SECURITY AND PRIVACY IN RESOURCE CONSTRAINED WIRELESS NETWORKS

By

Kanthakumar Mylsamy Pongaliur

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Computer Science

2012

ABSTRACT

SECURITY AND PRIVACY IN RESOURCE CONSTRAINED WIRELESS NETWORKS

By

Kanthakumar Mylsamy Pongaliur

Wireless networks use radio waves as a communication medium which allows for faster and cheaper deployment. The networks being wireless, are out in the open, which makes them vulnerable to malicious users that can hinder their performance. Of the several types of wireless networks, we focus on security and privacy in wireless sensor networks (WSN) and cognitive radio mobile ad-hoc networks (CR-MANET). The devices in these networks are limited in resources such as energy, low power radio, etc. CR-MANET devices are mobile, requiring them to run on limited amount of energy supplied by batteries, and conserve energy by reducing communication cost using low power radios. In addition, sensor devices have limited storage and a slower CPU. The purpose of a WSN is to sense and report event occurrences, whereas a CR-MANET provides improved spectrum utilization.

We studied three kinds of attacks on WSN. The first type of attack is on the source privacy of sensor nodes. This attack happens because an important characteristic of events detected by sensor devices is bound to the location of event occurrence that can be revealed by compromise of detecting sensor device's source privacy. Thus, protecting privacy of event detecting sensor device is of paramount importance for which we present an encryption based solution to protect source privacy under eavesdropping and node compromise attacks. The second type of attack by the malicious entity can be invasive in nature, which could possibly cause damage to the device, or can be passive as in side channel attacks. A comprehensive study of side channel attacks on WSN is presented, along with a process obfuscation tech-

nique. The third type of attack is on the propagation of data packet generated by the sensor device. The detected event data is sent to the base station. If a malicious entity is able to prevent such event reporting packets from reaching the base station and segregate the attack zone, it will be able to carry out its malicious activity without getting caught. To cover such scenarios, a proactive dynamic camouflage event generation solution is presented.

CR-MANET devices sense for vacant licensed spectrum and improve its utilization in an opportunistic manner. Accurate licensed spectrum occupancy detection by a CR-MANET device is hampered by signal fading, hidden terminal problems, etc. Spectrum occupancy decision can be improved by cooperative spectrum sensing (CSS). However, CSS is made difficult by the presence of malicious users. The malicious users can have two goals: one is to disrupt the network, another is to manipulate the network for its own personal gains. The malicious users can create havoc in a CR-MANET by falsifying spectrum sensing information leading to interference with the primary user. The devices in a CR-MANET are mobile, which gives an opportunity for the malicious entity to hide behind the changing neighborhood. We present three solutions to overcome the spectrum sensing data falsification attack and incorrect reporting of signal measurement due to byzantine failures. The first is a multi-fusion based distributed spectrum sensing (MFDSS) using reputation propagation. In the second solution, a continuously evolving virtual neighbor cluster of past neighbor devices aid in validating the input gathered from the current neighboring devices (ReNVaS). Third, a recursive partitioning around medoids based clustering is performed to identify a tightly bound set of valid inputs for decision making (TMC). A unified and non-unified decision making strategy is presented using ReNVaS and TMC. MFDSS performs better in a fast changing network while performance of unified fusion is better in a slow mobility network with respect to primary user spectrum occupancy detection accuracy.

To my parents

ACKNOWLEDGMENTS

Pursuing a Ph.D. was a long but eventful experience. During this journey, there were many people without whose support and guidance, I would not have been able to reach this far.

First and foremost, I am extremely grateful to Dr. Li Xiao for her guidance, vision and unending patience. We spent hours brainstorming ideas, she helped me define the problems and taught me to write research papers. Without the immense support and encouragement I would not have been able to complete my dissertation.

I would like to express my gratitude to Dr. Matt Mutka, Dr. Vidyadhar Mandrekar, and Dr. Alex Liu for sparing their precious time to serve on my dissertation committee. Their valuable suggestions helped tremendously to shape this thesis.

I would like to thank my colleagues in the ELANS lab. They inspired me and their friendship helped me to remain focused on days when things didn't seem to go right. We bounced ideas of each other which resulted in some wonderful research collaborations.

I would like to thank my parents and my brother for their unconditional love and support. Their love provided me inspiration and was my driving force. Last but not the least, I would like to thank my wife Meena, whose love and encouragement helped me finish this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

With the advancement of technology, the variety of applications and the possibility of attaining high throughput in wireless networks has increased significantly. This has led to the development of different forms of wireless networks designed for specific tasks to cater from field monitoring using sensor networks to attaining high throughput by monitoring the spectrum occupancy by the primary user in a cognitive radio network. The increase in the wireless networks application domain has been inversely proportional to the size of the devices with the devices becoming smaller. As an example, a typical sensor device measures the size of a matchbox (IRIS, MICA2, etc.) with the smallest ones being of the size of dust particles [1]. The reduced size introduces limitations like low processing power and battery life, limited memory, etc. This decrease in the form factor (smaller size) helps the sensor device to be inconspicuous in the sensing area and is useful for tracking events without being recognized. Another wireless device is a mobile ad-hoc device equipped with a cognitive radio which is capable of sensing for spectrum availability. With cognition, the device can opportunistically use unoccupied licensed spectrum, thereby improving the spectrum usage and throughput. The sensor networks and the cognitive radio networks are similar in characteristics. Sensor networks sense the environment to detect the occurrence of events, while a cognitive radio network senses the spectrum to check if it is vacant. The former results in the report of events while the later helps to improve spectrum utilization.

The reduced form factor, the growth in the number of applications, the wireless com-

munication medium and the introduction of different technologies has created new security problems. In these wireless networks, the attacks can be on the privacy of the sensor devices, side channel attack on sensor devices, attacks on data collected or exchanged by the sensor devices, and mal-intent in the spectrum sensing ability of a cognitive device. In this thesis, of the several types of attacks possible, we study the prior mentioned attacks; identify their unique properties and present solutions to prevent the same.

1.1 Attacks on Source Privacy

Sensor networks have significant salience to different fields, ranging from military applications to personal health monitoring. The sensor devices use wireless transmissions, which is out in the open and can be overheard in the communication vicinity. Without precaution, a malicious entity overhearing packet transmission can trace-back the packet to the source. This can lead to the position of the source being revealed along with the location and time of event occurrence. Considering a mission critical military application or a simple environmental application, any leakage of information such as event location or time can prove beneficial to the adversary and costly to the network goal. Hence, irrespective of the application type, privacy of the monitored event holds importance which in turn requires providing privacy to event reporting source nodes. Maintaining Source Privacy under Eavesdropping and Node Compromise Attacks (SPENA) is a scheme that aims to maintain source privacy under eavesdropping by the malicious devices which are also capable of compromising sensor nodes. A one-way hash chain based keying mechanism is used to hide the source information. The one-way hash function generates a series of one-time use keys. This is further used to obfuscate an additional partial hash by dynamically selected nodes preventing a trace-back

by the adversary. SPENA addresses super-local eavesdropping and node compromise attacks by malicious devices. The functioning of SPENA is evaluated under different attack scenarios and found to perform exceptionally well when compared to existing state of the art with modest overhead. Eavesdropping is an attack type that is also classified under side channel attacks.

1.2 Side Channel Attacks

In side channel attacks, an adversary accesses a sensor device in a non-invasive (i.e., non-tampering) manner and gains confidential information by observing the node under normal operation. In such attacks, the goal of the adversary is to deduce the inner workings of the hardware or the software. The adversary may use a variety of techniques such as power analysis (simple power analysis and differential power analysis), execution cycle frequency analysis, timing information (on data movement into and out of the CPU) analysis, electromagnetic leakage analysis, acoustic emission analysis, etc. The unique characteristics of sensor devices make the prevention of side channel attacks more challenging. We present a comprehensive study of side channel attacks on sensor devices. To begin with, a three-phase attack model on sensor nodes is formalized. Second, a proof of concept is presented with experimental results to show the feasibility of electromagnetic radiation leakage attacks using readily available equipment. Third, a family of side channel attacks on sensor devices is identified, which include electromagnetic radiation leakage attacks, optical side channel attacks, traffic analysis attacks, power analysis attacks, timing attacks, fault analysis attacks, acoustic attacks, and thermal imaging attacks. For each type of the attack, viable countermeasures are presented to limit vulnerability. Also proposed is a technique called

'process obfuscation', which can be used to counter a variety of side channel attacks.

The two attack forms discussed so far do not consider manipulation of the event data itself. The next attack is on the event information sensed and the event reporting packet generated by the sensor device.

1.3 Event Data Propagation Prevention Attack

The side channels and source privacy attacks are used to gain knowledge of the information available at the sensor devices. Another form of attack which is of importance is one in which the adversary is aware of the event information and wants to prevent that information from reaching the base station. In such attacks, collaborative adversaries' goal is to segregate part of the network in order to prevent event reporting packets from reaching the base station, by either dropping or by corruption of the packet. Also, if the events are sporadic, the base station will not be able to differentiate between non-occurrence of an event and non-report of an event due to malicious activity. This is important in military applications such as detecting heavy artillery movement.

To overcome this problem, a Dynamic Camouflage Event based Malicious Node Detection Architecture (D-CENDA) is presented. D-CENDA is a proactive architecture in which the base station exploits the spatial and temporal information of the camouflage event to detect the malicious device. D-CENDA uses a multi-phase approach that includes camouflage event generation, planning the mobile-node tour, encoding the address in the packet, packet analysis by the base station, detection and verification of the malicious node. A camouflage event is a reputable event generated in response to a base station request. The camouflage event generator is a mobile-device which can be mounted on robot or an unmanned aerial

vehicle. This mobile-device traverses the path decided by the base station and generates the camouflage events at regulated intervals of time. These events are called camouflage events since they mimic the real events, but are not real events in the true sense. A simple solution is for the base station initiating a camouflage event from the node by sending a message, but a powerful adversary can overhear the communication and allow such event packets. The nodes which are in the sensing range of the camouflage event location will detect the event and report back to the base station. A lightweight address encoding scheme wherein each node encodes the shortened relative address of the node from which it received the packet is presented. Each node also maintains the information about the overheard packet transmissions by the neighbors. For this, a bloom-filter [2] and a bit-array based method is presented, which are used for verification before branding a node malicious. The performance of D-CENDA is analyzed and compared to existing solutions. The results indicate that D-CENDA is not only able to identify the malicious device zone with high accuracy, but also recognize the attack type.

So far we saw attacks on the meta-information about the events in the form of compromise of source privacy, attacks on side-channels and attacks on the collected event information. The next type of attack is on the functioning of the cognition feature of a cognitive radio mobile ad-hoc network. This directly affects the throughput of the secondary user network and in some cases the feasibility of the network itself.

1.4 Attacks on Cognitive Radio Networks

In the attacks so far, we looked at sensor devices with limited resources and how the attack on either the data collected or the device itself (hardware/software) posed problems. The

growth in technology has led to development of intelligent devices which can deduce for themselves whether they should be communicating on a particular spectrum. These devices have been given cognitive features to be aware of the surrounding environment. Using a cognitive radio, the device can sense the environment and deduce if the spectrum is vacant or occupied. This is required because most of the spectrum has been awarded to a few licensed users (primary users), but this licensed spectrum has been found to be severely underutilized. Hence, FCC has mandated that the licensed spectrum can be utilized by unlicensed users (secondary users) when it is detected to be vacant, i.e. not being currently occupied by the primary user. One important requirement is that the secondary users (SU) should not interfere with the primary users (PU). This calls for accurate spectrum sensing and decision making which is made difficult by signal fading, shadowing, hidden terminal problems, etc. Due to these problems, some devices may be unable to reliably sense the spectrum, while the devices surrounding them may not be affected by these problems and hence sense the spectrum accurately. This has led researchers to believe that utilizing cooperative spectrum sensing can provide a robust solution. Cooperative spectrum sensing performs well, because you have multiple devices working together to improve the detection accuracy of primary user spectrum occupancy. Incorrect measurements by a few devices can be overcome by the collective accurate inputs from other devices. This becomes difficult to attain in the presence of malicious secondary users. The goals of a malicious user can be two pronged: one, they may want to disrupt the primary user network. Second, the adversary may wish to utilize the entire vacant spectrum band for itself when the primary user is not present. Such an attack launched by malicious devices encouraging other devices into making an incorrect decision about PU spectrum occupancy by spreading wrong data is called spectrum sensing data falsification attacks (SSDF).

To overcome this problem of malicious secondary users, a lot of research studies have utilized centralized cooperative spectrum sensing. A centralized cooperative spectrum sensing solution is robust since it will make decisions taking inputs from a large number of secondary users. When you consider the mobile secondary devices forming a cognitive radio mobile ad-hoc network, a centralized spectrum sensing approach is infeasible due to the absence of appropriate infrastructure. Additionally, the secondary users in such a network may be resource constrained and have limited energy, low power radios, etc. Since the devices are mobile, they depend on battery as a source for energy. Energy from a battery can last from a few hours to a few days depending on usage. This is particularly important when you consider small form factor hand held CR-MANET devices like smartphones, personal digital assistant, etc. The small form factor does not allow the device to accommodate larger batteries. One of the easier ways to conserve energy is by using low power radios, which restricts the transmission range. Communication consumes much higher energy compared to computation. It is noted that transmitting $1Kb$ of data over $100m$ consumes the same amount of energy as executing 3 million instructions on a processor with $100MIPS/W$ power [3], [4]. Hence, the solutions for a CR-MANET constrained in resource like energy should have the least possible communication overhead. In such cases, a distributed cooperative spectrum sensing solution gives the best opportunity for accurate spectrum occupancy detection.

In this thesis we present three solutions to overcome the problem of SSDF attacks and byzantine failure of devices. The first solution is called 'Multi-Fusion based Distributed Spectrum Sensing' (MFDSS). MFDSS includes three steps, namely, sensing data fusion, reputation propagation and fusion, and decision fusion. Unlike existing distributed schemes where a device presents only a local decision to its neighbor, in MFDSS the device transmits the actual measured sensing data to its neighbor. Collection of the actual data instead

of a binary decision from its neighbor allows the data collector to pre-process it prior to data fusion. It is our understanding that to catch byzantine failure you require information with much higher granularity than having a binary decision information. Also, compared to centralized approaches, the amount of data inputs is limited in distributed cooperative spectrum sensing, thereby hindering the use of statistical methods requiring large inputs. The effect of erroneous information due to SSDF attacks that escape detection during outlier detection are suppressed using a reputation scheme. During this, the remaining cohesive observations are weighed by their reputation and are fused to make a local decision about the presence of primary user signal. This is the data fusion step. This local decision data is exchanged with the neighbors and fusion of these decisions is performed to reach the final decision. This step is the decision fusion. Although this entire process occurs locally at one hop radius requiring only one hop communication, it incorporates sensing information from two hop neighbors, thereby encompassing inputs from a larger area. The devices are mobile leading to changing network topology and changes in neighborhood. To cater to the changing dynamics, we present a reputation propagation and fusion method based on the reputation of devices and the timestamp of the last update of the reputation. This has three benefits: First, it provides an opportunity for devices to make robust decisions using propagated reputation of new neighbors. Second, the reputation of a malicious device is propagated which reduces its likelihood to hide its malicious activities under the changing neighborhood. Third, it helps to maintain the freshness of the reputation information. In addition, to prevent attacks similar to a sybil attack wherein a device takes different identities in order to escape from bad reputation, we introduce an incubation period during which the data is collected, but not utilized for decision making.

