbN
 Sum = Sum1 + Sum2
 If Sum < Oldsum Then
   Minsum = Sum
   TIcritfind = TIcrit
 Else: Minsum = Oldsum
 End If
 TIcrit = TIcrit + 0.1
 Loop
 picOutput.Print Minsum
 picOutput.Print TIcritfind
```

APPENDIX F



Sample Snapshot of Fuzzy Memberships in Fuzzy Model

PUBLICATIONS FROM DISSERTATION RESEARCH

[1] S. P. Alampalayam, A. Kumar, and S. Srinivasan, "Mobile ad hoc networks security – a taxonomy," in *Proceedings of ICACT Conference*, 2005, pp. 839-844.

[2] S. P. Alampalayam and A. Kumar, "Predictive security model using data mining," in *Proceedings of IEEE GlobeCom Conference*, 2004, pp. 2208-2212.

[3] S. P. Alampalayam and A. Kumar, "Adaptive security model for mobile agents in wireless networks," in *Proceedings of IEEE GlobeCom Conference*, 2003, pp. 1516-1521.

[4] S. P. Alampalayam and A. Kumar, "Security model for routing attacks in wireless networks," in *Proceedings of IEEE VTC Conference*, Fall 2003, pp. 2122-2126.

[5] S. P. Alampalayam and A. Kumar, "An adaptive and predictive security model for mobile ad hoc networks," *Kluwer Personal Communications Journal, Security Special Issue for Next Generation Wireless Networks*, vol. 29, pp. 263-281, June 2004.

[6] S. P. Alampalayam, A. Kumar, J. Graham, and S.Srinivasan, "Intruder Identification and Response Framework for Mobile Ad hoc Networks," Accepted in *International Conference on Computers And Their Applications*, March 2007.

CURRICULUM VITAE

Sathish Kumar Alampalayam
532, N. Mentor Avenue, Apt 2
Pasadena, CA 91106
Tel: 502-767-4764 (M) 626-793-4302 (H)
Email : sathish.ap@gmail.com
Resident Status: US Citizen

Education

- Doctor of Philosophy (PhD) in Computer Science and Engineering from JB Speed School of Engineering, University of Louisville, KY USA Current GPA 3.88/4.00 – May 2007
  Dissertation: Intrusion Detection and Response Model for Mobile Ad hoc Networks
- Master of Business Administration (M.B.A.,) from College of Business, University of Louisville, KY USA. GPA 3.87/4.00. – Dec 2001.
- Master of Science (M.S.,) in Computer Science from JB Speed School of Engineering, University of Louisville, KY USA. GPA 3.77/4.00. – May 2001.
  Thesis: Design of the Inter Cell Mobility Path Prediction for the Wireless Internet Hosts Using Artificial Neural Networks.
- Bachelor of Engineering (B.E.,) in Computer Science and Engineering from Government College of Technology, Bharathiar University, Coimbatore, India. GPA 3.7/4.0. - May 1996.

Current Areas of Teaching and Research Interest

- Network Security and Information Security
- Data mining and Neural Networks
- Soft Computing, Fuzzy Systems and Artificial Intelligence
- Mobile, Wireless and Wired – distributed systems and networking
- Information Systems Management and Strategy
- Systems Analysis and Design
- Applications Programming and Software Engineering
- Database/Data warehouse design and development
- Operating Systems and Performance Evaluation of Computer Systems

Professional Experience Summary

- Technology/Software/Systems Professional with around 10 years of experience including Client/Server Distributed Systems, Unix Mini/Server, IBM-Mainframe and Internet applications environment
- Areas of business and functional expertise include Finance, Banking, Mortgage, Healthcare, Insurance, Accounting, Logistics, Network Analytics and Shipping Operation domains.
- Expert on all the stages/functional aspects of application and software engineering life cycle.
- Proven leadership, project management and result oriented abilities.
- Excellent communicational, interpersonal, technical, business, quantitative, analytical, problem solving, decision making and critical thinking skills.
- Effectively led teams in the execution of development, maintenance and enhancement projects.
- Tremendous ability to quickly learn and master any tasks and skills required to perform job.

Professional Memberships

- Member, IEEE (Institute of Electrical and Electronics Engineers).
- Member, Project Management Institute (PMI).
- Member, American Society for Quality (ASQ).
- Life Member, Beta Gamma Sigma, The honor society for AACSB accredited business programs.