The next two solutions are called 'Recursive Neighbor Validation' (ReNVaS) and 'Tight

Medoid Clustering' (TMC). In the recursive neighbor validation method every device maintains a virtual cluster of all the devices with which it has been neighbors in the past. The set of devices in the virtual cluster which have good reputation are used to validate the input from the neighboring devices before they are incorporated in the decision cluster. In the tight medoid clustering scheme, the inputs from the neighbor are recursively clustered to find the tightest bound set of inputs and these get added to the decision cluster. Since the neighborhood changes frequently, the medoid clustering scheme may not incorporate any reputation information. The data in the decision clusters are fused to make a decision about spectrum occupancy. We present two fusion algorithms: 'non-unified data fusion' and 'unified data fusion'.

MFDSS, ReNVaS, and TMC are evaluated, their overheads analyzed and performance compared to existing studies. MFDSS has a quick decision turnaround time. Compared to ReNVaS, MFDSS requires two sets of one hop communication and intermediate data processing to reach a primary user decision. ReNVaS needs to grow recursively to fetch inputs from devices further away (to validate the neighbor inputs) which may need additional time depending on the recursion number. In a very fast paced network, a combination of ReNVaS and TMC (using unified or non-unified fusion) ends up using just TMC for decision making, which is not as robust as a reputation propagation scheme like MFDSS. Unified fusion and non-unified fusion perform extremely well in a slow mobility network, with the unified data fusion incorporating ReNVaS and TMC performing the best with the lowest number of false negatives in primary user spectrum occupancy detection. In a fast paced mobile network, MFDSS performs better and the performance degrades gracefully as the number of malicious users grow beyond 40%.

1.5 Layout

Chapter 2 discusses the importance of source privacy in a sensor network and presents a solution to mitigate the same. Chapter 3 introduces the side channel issues and presents various countermeasures. Chapter 4 presents event data propagation prevention attack and provides a dynamic camouflage event based malicious node detection architecture to overcome the same. In Chapter 5, we discuss the spectrum sensing data falsification attacks in cognitive radio network and three solutions to overcome it. Finally, we present the conclusion in Chapter 6.

Chapter 2

Attacks on Source Privacy

A sensor network is composed of low-cost sensing devices which sense the environment for event occurrences and report back to the base station. The sensor devices use wireless transmissions, which is out in the open and can be overheard in the communication vicinity. Without precaution, a malicious entity overhearing packet transmission can trace-back the packet to the source. This can lead to the position of the source being revealed along with the location and time of event occurrence. Source privacy is generally compromised by the meta or contextual information of a packet, and not by the actual content of the packet. This has led many researchers to note that, source privacy cannot be addressed by encryption alone. A significant amount of studies guided towards using some form of simulating the source [5], [6], [7], [8], [9] or using a random walk [5], [10] has been performed to guarantee source privacy. The primary drawback of these approaches is the amount of overhead incurred to simulate a source or to redirect traffic randomly. Additionally, the studies so far only consider eavesdropping (mostly local, but some global) adversaries, while a malicious adversary that can easily compromise nodes is not considered as part of the threat model. We consider a strong threat model, in which the adversary compromises nodes while being able to eavesdrop over the communication network. The adversary has access to all the cryptographic elements of the compromised node.

In this chapter an encryption based method called 'Maintaining Source Privacy under Eavesdropping and Node Compromise Attacks' (SPENA) to improve source privacy is pre-

sented. A one-way hash chain based keying mechanism is used to hide the source information. The one-way hash function generates a series of one-time use keys. This is further used to obfuscate an additional partial hash by dynamically selected nodes preventing a trace-back by the adversary. The threat model considered allows the adversary to super-locally eavesdrop, while also being able to compromise nodes. In super-local eavesdropping, the adversary eavesdrops over a local area but can overhear communication over significantly larger coverage area than a sensor node. An example of such an adversary is a laptop class attacker. When a node is compromised, the adversary has access to all the cryptographic information available to the node. The case in which the adversary has access to data packets of past communications (stored at the compromised node) is also considered. In addition, in SPENA, the base station can recover a corrupt packet by querying an intermediate node and estimating the location of the compromised node. This is presented as part of the SPENA protocol for working in a single path routing as well as a flooding based approach. A detailed analysis of SPENA is provided and a comprehensive comparison with existing schemes is presented to show its effectiveness and efficiency.

The Chapter is organized as follows. In Section 2.1, the problem statement is presented. The SPENA scheme is described in Section 2.2. The SPENA protocol is presented in Section 2.3. This is followed by the analysis and evaluation in Section 2.4. Section 2.5 details the existing research on source privacy. Finally, the summary is presented in Section 2.6.

2.1 Problem Statement

In this section, we begin by describing the sensor network model followed by a threat model with the adversary having super-local eavesdropping capability, and the ability to compro-

mise a limited number of sensor nodes. Based on the network and the threat model, the problem statement is defined.

2.1.1 Network Model

The sensor network is a homogeneous network of sensor devices spread over a vast area. The sensor devices detect events and report back to the base station. The base station is secure and extremely powerful when compared to the deployed sensor devices. SPENA can be used for many applications requiring source node privacy. One class of applications is for tracking species of endangered animals or birds. Entities of such species need to be protected from hunters and poachers as they have great market value, while at the same time they need to be studied. The application considered is one in which the sensor network is deployed in a forest for sensing endangered birds (e.g. bald eagle). It is a homogeneous network, except for a secure base station, and consists of light weight sensor nodes dispersed over a large area. Nodes sense the environment for the presence of endangered bird(s), and on detecting the same, report back to the base station. The occurrence of the events can be sporadic and irregular in nature. Multiple sensor nodes can detect the event and will independently report it back to the base station. Aggregation of the data occurs at the base station. This scenario is depicted in Figure 2.1.

The base station gathers the information and is able to correlate it to identify the presence of the tracked entity, while also being able to study the flight patterns and their nesting habits. Given the tracked entity being swift and airborne, it could result in multiple sensors detecting the event in a short period of time, and then not having any detection for a prolonged duration of time. The assumption is that, if the bird swooped down at a particular location, it did so to either prey or to get water, and this increases the possibility of the bird

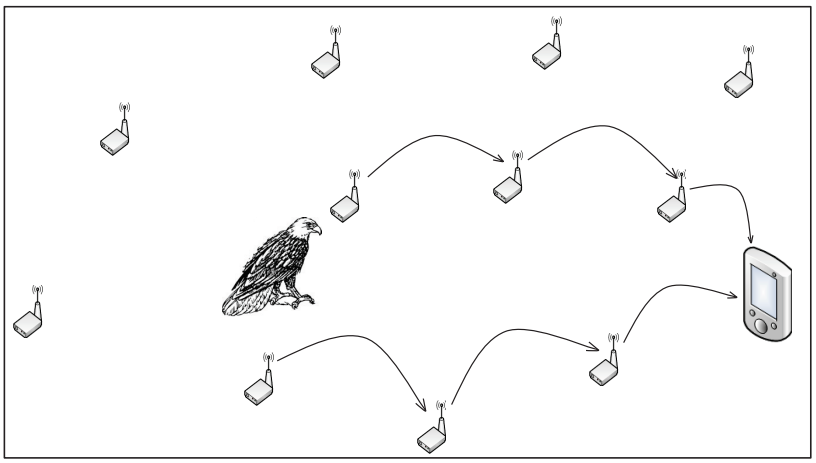


Figure 2.1: Sensor Network

descending to that location thereby requiring source location privacy (secrecy) for the event reporting packet. In some applications the event can be sporadic, but it must be noted that there is still other communication between the sensor nodes, resulting in the generation of packet traffic.

2.1.2 Threat Model

The adversary is a super-local eavesdropper, and inconspicuously eavesdrops over a sensor network. By super-local eavesdropping capability, we mean the adversary can overhear communication at much larger distances compared to a sensor node. This overhearing radius can be up to 5 times that of a regular sensor node, thereby providing the adversary a much larger overhearing area (up to 25 times) as compared to the sensor node. It also possesses the ability to compromise nodes. The adversary can be a single entity or a malicious network in itself that senses over the deployed sensors. The adversary can operate in two modes:

1. Super-local eavesdropper: The adversary in this mode is passive in nature, and can listen to the packet being transmitted over a large area in the network. The adversary can either have a very powerful transceiver to overhear the communication, or may

have a network of its own to achieve eavesdropping. As a super-local eavesdropper, the adversary overhears the communications and can correlate the transmission of the packet over multiple hops. Although, the adversary cannot ascertain the contents of the packet when it eavesdrops over the transmission, it has the capability to compare two packets.

2. Stealth mode: An attacker in the stealth mode can compromise a node and get access to all the cryptographic information stored in the node. In this mode, the adversary takes control over the sensor node, and can decode the packet, and get information as available to any rightful sensor node. The adversary can either compromise a certain percentage of nodes randomly selected from the network, or can compromise nodes which are geographically close to each other. Another attack form would be to compromise neighboring nodes.

The goal of the adversary is to identify the location and time of event occurrence, either by passive super-local eavesdropping or using intrusive node compromise. Alternately, the adversary may employ both. The problem is to conceal the event occurrence location and time by protecting source privacy under an eavesdropping and node compromise attack.

2.2 SPENA Scheme

SPENA is a source privacy protection scheme which uses one-way hash chains and mapping functions. The basic idea is introduced in the preliminaries subsection, and the details of the scheme are presented in the second subsection.

2.2.1 Preliminaries

In SPENA, a one-way hash chain is used to hide the source information. A one-way hash chain is a series of hash values generated by a one-way hash function. A basic idea of our approach is as follows. First, a unique hash function is used to generate a hash for source identification. This function is available at the source node and the base station. Second, dynamically selected intermediate nodes on the routing path alter the packet resulting in a change to the packet structure. Altering the packet structure disorients the packet trace-back by the adversary. This modification of the packet structure is done such that the base station can verify the information and connect it to the source generating the same. Third, on receipt of a corrupt packet, the base station is able to query the rehashing intermediate nodes for the packet.

Figure 2.2: Event Packet

SPENA can be used with a single path routing or a flooding based routing method. The event packet structure in SPENA is presented in Figure 2.2, and it has the following parameters:

1. DstID (Destination-id): It is the destination id of the packet. This is base station for event packets.

2. SrcID hash (Source-id Hash): A unique hash of the source that is used to identify it at the base station.

3. Obfuscating partial hash (OPH): This is generated by the source using the same hash function used to create the SrcID hash, and will be modified by dynamically selected

intermediate nodes.

4. Rehash seed: Used to determine the intermediate nodes to reconstruct the packet en route to the base station.

5. Payload Length: The length of a payload in the packet.

6. Payload: Payload is the actual data transmitted in the packet. It is encrypted using the symmetric key shared by the source node and the base station. In the application presented in Section 2.1, the payload consists of start time and duration of event occurrence.

7. Filler: Filler is used to provide a standard length to the packet and is populated with random garbage data. Having a constant packet size is required to prevent the adversary from tracking the packet based on increase in packet length during packet transformation. Keeping all the packets the same length forces the adversary to consider all packets while tracking back, and it cannot discount any packet analysis requirement based on the packet length. Using a garbage filler marginally increases the overhead, but the overhead is still very less compared to existing random walk or fake packet generation schemes as shown in Section 2.4.3.

In the pre-deployment phase, each node is bootstrapped with four functions and a symmetric key. A *hash function* to generate the one-way hash chain, *a second one-way hash function*, *a mapping function*, and *a rehash function*. The entries from the hash chain are used as source-id (SrcID) hash and the obfuscating partial hash (OPH) as depicted in Figure 2.2. The hashes (SrcID hash and OPH) perform two critical functions. First, they provide privacy to the packet by preventing an adversary from performing a trace-back to the source.

Second, they help the base station to verify the validity of the packet received. Additionally, two event reporting packets from the same source will have different SrcID hash, protecting the network from an adversary by preventing the source of the two packets to be correlated. The second one-way hash function and the mapping function are used to select the intermediate nodes for modifying the packet. These selected nodes use the rehash function to hash the OPH into a fixed length, which is then encrypted using the key. Additionally, the rehashing node concatenates the SrcID hash from the received packet to the payload which is then encrypted using the symmetric key of the rehashing node. It updates the payload length field with the new payload length and subtracts the corresponding amount of bits from the filler field so as to maintain the standard packet size. The new filler field is filled with random garbage bits. The rehashing node replaces the SrcID hash with the hash from its hash-chain. The *symmetric key* is unique and known only to the base station and the corresponding sensor node.

2.2.2 SPENA

As introduced earlier, SPENA uses a one-way hash function to hide the source information, a rehashing scheme to dynamically select the intermediate nodes for altering the packet, and finally, packet reception and verification at the base station. These steps are elaborated in the following subsections.

2.2.2.1 One-way Hash Function

One-way hash function is attractive to use in sensor networks, because the values can be verified in a fraction of the time as compared to a digital signature. Given the resource constraints in sensor networks, the light chain method is used to generate the hash chain as

described in [11]. Light chain generates a smaller hash when compared to the other methods. This is advantageous to use in sensor network given the limitations on the packet size and helps reduce the communication overhead. It can be further truncated without reducing the effectiveness with the use of two hash values, the SrcID hash and the OPH. It should be also noted that, as in all hash chain usages, it is used in the reverse order and once the elements are exhausted, the hash is reset by the base station. This is easily performed since the base station keeps track of the hash value usage. When the hash chain entries near exhaustion (less than 10% of entries are unused), the base station resets it by transmitting a new seed to the sensor node. This packet is encrypted using the unique key shared by the sensor node and the base station. This procedure consumes energy and will dictate the number of hash entries to be stored at the senor node. The number of entries stored at the sensor node has a tradeoff with the frequency of hash chain reset.

2.2.2.1.1 Hash-chain Usage Hash chain in SPENA is used for source identification and creating packet obscurity by certain randomly selected nodes on the packet path. This is different from the traditional sense of hashing that is primarily used for checking packet integrity in the form of a message digest.

Consider a network of n sensor nodes. $H_i()$ is the one-way hash function for node i, generating a sequence of values h_i^1, h_i^2, ... etc. When node i wants to send an event packet to the base station, and has already utilized $m-1$ entries from the one-way hash chain, it generates a packet shown in Figure 2.3, where h_i^m is the m^{th} entry in the hash chain of node i.

The entry h_i^m (SrcID hash) and h_i^{m+1} (obfuscating partial hash) are consecutive entries generated by the hash function. The hash value h_i^m identifies the source at the receiver (base

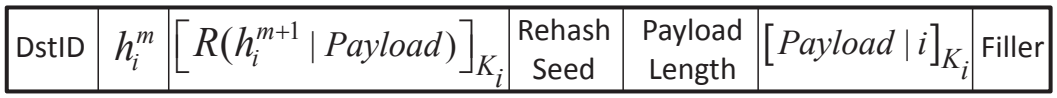


Figure 2.3: SPENA Event Packet

station). A hash size of 4 bytes is used to prevent collisions while usage of smaller hash size is discussed later. h_i^{m+1} is used for obscuring the packet during its traversal by dynamically selected nodes along the packet route and also functions as a message authentication code for the packet. The source modifies h_i^{m+1} to encode payload information in the obfuscating partial hash where R is the rehash function and K_i is the key for node i as seen in the third field in Figure 2.3. In single-path routing, these obfuscating partial hash altering nodes are easily identifiable at the base station. For flooding, an approach is presented in Section 2.3.

2.2.2.2 Intermediate Packet Transformation

The intermediate packet transformation occurs at selected nodes on the path of the packet en route to the base station. It has 2 steps: Selection of rehash nodes, and packet transformation.

2.2.2.2.1 Selection of Rehash Nodes The random selection of rehashing nodes on the packet path is done using the rehash seed and a set of two functions: a one-way hash function and a mapping function. The functions are set in the node during pre-deployment bootstrapping, while the seed is read from the packet. This one-way hash function is different from the one used to generate the SrcID hash and the obfuscating partial hash. This method was used in [12] to generate random checkpoints and shown to be feasible to use in sensor networks.

For a node j, the one-way hash function is represented as $F_j(x)$, where x is the rehash

seed. The mapping function is $f_\rho(y)$, where ρ is the rehash probability and $y = F_j(x)$. The mapping function has a range of $\{0,\ 1\}$. For the range of output from the hash function, the mapping functions maps to 1 with probability ρ and to 0 with probability $1 - \rho$. Such a system of using two functions is beneficial because it gives the base station a simple way to change the rehash probability by just replacing the mapping function using a broadcast. When a node j receives a packet, it calculates $f_\rho(F_j(x))$, and if the result maps to value 1, it modifies the packet before forwarding to the next hop. If the mapping function maps to 0, it forwards the packet unmodified. The number of intermediate nodes selected for rehashing is set by the base station using the rehash probability parameter ρ. Additionally a rehashing node will hold the packet introducing a random delay of d time units. This delay forces the adversary to consider all the packets that passed through the node in the d time units while tracing back since the packet is reconstructed.

2.2.2.2.2 Packet Transformation The intermediate packet transformation occurs at nodes where the mapping function maps to a value of 1, as described in selection of rehash nodes. Here, the rehashing node j saves a copy of the packet before applying the transformation to it as follows:

1. SrcID hash: The value is replaced by a SrcID hash corresponding to rehashing node j.

2. OPH: $OPH_j = [R(OPH_{Packet})]_{K_j}$ where OPH_{Packet} is the obfuscating partial hash read from the packet, R is the rehash function and K_j is the key shared between the node j and base station.

3. Payload: Concatenate the SrcID hash received in packet to the payload and encrypt this with the symmetric key of node j. $Payload_j = [R(Payload_{Packet}|h_i^m)]_{K_j}$,

where $Payload_{Packet}$ is the payload of the received packet, h_i^m is the SrcID hash of the received packet. Additionally, the network can decide to concatenate only partial SrcID hash to the payload in an attempt to reduce the overhead. This can lead to collisions, which can be deciphered by the base station with the knowledge of network topology.

4. Payload Length: Length of the new payload.

5. Filler: Reduce the size of the filler to accommodate for the increased payload size and fill with garbage data.

2.2.2.3 Packet Reception and Verification

On receiving a valid packet presented in Figure 2.3, the base station uses a sliding window approach to look for the source-id corresponding to the SrcID hash within the sliding windows of the hash table maintained for each node.

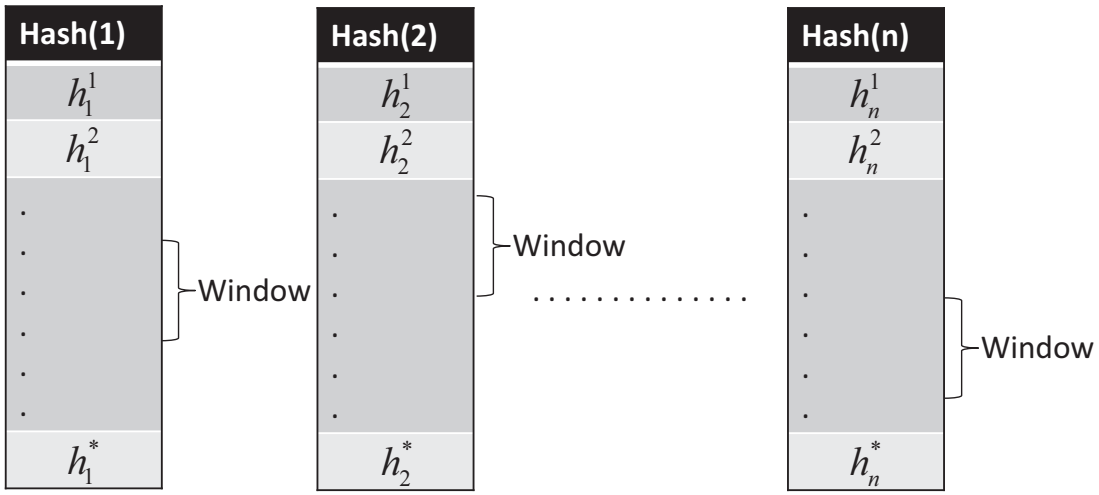


Figure 2.4: Hash Tables with Sliding Windows

A sliding window approach helps improve the search efficiency since the base station does not need to go through the whole hash table. This helps scale with the increasing

traffic. The length of the sliding window is governed by the packet loss rate in the network. The larger the packet loss rate, the larger this window is. When a valid packet is received, the starting point of the window is moved beyond SrcID hash in the table corresponding to the source. This is depicted in Figure 2.4. One optimization is to search only a subset of nodes for a possible match. With the knowledge of network topology, the base station can deduce the subset of nodes which could possibly be the previous rehashing node. This subset of nodes can again belong to only the downstream nodes, *i.e.* the nodes which should be propagating the packet to the current node and topologically lie further away from the base station compared to the node under consideration. It further reduces the subset of nodes for which reverse lookup needs to be performed thereby improving the efficiency in terms of time and computation required.

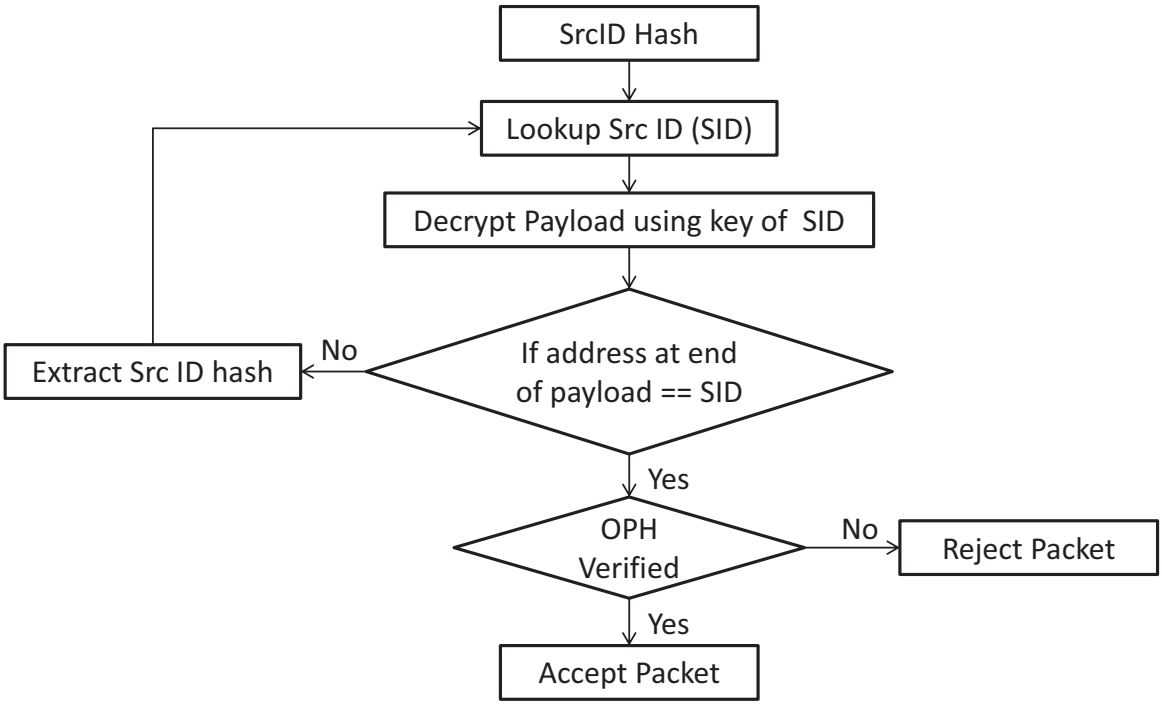


Figure 2.5: Packet Reception and Verification

The base station, on receiving the packet does a reverse lookup on the SrcID hash to identify the source of the packet. The assumption is that the packet has undergone at least

one rehash transformation during its traversal. It applies a recursive process as depicted in Figure 2.5, until it reaches the true source-id of the packet. During this process, the base station identifies the intermediate nodes responsible for rehashing the obfuscating partial hash. With this knowledge, the base station applies the rehashing process to the value of obfuscating partial hash starting with the hash value as encoded by the source. If the base station reaches the same value as received in the packet, this verifies the integrity of the packet received.

2.2.3 SPENA Example

Consider a twenty node network with one base station as depicted in Figure 2.6. Node 5 detects an event that it wants to report to the base station. From the hash chain generated by function $H_5()$, the source node has used $m-1$ entries for past communications. Hence, it generates an event detection packet with the structure depicted in Figure 2.7, where h_5^m is the SrcID hash and h_5^{m+1} is the obfuscating partial hash. The rehash seed value is set to 3, and the rehash probability (ρ) is 0.3.

The packet is transmitted to node 7, which reads the seed value (3), and calculates $f_{0.3}(F_7(3)))$. The result maps to 0, resulting in the packet to be forwarded unaltered to node 10. Node 10 repeats the process, and this time the rehash probability is true resulting in packet rehash. The rehash is performed using the pre-defined key shared between the base station and node 10. This takes the current OPH parameter as input and returns an encrypted hash of the same length as OPH parameter. It replaces the SrcID hash h_5^m by the SrcID hash of node 10 (h_{10}^m). Next, it calculates the new payload by concatenating h_5^m to payload ($[Payload|h_5^m]_{k_{10}}$) and encrypting with its key. The payload length field is set to 12 and the filler field size is reduced to 6 bytes. The filler is filled with garbage data.

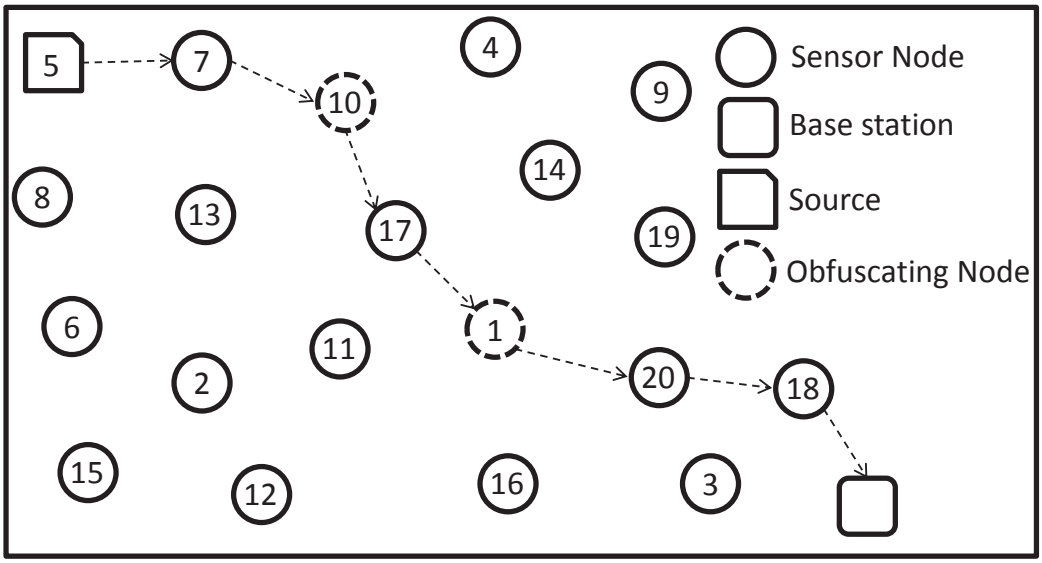


Figure 2.6: SPENA Network Example

DstID	h_5^m	$\left[R(h_5^{m+1} \mid Payload)\right]_{K_5}$	3	8	$\left[Payload \mid 5\right]_{K_5}$	Filler

Figure 2.7: SPENA Example Packet

Additionally, the rehashing node adds a random delay before forwarding the packet. This delay may result in a reordering of the packets in the queue resulting in further obfuscation and help prevent timing based attacks.

The packet is transmitted over multiple hops to the base station, with each intermediate node performing the actions as described. During the packet progression, node 20 is selected as a rehashing node while nodes 17, 1, and 18 are not. This transformation of the packet as it progresses through the path is depicted in Figure 2.8.