Professional Experience

IT Architect, Countrywide Financials, Pasadena, CA 05/2005 – Present

Adjunct Faculty, California State University, Los Angeles, CA 03/2006 - Present

Adjunct Faculty, National University, Los Angeles, CA 03/2006 - Present

Sr. Applications Developer/DBA 03/2004 – 04/2005
Financial ServiceSolutions, Bank of America subsidiary, Louisville, KY

Systems Analyst II, Bank of America, Louisville, KY 05/2003 – 02/2004

Software Engineer, APS Technologies, Louisville, KY 01/2003 – 05/2003

Systems Engineer, Siemens Dematic (For UPS), Louisville, KY 11/2001 – 12/2002

Software Engineer, Humana Inc., Louisville, KY, USA 10/1997 – 10/2001

Asst Systems Analyst, Tata Consultancy Services (TCS), India 10/1996 - 10/1997
Client: American President Lines (APL), Oakland, CA USA

Technical Skill Set

Programming Languages	C#.NET, VB.NET, Visual Basic 6, C++, ASP.NET, ADO.NET, C, Prolog, Lisp, Java 2, JSP, JavaScript, Visual C++, Shell Script, HTML, XML, Perl, OTcl, T-SQL, PL/SQL, Pascal, Fortran, Cobol, CICS, Rexx, Adso, JCL, Assembler.
Databases	SQL Server 2000, Oracle 9i/8x, DB2, Terra Data, IDMS, VSAM, MS Access.
Operating Systems(OS)	Windows NT/2K/2K3/XP, RedHat Linux 8, Sun Solaris 10, HP-Unix 11, IBM MVS ESA, OS/390, IBM AIX/6000 version 4.3.3, SVR 4.2 Unix.
Hardware	Intel Xeon, HP 9000, Sun V1280, IBM 9672-R44, IBM 3090, Modicon PLC
Reporting Tools	Actuate 7, Easytrieve, Doc-Analyzer, Focus, CA-Culprit.Database Tools Toad, AQT, QueryMan, SQLDiff, SQL * Plus, SPUFI.
Network Tools	NetOP, GloMoSim, SNNS, NS-2, Sniffer, VNC, PC Anywhere, FTP, NDM, Putty, SSH, Telnet, EasyTerm, TCP/IP, HTTP, SNMP.
OS Tools	Sniff, X-Windows, HP SAM, TSO/ISPF, Shell Script, AWK, MC/Service Guard, Tivoli Agent, HP Glance Plus.
Development Tools	MS IIS 6, IBM Web Sphere, Java Webserver, Visual Studio.NET, Visual Interdev, Visual SourceSafe, Quickjob, VBS, VBA, MS Project, MS Office, Primal Script
Other Software Packages	MiniTab 14, CART 2.0, GoldMine.
Testing Tools	Mercury Quick Test 8, Test Director, LoadRunner 8, GDB, Viasoft Smarttest.

Other Professional Activities:

- Reviewer for Kluwer Wireless Personal Communications Journal Security for Next Generation Special Issue
- Reviewer for ISCC 2004 Conference
- Reviewer for Journal of Intelligent and Fuzzy Systems, 2005

Certification

- Advanced Diploma in Systems Management from NIIT – July 1996.
- Project Management Professional (PMP) certification from PMI – September 2005.

Publications:

[1] S.P. Alampalayam, A.Kumar, "An Adaptive and Predictive Security Model for Mobile Ad hoc Networks", Kluwer Personal Communications Journal, Security Special Issue for Next Generation Wireless Networks, 2004.

[2] S.P. Alampalayam, A.Kumar, "Security Model for Routing Attacks in Wireless Networks", Proceedings of IEEE VTC 2003.

[3] S.P Alampalayam, A.Kumar, "An Adaptive Security Model for Mobile Agent Based Wireless Networks", Proceedings of IEEE GlobeCom 2003.

[4] S.P. Alampalayam, A.Kumar, A.Sleem, R.K.Ragade and J.P. Wong, "Artificial Neural Network for mobility prediction in enterprise integration", Proceedings of EDA-2001, pp [30-35].

[5] S.P. Alampalyam, A.Kumar, "Predictive Security Model using Data Mining, IEEE GlobeCom 2004.

[6] S.P. Alampalayam, A.Kumar, S.Srinivasan "Mobile Ad hoc Networks Security – A Taxonomy", IEEE ICACT, Korea 2005.

[7] S. P. Alampalayam, A. Kumar, J. Graham, and S.Srinivasan, "Intruder Identification and Response Framework for Mobile Ad hoc Networks," *Accepted in International Conference on Computers And Their Applications*, March 2007.

[8] S.P Alampalayam, V. Chandrakantan "Parallel Computing: A Parallel Algorithm for Multi Stage Optimization Problem and Shortest Path Problem" IEEE- Anna University, Madras 1994.

References:

Available upon request.