The base station on receiving the packet, performs a reverse lookup on the SrcID hash value (h_{20}^m), and identifies node 20 as the source. It decrypts the payload with the key for node 20 and compares the address field at the end of decrypted payload to 20. It does not match and the base station reads the SrcID hash from the end of the payload. Doing a reverse lookup, it identifies the source and repeats the process until it reaches the source-id

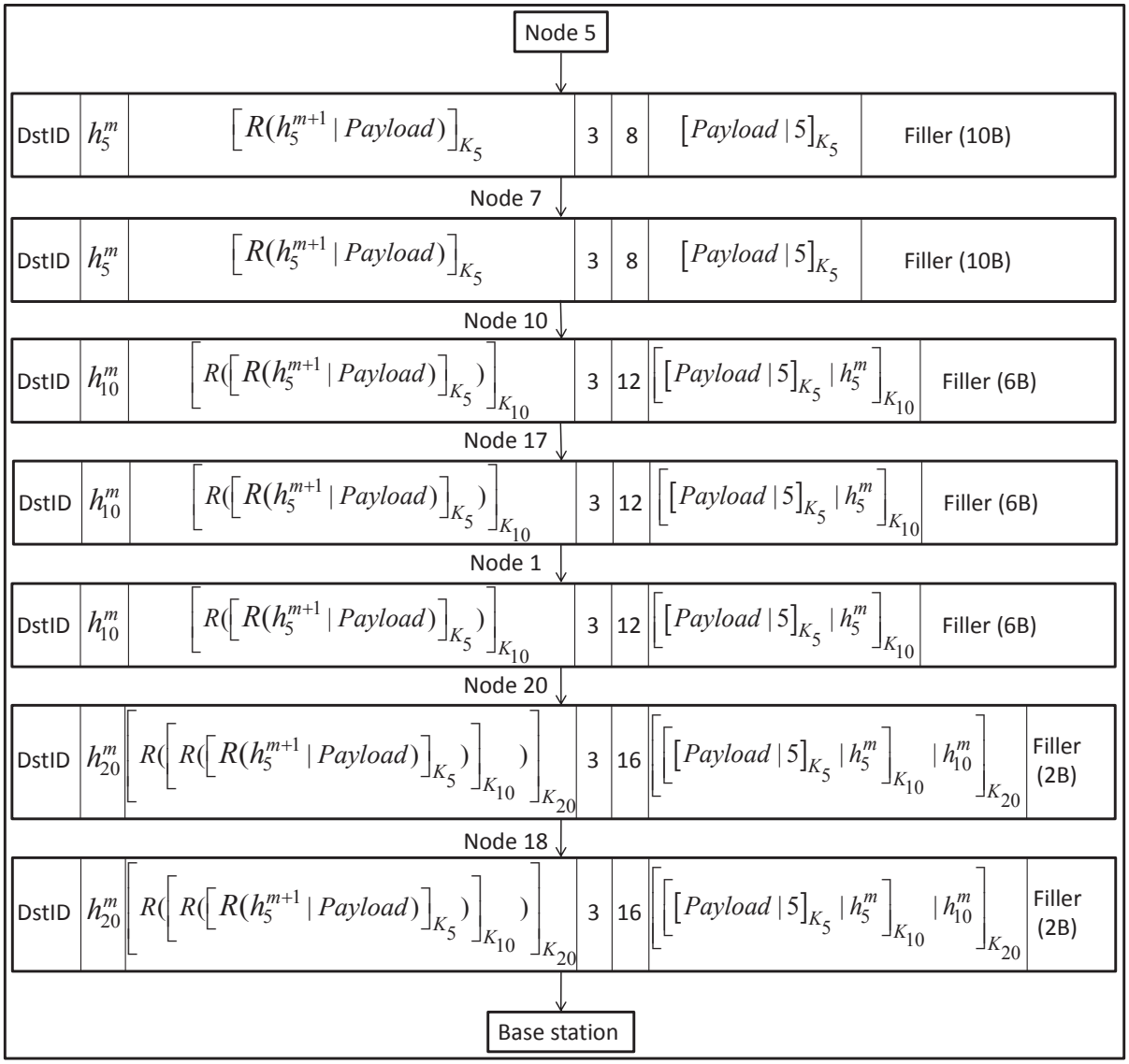


Figure 2.8: Packet Progression

information as depicted in Figure 2.5. Starting with the input given by the source OPH, the base station performs the rehashing process using the rehashing nodes hash functions as shown in Figure 2.9. The base station compares the final result with the obfuscating partial hash value received in the packet. If the two are equal, a valid packet is received.

If an invalid packet is received, the base station queries the intermediate rehashing nodes for the copy of the packet stored with them, given that the SrcID hash and the rehash seed are not modified. In single path routing, the intermediate node transmits the packet to a

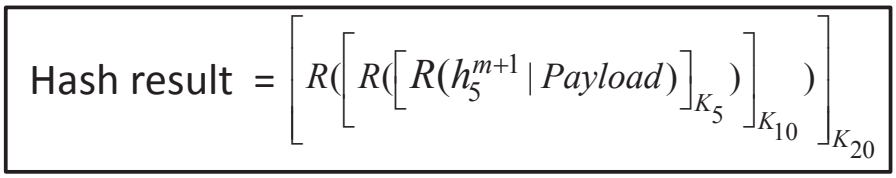


Figure 2.9: Hash Verification

different neighbor to forward the packet to the base station or can use a selective flooding scheme. Before transmitting the packet, the node modifies the packet to identify itself as the source, but includes the original SrcID hash for the base station to identify the actual source information. This is further elaborated in Section 2.3.

2.3 SPENA Protocol

In this section, the nuances of a working protocol is detailed and consider two routing schemes. The first scheme uses single path routing to forward packets from source to the base station. The other routing method considered is flooding, which happens to be the foundation for many routing protocols since it is easily manageable in sensor networks. Even the single path routing scheme uses flooding principles to respond to a verification query packet as discussed further.

The protocol requires three packet types, and the functionality of these packet types is the same for both 'single path routing' as well as 'flooding'. The difference is in the implementation of these packets and the header structure.

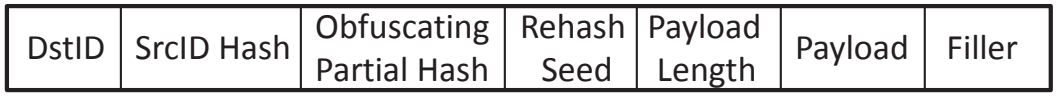
1. Event data packet (EDP): An event data packet is generated by a sensor node in response to an event detection.

2. Verification query packet (VQP): A verification query packet is generated by the base station to query the intermediate rehashing nodes on receipt of a corrupt packet.

3. Verification response packet (VRP): A verification response packet is generated by a rehashing intermediate node in response to a verification query packet sent by the base station.

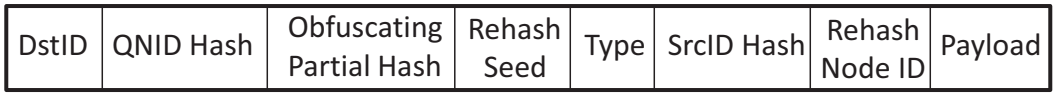
2.3.1 Single Path Routing

The EDP structure for single path routing (SPR) is different from the packet structure described in Section 2.2. In single path routing, the packets from a source node follow the same route to reach the base station. Hence, it is safe to assume that the base station is aware of the path followed by the packets. This makes it seem that the requirement of intermediate rehashing nodes encoding their SrcID hash information as unnecessary. But, one of the characteristics of SPENA is transformation of the packet en route to the base station requiring SrcID hash encoding. The packet structures for the SPR scheme are depicted in Figure 2.10.



(a) Event Data packet

(b) Verification Query Packet



(c) Verification Response Packet

Figure 2.10: SPENA Protocol Packet Structure

In VQP and VRP, the source-id from the original packet (SrcID Hash) is used as a packet-id. The SrcID field in VQP is the base station. Type is a packet type identifier. In response

to the VQP, the VRP is broadcast (selective flooding) back to the base station. A flooding based scheme is used for VRP to overcome the problem of the presence of compromised nodes on the single path routing route. If the VRP packet is transmitted back over the same single path routing route, there is a high probability that it could be corrupted by the same compromised node responsible for corrupting EDP.

The base station queries the intermediate rehashing node using the VQP. The DstID (destination id) is the id of the queried sensor node. The SrcID Hash is the source-id hash of the packet for which this query is run. Type identifies the packet to be of type VQP. In response to VQP, the node will respond with VRP. It creates a VRP packet with DstID as the base station and broadcasts it using selective flooding. The same process as applied to EDP is applied to VRP, but the rehashing nodes will include their hashed id in the packet similar to EDP packets in case of flooding. The 'QNID Hash' is the hash identifying the queried node at the base station. The 'obfuscating partial hash' is generated by the queried node and it has the same usage as the OPH in EDP. The structure of VQP and VRP is the same in both single path routing and flooding.

2.3.2 Flooding

Flooding is a technique in which a node broadcasts a packet and the neighboring nodes will repeat this process until the packet reaches the base station. The difference from single path routing is the lack of path knowledge at the base station when using flooding. To overcome this, the intermediate rehashing nodes on the packet path are required to encode their address into the packet header. It is shown in the next section that the overhead to achieve the same is minimal. In this scheme, the base station can recover by backtracking the packet information to the last non-malicious node provided the packet is not completely

corrupted. Under such a case, the base station can decide to query the neighborhood nodes for overheard packet transmission using the VQP. Even if the base station does not decide to recover the packet using VQP, it still knows the region of the presence of an adversary and can instruct the nodes to bypass the area while forwarding packets to it. Additionally, if it is a regional node compromise, in a flooding based approach the packet could have reached the base station using another path.

2.4 Analysis and Evaluation

The analysis for SPENA in terms of its robustness and evaluation of the protocol overhead while comparing it with other existing methods is presented.

2.4.1 Security Analysis

To check SPENA's ability to withstand different attacks, the adversary is provided with varying capabilities and we measure the effectiveness of SPENA against each form of the adversary. For this, we consider the adversary to be in super-local eavesdropper mode and then give it additional capabilities of node compromise in super-local eavesdropper stealth mode. The notations used are listed in Table 2.1.

Table 2.1: SPENA Table of Notations

Symbol	Definition
n	Number of nodes
d	Delay introduced by rehashing node
l	Average path length
e	Number of rehash nodes on path
c	Total compromised nodes
p	Packet generation rate
ρ	Rehash probability
v	Rehashing node address size

2.4.1.1 Super-local Eavesdropper

A super-local eavesdropping adversary can overhear packet communication over a large part of the network. The adversary can compare two packets as string length of bits, but it cannot decrypt the contents of the packet. Also, the adversary can compare packets (same packet over different hops) and identify if the packets are the same. When packet traversal occurs from source to the destination, the adversary can overhear the packet transmission and verify if the packet transmitted over different hops are the same. For example, in Figure 2.6, the adversary can overhear packet transmissions from node 10 to node 17, and from node 17 to node 1, and verify if it is the same packet.

In SPENA, a rehash node selection method dynamically selects the rehashing nodes on the packet path, and the number of selected nodes is determined by the rehash probability parameter (ρ). These selected nodes modify the packet structure by reconstructing the SrcID hash, OPH, payload, filler, and payload length fields while maintaining the same packet length. This results in the packet received and the packet forwarded by the node seem different to the eavesdropper. Hence, while tracing back, the adversary cannot follow the packet path without considering all the packets generated. It should be noted that in addition to the event packets, there will be other communication among the nodes resulting in generation of a large number of packets, which makes it costly for the adversary to perform detailed analyses on each hop. The rehash algorithm uses the rehash probability parameter ρ (defined by the rehash function and the rehash seed) and this determines the number of nodes selected for rehashing. Consider a network of size n with an average path length of l hops. Given the rehash probability parameter ρ, the number of intermediate nodes selected for rehashing is $l \cdot \rho$. The packet generation rate is p packets per node per unit time, giving

the total number of packets generated in the network in a unit time as $p \cdot n$. The number of packets generated over the period of time over l hops with each hop utilizing one unit time is $p \cdot n \cdot l$. For the duration of l hops, the number of packets generated, considering each rehashed packet as a different packet is $(n \cdot p \cdot l)(1 + \frac{\rho(l+1)}{2})$.

In the case without the usage of our scheme, the overheard packet transmissions across the nodes can be correlated by the adversary using simple bit comparison, since the packet is retransmitted without any alterations. Using this correlation, the packet can be traced back to the source and the identity of the source revealed. Also, this is independent of the number of packets generated. In SPENA, the packet gets altered en route to the base station, hence the adversary cannot apply a simple bit comparison to setup a correlation. Under the circumstances of doing an in depth packet analysis, the adversary will have to compare the packet with $(n \cdot p \cdot l)(1 + \frac{\rho(l+1)}{2})$ packets considering no additional delay is introduced by the rehashing nodes. For a network of 500 nodes, the number of packet comparisons required for varying path lengths and different rehash probability is presented in Figure 2.11. The packet generation rate is 0.1, and the rehash probability is 0.1.

If d is the random delay introduced, the average number of packets forwarded by each node is $d \cdot p \cdot l$. The adversary will now need to consider each of the source of $d \cdot p \cdot l$ packets as a potential sender of the packet. If it randomly selects a neighbor to be the sender, the probability of selecting the valid sender is $\frac{1}{q}$, where q is the number of neighbors forwarding packets. On selecting the valid neighbor, it needs to consider all the packets that were transmitted from the neighbor to the node in the last d time units *i.e.* $d \cdot p \cdot l$. This gives an overall probability of tracing the packet back as $(\frac{1}{d \cdot p \cdot l \cdot q})^{\rho \cdot l}$. The higher the delay, the larger is the set of packets required to be analyzed, but the drawback being the increased cumulative delay introduced in packet delivery at the base station.

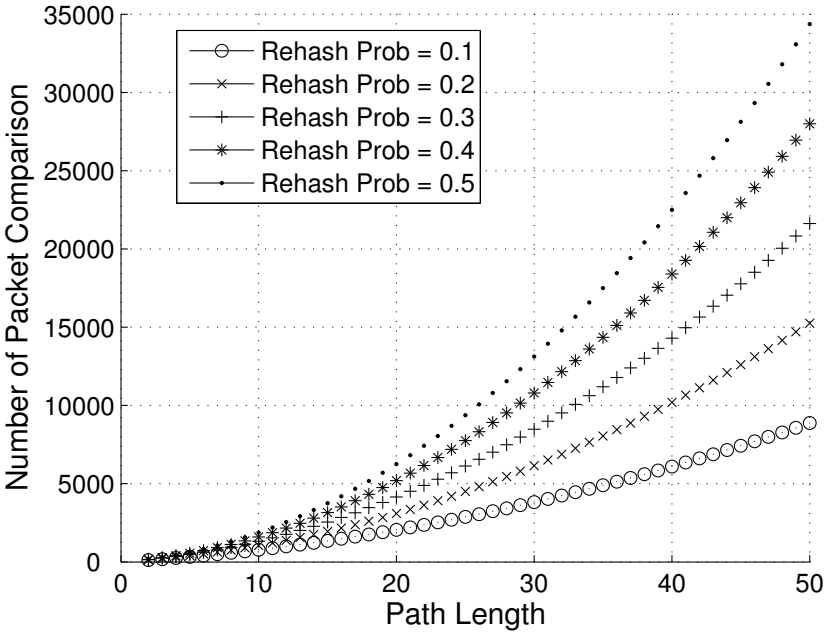


Figure 2.11: Packet Comparison

A network with a long path l or a big neighborhood is beneficial to our scheme. Hence, either a network with a long path length l or a dense network with a big neighborhood, which are both characteristics of a sensor network bodes well to our scheme. Also, high packet generation rate reduces the probability of a successful trace-back. In a random sensor network deployment, the factor under the control of the base station is the rehash probability, which can be increased to reduce the probability of a successful trace-back. It should be noted that the base station is significantly more powerful than a sensor node and with advent of technology, it can handle a very high packet rate even in a large network.

2.4.1.1.1 Impact of Rehash Probability Parameter (ρ) The higher value of ρ results in increased processing by the intermediate nodes. This can add to energy consumption. Additionally, it will have an impact on the length of address addition to the packet as each rehashing node encodes the SrcID hash from the packet received. However, the higher ρ

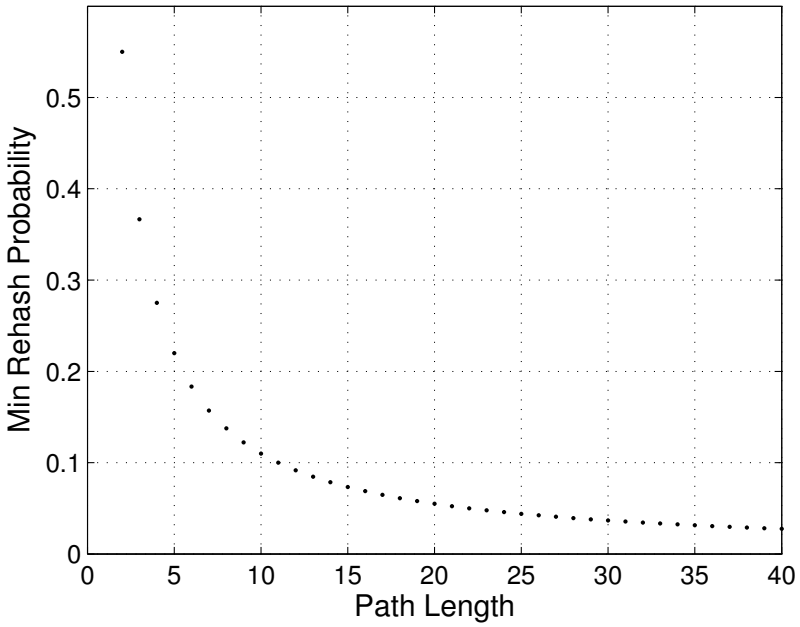


Figure 2.12: Minimum Rehash Probability

value leads to increased selection of intermediate rehashing nodes, thereby obfuscating the packet further. At least one node is required to be selected as a rehashing node, hence based on the value of l, there is a minimum requirement for ρ. Taking into consideration an error of 10%, the minimum ρ value for different path lengths is depicted in Figure 2.12. It should be noted that the probability of successful communication is not dependent on ρ, but on channel charateristics and is the same as in communication without rehashing.

2.4.1.2 Stealth Mode

In the stealth mode, the adversary is a powerful entity that can compromise a set of sensor nodes. We analyze if the access to cryptographic information due to node compromise allows an adversary to identify the original source of the packet when using SPENA. The adversary has access to all the functions and the keys at the node. On receipt of the packet at the compromised node, it can read the entire packet and scrutinize each component of the packet.

Hence, access to each component in the packet is first analyzed.

1. DstID: Readable and is available in plain text.

2. SrcID hash: This is a hash value generated by a one-way hash function, and unless the source itself, or the base station is compromised, this information is secure. Additionally, this field changes when it is updated to the SrcID hash of the selected rehashing node.

3. Obfuscating partial hash: This is again a hash value generated by the one-way hash function, which may have been rehashed en route to the compromised node. Access to this hash value in its true form (as encoded by the source) or the rehashed form does not leak any information about the source.

4. Rehash Seed: The seed value can be used by the adversary to identify if it needs to rehash the packet and has no correlation to the source generating the packet.

5. Payload Length: It identifies the length of the payload and does not leak any other information.

6. Payload: It is an encrypted field and without access to decryption keys does not disclose any information.

7. Filler: The filler is garbage data which is changed at each rehashing node.

A compromised node can belong to one of the three types, an intermediate node, an intermediate rehashing node, or a source node. The function of an intermediate node is to forward the packet as it is. An intermediate node compromise does not yield any additional information when compared to the information already gained by eavesdropping. When the

intermediate node is a rehashing node, the adversary gains the knowledge of the incoming packet and its transformed next hop. By itself, this does not provide any advantage, but when used with eavesdropping this becomes useful to the adversary. A compromise of all rehash nodes on the packet path will allow the adversary to trace-back the packet while performing eavesdropping. Access to the key and the functions at the compromised node will not allow the adversary to decode the passing packet since the payload is encrypted using the symmetric key of the source node and the source-id is hidden in the form of a hash value and can be mapped to the source node only by the base station. The last case, i.e., compromise of the source node will reveal the source information. It can be seen that, just compromising the nodes does not give the adversary much advantage unless the source node is compromised, but when done in addition to eavesdropping, it becomes a useful tool. So, further analysis of stealth mode is done in addition to the adversary having eavesdropping capability.

2.4.1.3 Super-local Eavesdropping and Stealth Mode

This is the most advanced form of attack in which the adversary eavesdrops over a large portion of the network with ability to compromise nodes and access the cryptographic information.

The method of selecting the compromised nodes is categorized into three types. In the first method, the compromised nodes are randomly selected by the adversary and they can be dispersed anywhere in the whole network. In the second case, the compromised nodes are bound to a geographical area and the nodes are assumed to be interconnected. This can occur if the adversary plans on compromising nodes by picking a neighboring node of the last compromised node. The second method can be assumed to be a special case of the

first method wherein all the randomly compromised node happen to be together. Third, the adversary can compromise all available sensor nodes within a given radius. This is the easiest scenario, but is less plausible, since the adversary may not apply its resources to perform such an attack unless it is absolutely certain of the origination of the packet in the geographical area.

2.4.1.3.1 Random Node Compromise In random node compromise, the adversary selects $r\%$ of nodes to be randomly compromised. Random node compromise is employed by the adversary when it lacks the ability to intelligently select nodes to compromise. Given the network of size n, the number of compromised nodes on a given packet route is $\frac{c \cdot l}{n}$, where c is the total number of compromised nodes given by $\frac{r \cdot n}{100}$. This gives us the number of compromised nodes on a path as $\frac{r \cdot l}{100}$. Let's analyze the worst case scenario when all the rehash nodes on a packet path are compromised, because, the only possibility for an adversary to do a successful trace-back is when all the rehash nodes are compromised. If the number of rehash nodes on the packet path is e, the probability of selecting all e rehash nodes is $\binom{c}{e}/\binom{n}{e}$. An example is depicted in Figure 2.13, which shows the probability of an adversary compromising all the rehash nodes under different network sizes given the number of compromised nodes is 25.

It is noted that as the number of nodes grows larger than a few hundred, the probability of all rehash nodes being compromised decreases significantly. For example, the probability of 3 rehash nodes compromised in a 400 node network when 25 randomly selected nodes are compromised is 0.00021. Also, with the increase in the number of rehash nodes, this probability value decreases further.

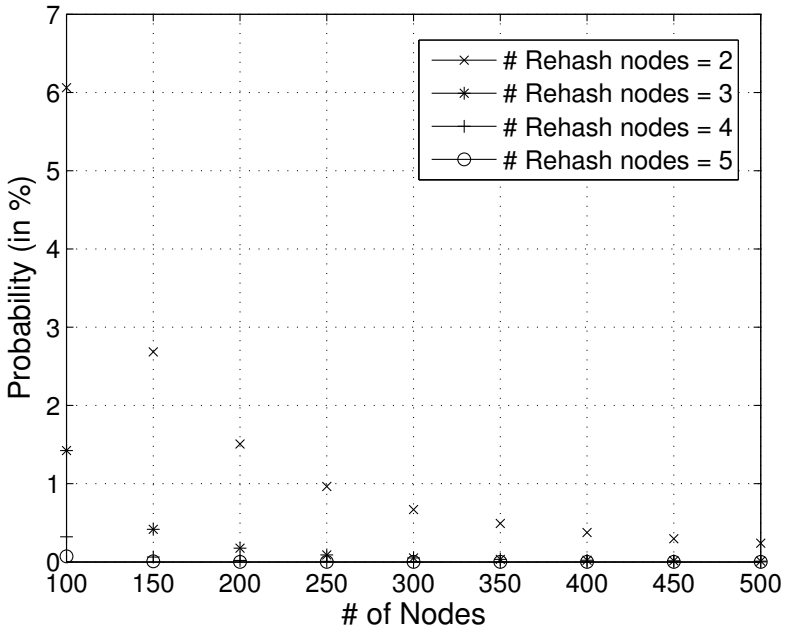


Figure 2.13: Probability of Compromising all Rehash Nodes

2.4.1.3.2 Compromise Neighboring Nodes The $r\%$ of nodes compromised belong to a neighborhood. By neighborhood, we imply that the nodes can all reach each other, albeit over multiple hops. When considering a path of length l, the worst case scenario occurs if all the c compromised nodes belong to the same path. We are interested in the cases where $c < l$, because if $c \geq l$, then the source itself is compromised and it is not possible to protect its privacy. With ρ as the rehash probability, the trace-back will be unsuccessful as long as $c < l - 2 \cdot \frac{1}{\rho}$. Figure 2.14 depicts the minimum number of consecutive nodes on a route that needs to be compromised for different path lengths with varying rehash probabilities.

A neighboring node compromise scenario is used by the adversary with the hope that sufficient consecutive nodes on a given path are compromised to reveal the source information. Another case in point is to compromise nodes surrounding an area so as to either track packets originating from the area (source tracking) or track packets to a particular destination.

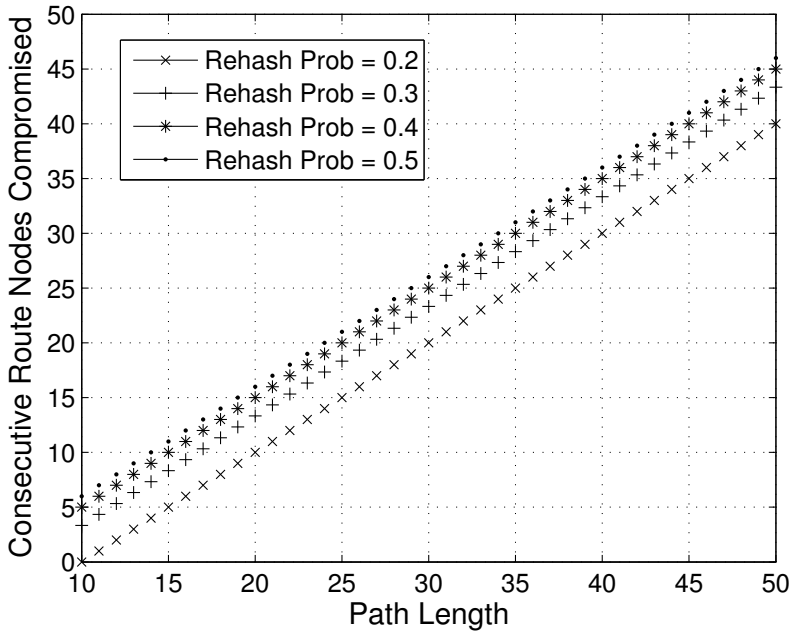


Figure 2.14: Consecutive Route Nodes Compromised

2.4.1.3.3 Compromised Nodes in a Geographical Area Compromising nodes in a geographical area is employed when the adversary estimates the presence of the source node in the area. To ease the analysis we assume the geographical area to be circular with an attack radius equal to transmission range t. The number of compromised nodes will vary with the attack radius. Let us consider a sensing range of $25m$. To have a unit coverage over 1000x1000 area, 509 nodes are needed. Simulation results of the average compromised nodes and maximum compromised nodes are presented in Table 2.2.

Table 2.2: Attack Distribution

Attack radius	Avg Compromised Node	Max Compromised Node
13	1.17	5
25	1.6	8
50	4.19	13

From Table 2.2, a very uneven distribution with the node compromise area being densely populated is required for such a scenario. If all the rehashing nodes on the packet path are

compromised, and if the adversary overhears over the whole network path, it will be able to trace the packet back to the source. Hence, there needs to be at least one rehashing node that is not compromised. The distance between the first and the last rehashing node on the path is $l - 2 \cdot \frac{1}{\rho}$. In the worst case scenario where all the compromised neighbor nodes are on the path, if $(l - 2 \cdot \frac{1}{\rho}) > a$ (where a is the longest path possible within the compromised geographical area), then the source information is secure, since at least one of the rehashing nodes is still un-compromised.

2.4.2 Overhead Analysis

It is accepted that any inclusion of security principles will incur additional overhead. It needs to be seen if the overhead incurred is within reasonable limits and whether the trade off against increased energy consumption and delay is justified. This analysis is performed only for the event data packets and not for the verification query packet and the verification response packets, since the verification packets are additional features used to recover a corrupted packet and not to provide source privacy.

The packet is assumed to be the standard 36 byte packet. Instead of the source-id, a hash generated by a one-way hash function with the SrcID hash equal to 4 bytes instead of the regular 2 bytes is used. The obfuscating partial hash can be either 1 or 2 bytes as dictated by the base station. This obfuscating partial hash can be any pre-decided part of the 4 byte hash generated by the one-way hash function. The rehash seed and the payload length together are 1 byte long. Hence, in SPENA, an event packet uses 4 additional bytes in a regular sensor packet of size 36 bytes.

2.4.2.1 Smaller Hash Usage and Collision

The size of SrcID hash is recommended to be 4 bytes long while the OPH can be smaller at 2 bytes long. The reason OPH can be short is because it is used in addition to the SrcID hash for packet validation purposes and doubles as a message authentication code. If a system requires to save on overhead and further reduce the SrcID hash, it can do so provided it satisfies the following condition. The base station is aware of the topology of the network to deduce collisions to identify the actual source. In SPENA, the rehashing node needs to be one of the nodes within the downstream (nodes which forward packet to the current node in order to be forwarded to the base station) neighborhood. So the first step for the base station is to discard any node which does not satisfy this condition. If the two colliding nodes satisfy the condition, the base station can decode the packet considering both sources and if it reaches a termination as presented in Figure 2.5 and also satisfies the condition of verification using the OPH, that source is accepted as the correct source.

SPENA also uses additional space for the intermediate nodes to program the SrcID hash information into the payload. For this purpose, a partial SrcID hash is used and not the full 4 bytes to conserve overhead and trade off a small percentage of accuracy, in the case where the partial hash can belong to more than one node. Under these circumstances, the base station makes an estimate of the rehashing node based on the geographical location. The space reserved for the address will depend on the number of rehashing intermediate nodes selected. Also, it should be noted that the payload in an event packet for an application like tracking for endangered birds is small, hence allowing the rest of the packet to be used for header. So, to improve the accuracy, the base station can decide on allowing the intermediate rehashing nodes to include a larger partial address hash. For applications requiring larger payload, the

data can be split and transmitted in multiple packets, thus preserving the accuracy (with larger hash sizes) but increasing the overhead. If v is the size of the intermediate rehashing node address size, the overhead is calculated as $\frac{4+v \cdot l \cdot \rho}{36}$.

The rehashing nodes introduce a random delay of d units. The intermediate node checking for rehash variability takes minuscule amount of time compared to the delay introduced. On a l hop path with rehash probability of ρ, the total time taken for the packet to traverse its path is $l(1 + \rho \cdot d)$. The delay parameter can be set by the base station based on the applications real-time requirements.

The storage of the hash chain entries in the sensor node consumes memory. For a standard IRIS mote with 128KB program flash memory (512KB measurement flash memory), to store 128 hash chain entries takes 512B of memory. To store up to 250 hash chain entries, it takes less than 1% of the program flash memory. Additionally, the number of hash chain entries to be stored in a node can be decided by the base station. A smaller number of entries will require the base station resetting the chain more frequently which consumes communication energy. Hence, based on the application, the base station needs to decide the chain length.

Next, the additional processing occurring at each node is considered. It is noted that transmitting $1Kb$ of data over $100m$ consumes the same amount of energy as executing 3 million instructions on a processor with $100MIPS/W$ power [3]. Kaps, *et al.* [4] measured the energy consumed to encrypt a packet using SHA-1 as $43.32nJ$, while the total energy to transmit a packet at maximum power is $93.3\mu J$, making transmission costlier by an order of 3. This makes the intermediate node processing energy consumption insignificant when compared to the communication cost incurred. Also, only a few nodes are involved in the packet reconstruction process, while the other nodes only use the rehash seed to check if they are selected as a rehashing node for this packet.

The number of address lookups and packet verification will put an additional load on the base station. Considering the base station to be running on a standard server configuration, having a quad-core CPU, with 8GB of RAM, it should be able to administer any amount of packets generated within practical limits. If the sensor network grows beyond a certain size, multiple base stations can be used, allowing implicit load balancing while also conserving the resources of the intermediate sensor nodes (reducing packet propagation overhead).

2.4.3 Comparison

SPENA is compared with destination controlled anonymous routing protocol for sensornets (DCARPS) [13], random walk [5], and fake event packet generation [6] [8]. DCARPS uses a label-switching method, while SPENA uses a hash chain and intermediate packet altering scheme to hide source traversal information. SPENA is lightweight and makes much less assumptions while considering a strong threat model. DCARPS requires the base station to have knowledge of the topology of the network, and our solution makes no such assumption. The base station creates a routing tree and routing paths in DCARPS and is relayed to the nodes. SPENA has no such requirements and has the advantage of being set to work with single path routing as well as flooding based protocols. In DCARPS, all the nodes on the path need to decrypt and re-encrypt the packet with a new label, while in SPENA, only a fraction of the nodes are required to reconstruct the packet before forwarding. Additionally, in SPENA, this reconstruction is done such that it allows the base station to validate the packet, whereas in DCARPS it is only used for routing. Node compromise attacks are not considered in DCARPS.

Comparing SPENA to technique involving random walk [5]. The length of the random walk determines the distance from the source to which a local eavesdropper can trace-back.

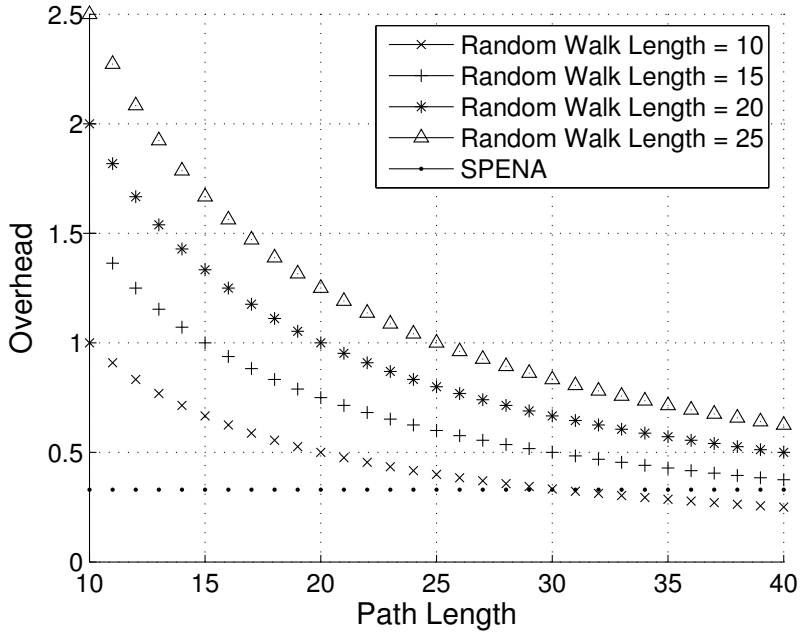


Figure 2.15: Random Walk Overhead

When you consider an eavesdropping adversary, a random walk scenario fails, since the eavesdropper can trace-back the packet to the source through the random walk. The overhead involved in random walk is higher than SPENA and is presented in Figure 2.15. The random walk length values of 10, 15, 20, and 25 are the number of random hops over which the packet propagates before moving towards the base station. Rehashing value is set to 0.1 for SPENA, and the filler is such that it compensates for the largest path size. For analysis, longest path length of 40 hops is assumed. Thus, the number of rehashing nodes is 4, requiring a filler of size 8 bytes to maintain a fixed packet size even after transformation. This gives us an overhead of 0.34, i.e., the total additional overhead required for SrcID hash, OPH, rehash seed and payload length, rehashing node address (total 12 bytes) over 36 bytes when the intermediate node incorporates a 2 byte hash address in the payload. We define privacy in random walk based methods as the length of distance the random walk takes the source information away from the actual source when compared to the path length. Figure

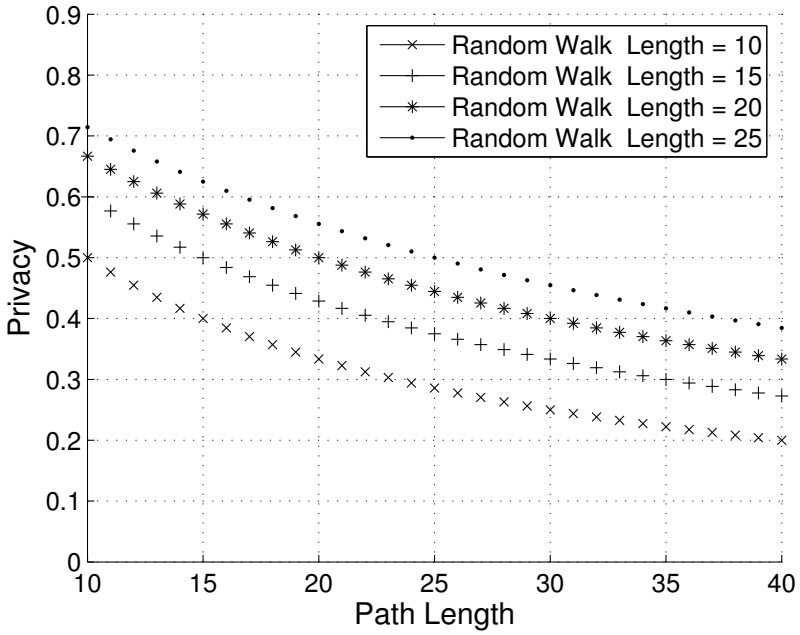


Figure 2.16: Random Walk Privacy

2.16 presents the privacy achieved for the same path lengths and random walk lengths as presented in Figure 2.15. The overhead is the resources (energy in our case) consumed due to propagation over the random path before the packet starts its journey towards the base station. As the overhead is reducing for random walk, the privacy achieved also reduces as in Figure 2.16 while the overhead of SPENA is constant irrespective of the path lengths.

The other methods using fake packet generation [6] [8] by nodes to emulate a packet introduce extremely large overhead. This is an obfuscation technique and is only successful under heavy load of fake packet generation. If one node detects an event and generates the event reporting packet, there needs to be at least one more source node generating a fake event reporting packet. Under these circumstances it is seen that, of all the traffic generated, 50% of the traffic corresponds to fake traffic. Furthermore, to increase security will incur more overhead in the form of fake packets. Hence, the valid functionality of such a system is dependent on the amount of fake packets generated and the scenario providing very minimal

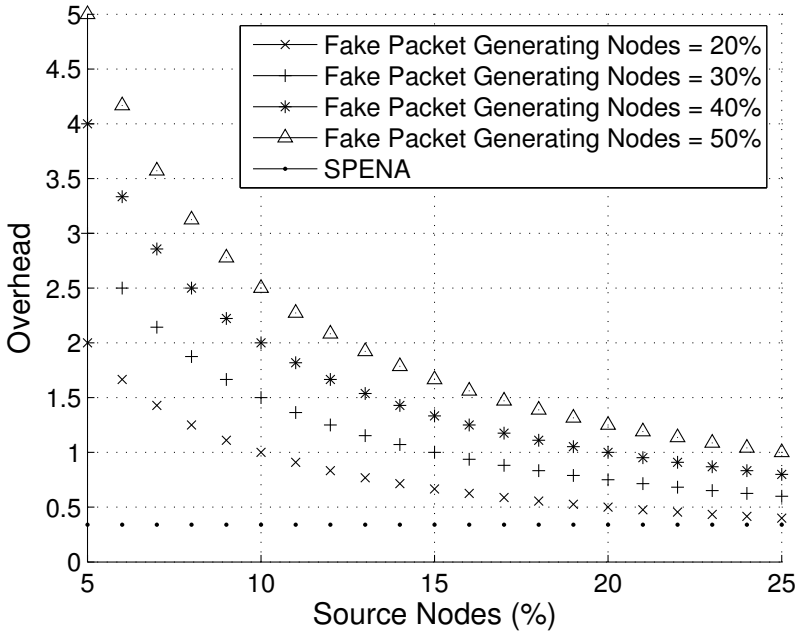


Figure 2.17: Fake Packet Generation Overhead

source privacy still has more overhead than SPENA. Additionally, the fake packet generation scheme does not consider node compromises.

Figure 2.17 presents the overhead incurred in fake packet generation compared to SPENA. Different numbers of fake packet generating nodes are considered, with the number ranging from 20% to 50% of all nodes. The 'source node (%)' on x-axis corresponds to the number of actual source nodes in the network as a percentage of all nodes. It is seen that any form of fake packet generation technique will have significantly higher overhead compared to SPENA. Even the conservative case of having just 20% of sensor nodes generating fake packets results in higher overhead compared to SPENA.

2.5 Related Work

There has been a lot of research to incorporate the security principles in sensor network. Protocols like TinySec [14], TinyPK [15], SERP [16] have been developed to provide security and to maintain integrity of data. In [17], Roosta, et al. presented a taxonomy of attacks on sensor networks and emphasized how traffic analysis attack can leak source information.

Privacy in sensor networks has been studied in detail with some researchers giving significance to hiding the base station while others providing source privacy. Kamat, et al. developed a phantom routing technique for flooding as well as single-path routing [5]. They were among the earlier groups to research source privacy and present multiple techniques to guarantee the same. First technique uses fake sources, with nodes sending fake event packets to confuse the adversary. The second technique called phantom routing, involves taking a random walk before forwarding the packet towards the base station in an attempt to increase the complexity of the adversary to backtrack to the source. Although the schemes are robust, they have a large overhead involved and may not withstand attacks under a collaborative adversary model.

In [6], Mehta, et al. similarly present two techniques; namely periodic collection and source simulation to wade off global eavesdropping attack. The source simulation technique is similar to the fake sources technique presented in [5]. In periodic collection, each node reports back to the base station periodically, irrespective of whether it detects an event or not. The weakness in case of periodic collection type technique is the latency incurred as well as overhead, while in source simulation, it is the overhead involved.

Wang et al. [10] present a privacy-aware parallel routing scheme to maximize the source trace-back time for adversary. The event packets from the same source are routed over

different paths to the base station. Additionally, a weighted random stride routing is proposed that breaks the entire routing into strides. Although it is a nice scheme, one of the restrictions is the requirement of knowledge of sensor locations to know the forwarding angle. Unless a method to deduce the angle is provided, it becomes a special case of random walk technique discussed by Kamat et al. [5]. Also, the parallel routing scheme will not be effective in protecting source privacy in case of a global eavesdropping adversary.

In [13], Nezhad, et al. present an exhaustive label switching based protocol to provide source and base station anonymity. One of the drawbacks is the requirement of the availability of global network information to the base station. With the knowledge of the topology of the network, the sink constructs a routing tree (sink as the root) with each link having a separate label. When a node receives a packet, it switches the label of the packet to the upstream link and the packet gets propagated. Other than the base station requiring global knowledge of the network, each node needs to perform detailed processing and reconstruction of the packets passing through them.

Shao, et al. present a statistically strong source anonymity scheme for sensor networks [8]. The authors use a exponentially distributed dummy traffic generation scheme called FitProbRate. The scheme differs from other similar studies such that the dummy traffic is not generated at a constant rate but at a dynamic rate decided by the Fitprob parameter. This scheme is a vast improvement over source simulation and fake sources, but again has the drawback of having overhead due to dummy packet generation even though it will not be as much as source simulation presented in [6].

All the studies discussed so far only consider a passive adversary. Most consider a local eavesdropping adversary while some provide solution to a global eavesdropping adversary. SPENA considers an eavesdropping adversary having node compromise capabilities.

A packet altering scheme is presented, which has lesser overhead compared to a source simulation or random walk scheme. Also, when compared to a label switching scheme like [13], SPENA does not need every node on the path to perform packet transformation and does not need the base station to be aware of the network topology.

2.6 Summary

Source privacy of sensor node generating an event packet is important as it can give away the event occurrence location. Existing techniques consider an eavesdropping adversary and refrain from providing source privacy solutions to an intrusive node compromise attack with eavesdropping. In this chapter, a hash-chain based source information hiding scheme is presented first. Second, a packet reconstruction strategy by dynamically selected intermediate nodes on the packet path that can protect source privacy under eavesdropping attacks while also tolerating node compromise attacks, is presented. Third, using the same one-way hash chain function, a packet verification method for the base station to validate a received packet is provided. Fourth, a packet recovery and node compromise area identification method is discussed in the form of SPENA protocol. Finally, the protocol is analyzed under different attack scenarios, and evaluated with regards to its efficiency in terms of overhead.

SPENA is compared to random walk based schemes, fake packet generation methods, and DCARPS which uses a label switching method. Random walk and fake packet generation methods have increasingly high overhead as the path length increases, whereas SPENA has a constant communication overhead irrespective of path length. This is because privacy in random walk based methods is dependent on the distance of the random walk. Fake packet generation schemes are obfuscation techniques which will be successful only under heavy

load of fake packets. Hence, SPENA is able to provide better source privacy than either of the two methods incurring a much lesser overhead.

When compared to DCARPS, SPENA does not require the base station to know the topology and only dynamically selected nodes reconstruct the packet in SPENA which reduces the computation overhead. In SPENA, the reconstruction of the packet has a second benefit as it is performed such that the base station can validate the packet, whereas in DCARPS it is only used for routing. The evaluation results exhibit the superior ability of SPENA to withstand the different attacks such as eavesdropping and node compromise while preserve source privacy with a modest overhead.

Chapter 3

Side Channel Attacks

Wireless sensor networks (WSN) are a special class of networks which primarily consist of a number of autonomous sensors to collaboratively monitor physical and environmental conditions. In Chapter 2 we saw why protecting the source privacy of nodes in a sensor network is important. The eavesdropping attack on source privacy was introduced which is also classified under side channel attacks.

The sensors, due to their small form-factor and limited physical shielding are vulnerable to attacks, in particular non-invasive attacks. Although it is possible for adversaries to launch invasive attacks once they obtain access to the sensor node, there are many reasons due to which the adversaries prefer non-invasive attacks. Some of these are listed below:

1. The adversary may not have direct physical access to the node, for example, the node is embedded within a concrete wall or beneath a road surface.

2. The node is built with tamper resistant material.

3. The adversary does not want his attack to be discovered, especially when sensor nodes are used in critical (such as military) applications.

4. The adversary may have access to only one node and would not want to damage it using invasive attacks.

This chapter considers side channel attacks, in which an adversary accesses a sensor

node in a non-invasive (i.e., non-tampering) manner and gains confidential information by observing the node under normal operation. In such attacks, the goal of the adversary is to deduce the inner workings of the hardware or the software. The adversary may use a variety of techniques such as power analysis (simple power analysis and differential power analysis), execution cycle frequency analysis, timing information (on data movement into and out of the CPU) analysis, electromagnetic leakage analysis, acoustic emission analysis, etc.

Previous research on side channel attacks has studied information leakage from monitors [18], keyboards [19–21], consumer mobile devices (such as PDAs, pagers) [22], IC chips [23], smart cards [24], etc. However, no previous study has focused on preventing side channel attacks on sensor nodes. Compared to other computing devices, sensor nodes are particularly vulnerable to side channel attacks due to two major reasons. First, sensor nodes are typically deployed in unsafe and unsupervised environments and could be easily accessible to adversaries. Second, sensor nodes are typically not encapsulated in tamper-proof bodies due to the demand on reducing manufacturing costs. Under these circumstances, the attacker may initiate attack on a part of the node by collecting leaked information that is available at close proximity. Thus, even a node that is compliant with FCC regulations may still be leaking sufficient information through its side channels for an adversary to obtain confidential information. As sensor nodes are increasingly being deployed in safety critical environments such as battle fields and power grids, preventing side channel attacks is indispensable in securing sensor networks. Because of the non-invasive nature of side channel attacks, they are often used together with other kinds of (possibly invasive) attacks.

The unique characteristics of sensor nodes make the prevention of side channel attacks more challenging. First, sensor nodes have low CPU power. A sensor node typically runs a 4-8MHz micro-controller. Second, sensor nodes have a small amount of memory. A sensor

node has RAM and flash memory in the order of few hundred Kilobytes. Third, sensor nodes have limited battery life. Some sensor nodes are powered by two AA batteries [25], and some are powered only by type-5 hearing aid batteries [26]. Fourth, sensor nodes have low bandwidth. They have a restrictive wireless bandwidth of 19.2 Kbps. This low bandwidth channel becomes even more important because of the hostile environment where the channel might have lossy characteristics. They generally have an integrated on board antenna with a transmission range of 100 - 300Ft. Fifth, sensor nodes are often deployed in unguarded and even hostile environments. Last but not least, most of their physical components are usually exposed. These physical constraints make previous solutions for preventing side channel attacks on resource rich devices unsuitable for sensor nodes, since most of the solutions increase the power consumption.

The chapter presents a comprehensive study of side channel attacks, by presenting an attack model, a proof of concept with experimental results, describe the family of side channel attacks on sensor devices and providing countermeasures. The rest of this chapter is organized as follows. The attack model is described in Section 3.1. In Section 3.2, the feasibility of a side channel attack by taking the electromagnetic attack as a special case is presented. A family of side channel attacks on sensor nodes is identified and countermeasures for each attack is presented in Section 3.3. Section 3.4 presents the related work. Finally the chapter is summarized in Section 3.5.

3.1 The Attack Model

In this attack model, it is assumed that an adversary can get close to a sensor node, but the physical access of the node is restricted. The adversary is a laptop class attacker. An

adversary may use specialized attack strategies for different components of a node. Depending on the type of node or the functionality of the node, an adversary can adapt its attack strategies. For example, if an adversary identifies the node as a gatherer of event information, which means the node needs to write the information into memory, the adversary may focus on obtaining information leakage during memory access cycles.

The adversary can use commercially available equipment such as Agilent E7401A EMC Analyzer, Agilent 11955A biconical antenna, or a log-periodic antenna. The biconical antenna is large in size and can measure up to 300MHz while the log-periodic antenna can measure up to 2GHz.

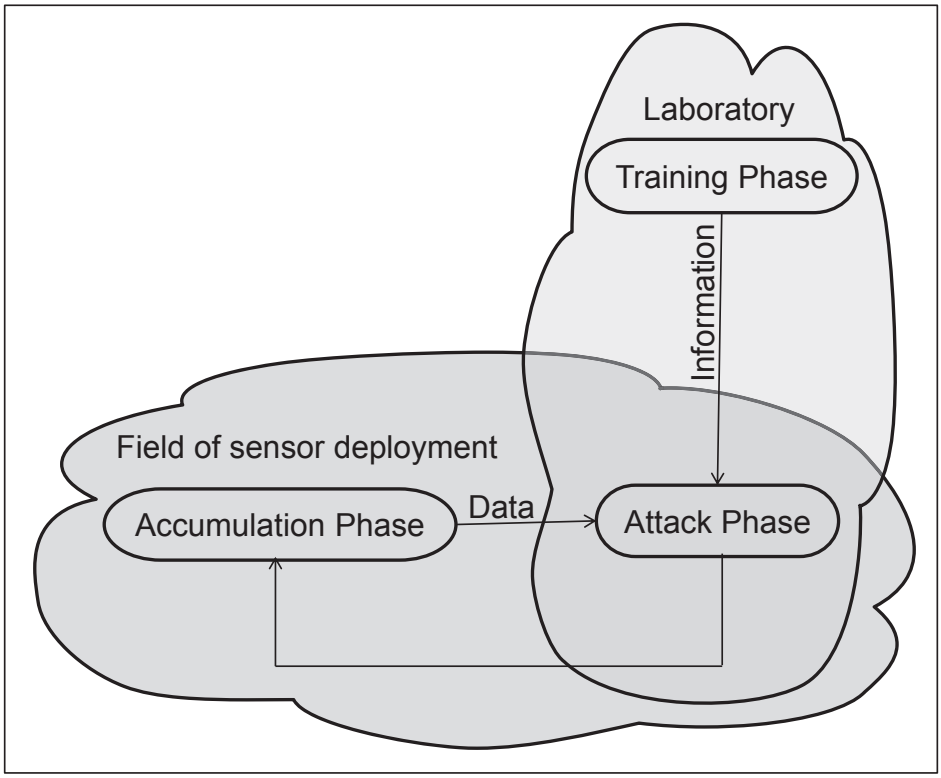


Figure 3.1: Attack Model

Our attack model consists of three phases as depicted in Figure 3.1. It follows the principles of a template attack [27].

1. **Training Phase**: Depending on the functionality of the sensor node from which the information leakage is studied, adversaries may train themselves by studying the information leakage from a similar sensor node performing the same functionality. This training process could iterate multiple times, thereby making it easy to decipher the information leaked. This phase can be accomplished by the adversary at its own convenience and does not need the adversary to be in the field of sensor deployment.

2. **Accumulation Phase**: In this phase, the adversary passively monitors the environment in which the node is deployed and in the due process gathers the data leaked.

3. **Attack Phase**: In this phase, the adversary uses the information learned in the training phase to decipher the valid information from the data gathered in the accumulation phase. The attack phase can be either integrated with the accumulation phase and implemented dynamically in the field; or it can be a stand alone phase, implemented in a static manner after the completion of the accumulation phase possibly in a more equipped environment.

3.2 Electromagnetic Leakage Attack- A case study

A proof of concept by demonstrating the feasibility of side channel attacks on sensor nodes is presented. For this purpose, electromagnetic leakage attacks are studied with commercially available equipment.

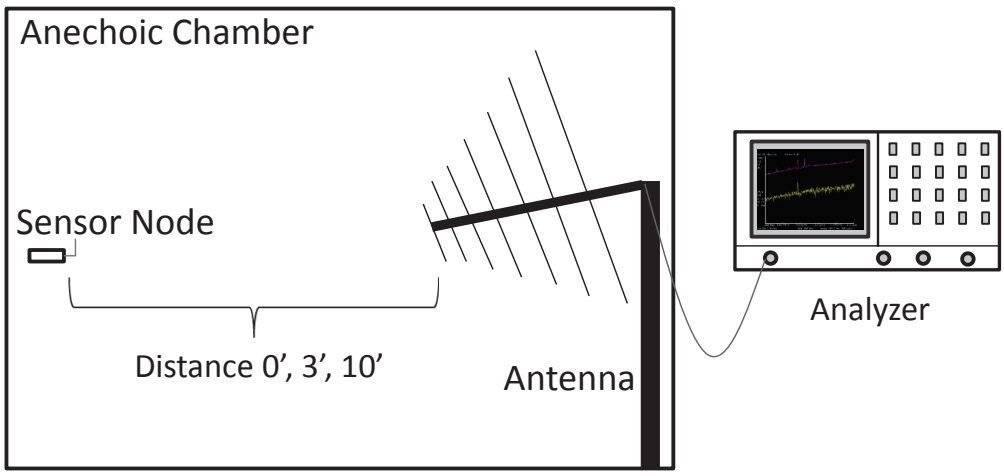


Figure 3.2: Experiment Setup

3.2.1 Experimental Setup

The experiments were conducted in a radio frequency anechoic chamber to suppress the electromagnetic wave analogy of echo's i.e. reflected electromagnetic waves and also reduce the interference of electromagnetic radiations from other devices. The setup depicted in Figure 3.2 is used for our measurements.

1. Tmote-sky sensor node: Program Space 48kB, RAM 10kB, Frequency 2400-2483 MHz. These types of sensor nodes are popular and commercially available.

2. Log-periodic antenna: It is used to capture electromagnetic emission up to 2GHz frequency.

3. Agilent E7401A EMC Analyzer: It is used to analyze the electromagnetic radiations captured by the antenna.

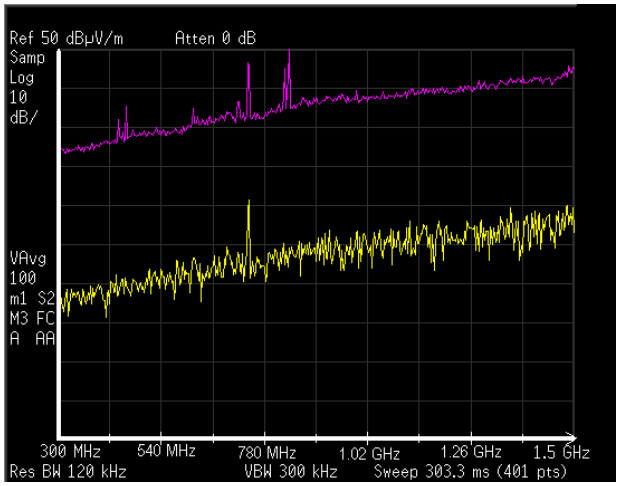
In the experiment there were two varying factors. The data the sensor node transmits and the distance of the antenna from the transmitting sensor node. The sensor node was programmed to send consecutive zeros or consecutive ones at different intervals of time. The

experiments were conducted by placing the antenna at different distances of 10 feet, 3 feet, and less than 1 feet (close to 0 feet) from the sensor node. It was decided to send zeros and ones because we wanted to study the electromagnetic radiation under fundamental but antipodal data. Different distances of measurement were selected to see if there was any prominent variance in the electromagnetic radiation strength over distance. This was important because of the low power attribute of the sensor nodes. Additionally, different distances helped us gauge if a physically inaccessible sensor node (for example, a node hidden behind a fence) can still leak information.

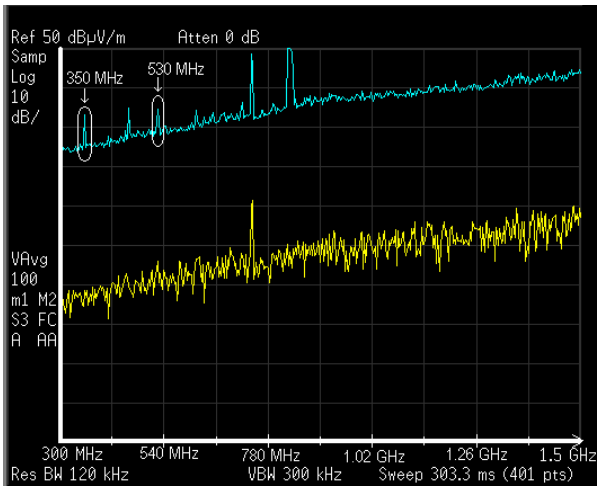
3.2.2 Observations

Upon studying, we made two important observations:

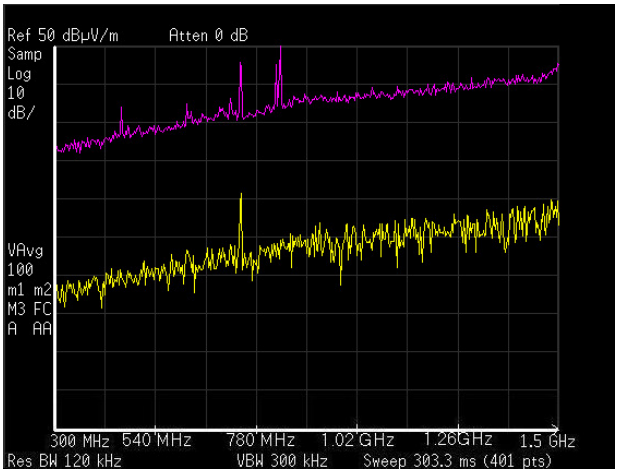
1. *EM radiation correspond to data*: Unique peaks were observed in the frequency readings that correspond to the input. These readings were consistent over multiple runs emphasizing the correlation between the input and the unique peaks. The bottom plots within the Figure 3.3 represents the idle state, while the top plot in Figures 3.3a and 3.3b represents the resultant electromagnetic radiations while transmitting data packets containing zeros and containing ones when the measurement distance is 0 foot i.e. the antenna was right next to the transmitting node. Similarly, the top plot in Figures 3.3c and 3.3d represents the electromagnetic radiation while transmitting data packets containing zeros and ones when the measurement distance is 10 feet. As depicted in the graphs, distinctive peaks are observed at 350MHz and 530MHz (marked in Figure 3.3b) while transmitting packet containing ones that are absent while transmitting packets containing only zeros. This leads to the conclusion that there exists



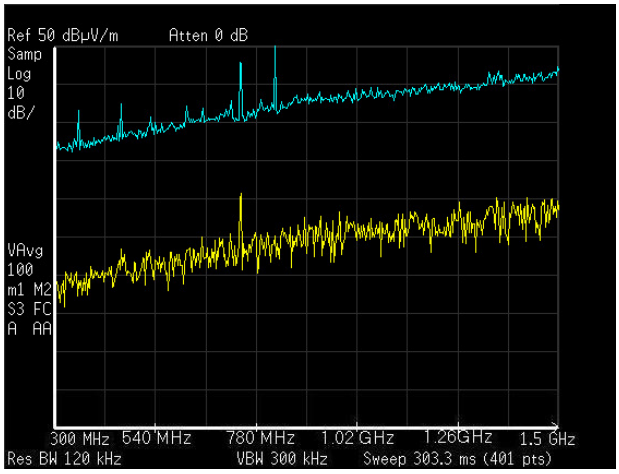
(a) Transmit 0's, Distance 0 feet



(b) Transmit 1's, Distance 0 feet



(c) Transmit 0's, Distance 10 feet



(d) Transmit 1's, Distance 10 feet

Figure 3.3: EM Radiation while Transmitting 0's and 1's. 'For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this dissertation'

a strong correlation between the electromagnetic radiation and the data processed in the sensor.

2. *Non-degradation of signal strength over distance*: There was no significant impact of distance over signal strength. This observation was critical because under certain circumstances, the adversary may not get to be in close proximity with the sensor node. Consequently, we only present results from two sets of observations depicted in

Figure 3.3. It is noticed the use of an anechoic chamber highly reduced the ambient noise. An anechoic chamber replicates a rural or forest environment characteristics wherein there is not much interference from other devices.

3.3 Taxonomy of Attacks and Countermeasures

In this section, the different types of side channel attacks on sensor nodes is described and the countermeasures for each attack type proposed. The goal of these countermeasures is not to fully prevent side channel attacks, but to make the attacks practically infeasible. We begin by providing two general countermeasures which will be applicable to multiple side channel attacks, and they may be used in conjunction with other attack-specific countermeasures followed by the taxonomy of attacks and their specific countermeasures.

3.3.1 General Countermeasures

The two general countermeasures are obfuscation (code obfuscation, process obfuscation) and tamper-proofing.

3.3.1.1 Obfuscation

Obfuscation techniques are used to conceal the meaning of a certain computation by making it obscure and harder to understand. Code obfuscation [28] has been proposed previously as a technique to limit side channel information leakage. In this chapter, a new obfuscation technique, called *process obfuscation*, to circumvent side channel attacks is proposed.

3.3.1.1.1 Code Obfuscation Code obfuscation is a process that has been studied well previously [28] and it can be described as follows. Suppose that (part of) an algorithm

consists of a loop where the execution of a given set of instructions depends on certain input values. If, from some side channel information (e.g., timing, power consumption etc.), one can distinguish which set of instructions is processed and retrieve some secret data (if any) involved during the course of the algorithm. To prevent this from happening, the code can be created such that the different paths of the execution cycle take the same amount of time or the same execution path over different execution cycles takes different amount of times. This can be achieved by using fake instructions in the code so as to either maintain the execution time over different paths or to randomly differ the execution time over the same path. Code obfuscation techniques for prevention of side channel attacks has been classified as follows:

1. Data Independent Calculations: The basic idea behind this technique is to make the amount of time for performing operations independent of the data being processed.

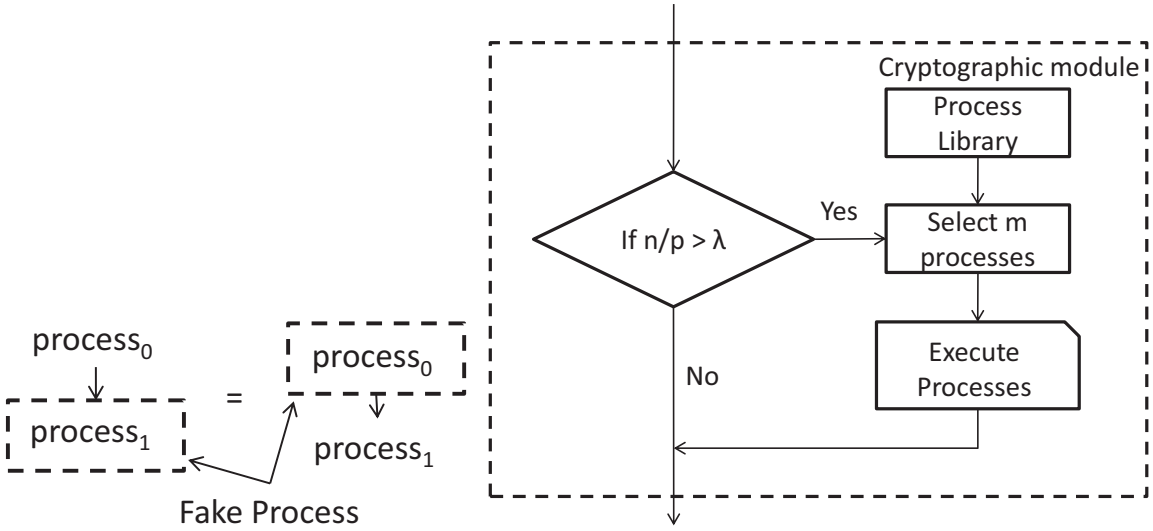
2. Blinding: Blinding is a technique in which the content of the message is disguised before it is signed. This technique is inspired from blinding signatures [29], which was further elaborated by Kocher [30]. It is especially useful for preventing electromagnetic side channel attacks.

3. Avoid conditional branching: This technique avoids conditional branching wherever possible. In the case where a conditional branch is required, take precautions to enforce the same execution time in each branch. This is further elaborated under timing attacks in Section 3.3.2.5.

If the sensor node is capable of simultaneously possessing multiple encryption algorithms (depending on memory/space availability), it can randomly select the encryption algorithm

it uses. In doing so, it will be required to make the side channel characteristics like power consumption, timing etc. of the same order. Otherwise, it may defeat the purpose of having such an implementation.

3.3.1.1.2 Process Obfuscation Process obfuscation is a technique in which reactive decisions are taken during runtime to mask the process being executed. For an example as shown in Figure 3.4a, process obfuscation can be attained by executing $process_0$ followed by a fake execution of $process_1$ when $process_0$ must be executed, and by executing a fake execution of $process_0$ followed by $process_1$ when $process_1$ must be executed. Obfuscating a process can be achieved in two ways, namely a proactive methodology which involves obfuscation of each and every process and a reactive methodology which obfuscates only those processes that can end up leaking information through the side channel. Given the resource constraints of a sensor node, the proactive approach is not suitable and its recommended to use the reactive approach.



(a) Process Obfuscation (b) Process Obfuscation Example

Figure 3.4: Obfuscation

Next, a simple methodology is discussed which can attain process obfuscation while consuming just the minimum resources. The reactive method will base on the past transmissions or computations. If the past n computations had a recurring pattern p and the frequency of recurrence of the pattern p in n is greater than the leakage threshold λ, it will need to take a deviation from the actual processing and perform obfuscation. Unless some countermeasure is taken, the recurring pattern will leak information. The value of λ can be varied and will be based on the sensitivity of data under consideration. The pattern can be simple computations such as consecutive bit shift operations.

A library of simple generic processes is maintained which can be summoned by the processor. This set of processes will be of different complexity in terms of execution time, computation power, memory accesses required. It should be noted that the larger and more varied the library the better the obfuscation. Figure 3.4b represents a simple flow of process obfuscation. The process obfuscation happens in three steps. First, the condition of recurring pattern is checked to see if its required to add obfuscation. This is done by the obfuscater module. In the second step a set of m processes are randomly selected from the process library. The value of m can range from 1 to infinity. In the third step, the m processes selected are executed and the control returns back. Process obfuscation is done only in the cryptographic module to limit cost. When the control shifts to process execution in step 3, there should not be noticeable delay. For this purpose the processes are located in fast access memory to minimize load times.

3.3.1.2 Tamper-proofing

A sensor node in a tamper-proof body ensures that any attempt to break the body results in all confidential information in the node being destroyed. The main hindering factor of

building sensor nodes in tamper-proof bodies is cost. Indeed, there are many advantages of tamper-proofing other than obstructing side channel attacks. For example, a tamper-proof body could prevent wear and tear due to environmental factors, thereby increasing the overall life of the node assuming the availability of sufficient power. In practice, the encapsulating body of a sensor node may not need to be tamper resistant for preventing side channel attacks. A simple cover over the sensor node sometimes suffices and reduces the strength of certain side channel leakages, while restricting access to internal components.

3.3.2 Taxonomy of Attacks

Side channel attacks on sensor nodes are classified into eight categories, namely power analysis attacks, electromagnetic leakage attacks, optical side channel attacks, traffic analysis attacks, timing attacks, fault analysis attacks, acoustic attacks, and thermal imaging attacks. For each type of attack, the corresponding countermeasures are presented.

3.3.2.1 Power Analysis Attacks

In power analysis attacks, an adversary studies the power consumption of devices, especially the cryptographic modules. Power analysis attacks require close proximity to a sensor node, so that an adversary can measure the power consumption of the sensor node. Since it requires very close proximity to the sensor node, it can also be classified in the semi-invasive class of side channel attacks. Power analysis attacks were first studied by Kocher et al. [31]. There are two types of power analysis, namely simple power analysis (SPA) and differential power analysis (DPA). In simple power analysis, the adversary studies the power traces visually to interpret type of activity. For example, SPA can distinguish power consumption across conditional branches where the difference is significant. If the difference is insignificant,

simple power analysis attack will be unsuccessful. In differential power analysis, the adversary studies the power traces and is able to apply mathematical and statistical principles to determine the intermediate values.

3.3.2.1.1 Countermeasures The countermeasures are implemented by either preventing or complicating power analysis attacks. Prevention of power analysis attack can be easily achieved by introducing a tamper resistant body. This increases the one time cost of the sensor node, but will allow the node to conserve the power usage when compared with other countermeasures. Replacing the power source when the battery runs out will be an inherent issue with having a tamper resistant body. Complicating the power analysis attacks can be achieved by several strategies, which are as follows:

1. Power randomization: Power randomization is a technique in which a hardware module is built into the chip that adds noise to the power consumption [32]. This countermeasure is simple and easy to implement, but is not energy efficient. Also, it adds to the fabrication cost of the device.

2. Novel Circuit designs: In [33], Wagner et al. gave solutions to design hardware circuits that mask the changes in power consumption so as to increase the cost of a power analysis attack.

3. Obfuscation: Obfuscation is a good solution to prevent SPA, but is susceptible to DPA [31].

3.3.2.2 Electromagnetic Leakage Attacks

Electromagnetic radiations are emitted and propagate following Maxwell's equations.

$$\boxed{\begin{aligned} \nabla \cdot D &= \ell_f \\ \nabla \cdot B &= 0 \\ \nabla \times E &= -\frac{\partial B}{\partial t} \\ \nabla \times H &= J_f + \frac{\partial B}{\partial t} \end{aligned}} \tag{3.1}$$

The electromagnetic (EM) radiations attain importance when they are hardware generated emissions, especially emissions from the cryptographic module. Electromagnetic leakage attacks have been shown to be more successful than power analysis attacks on chip-cards [34]. In the case of sensor nodes, the circuitry is exposed and hence leads to stronger emanations of EM radiations.

Similar to power analysis, electromagnetic analysis can be performed at two levels of sophistication, namely a simple analysis called simple electromagnetic analysis (SEMA) and a differential analysis called DEMA. Furthermore, the emanations can be either direct emanations from a component or a mixture of emanations from a group of components. There has been studies considering both circumstances and contrary to the thought, the multiplicity of signals has been found to be useful [34]. Experiments were performed to study EM emanations from sensor nodes, which was discussed in Section 3.2.

3.3.2.2.1 Countermeasures Unlike power analysis attacks, electromagnetic leakage attacks can be completely non-invasive while being equally effective. The basic countermeasure to prevent electromagnetic information leakage is to encompass the node with a casing so as to prevent access to individual components in a sensor node. The countermeasures as follows:

1. Secret Shares: Secret shares is a technique in which the original computation is divided

probabilistically such that the power subset of shares is statistically independent. The use of secret shares was proposed by Goubin et al. [35]. One drawback is increased power consumption.

2. Masking: The next countermeasure is to use masking methods. Masking is a scheme in which the intermediate variable is not dependent on an easily accessible subset of secret key [36]. This results in making it impossible to deduce the secret key with partial information gathered through EM leakage.

3.3.2.3 Optical Side Channel Attacks

The intensity of light emissions from a monitor or liquid crystal display could be used to study the contents of the last displayed screen. Given the form-factor, optical side channel attacks on sensor nodes are formulated differently from the attacks on devices that use a visual display to output information. The sensor nodes have light emitting diodes (LED), which have two primary purposes. The first purpose is in debugging the application program while programming the node. The second use of the LED is for the purpose of signaling. LEDs are externally visible to both a user as well as an adversary, unless the node is used for an application in which they are not in the line of sight.

There are often multiple sets of LEDs on a sensor node having different functionality. These LEDs can leak valuable information. For example, the adversary can program his attacking station to listen to the channel only when there is an optical signal from the node, thereby conserving a lot of its (adversary's) battery. Another case is during node re-programming by the base station. The adversary is alerted about the occurrence of changes to the node by just observing the optical side channel.

3.3.2.3.1 Countermeasures First, its required to prevent LEDs from leaking any confidential information by the way they light up. Any signaling information, which can be used by legitimate users for their application, can also be used by adversaries with malicious intent. Second, sensor node programmers should remove the debugging information for which the programmer gets a feedback from the LED, before sending it to production. This is an important requirement because in the event of a malfunction, the execution of the debug code may result in information leakage from the optical side channel.

3.3.2.4 Traffic Analysis Attacks

Traffic analysis attacks are attacks that analyze traffic flow to gather topological information. This traffic flow could divulge information about critical nodes, such as the aggregator node in a sensor network. Such attacks primarily relate to the intermittent transmissions that are inherent in sensor networks. Due to the limited energy capacity of nodes and the fact that the transceiver component of a node consumes the most power, the nodes in a sensor network limit the use of the transceiver to transmit or receive information either at a regulated time interval or only when an event has been detected. This generally results in an architecture comprising some aggregator nodes (also called supersensor or actor) within a sensor network. Aggregator nodes are the sensor nodes whose primary purpose is to relay transmissions from nodes to the base station in an efficient manner, instead of monitoring events like a normal node.

The added functionality of acting as a hub for information gathering and preprocessing before relaying makes aggregator nodes an attractive target to side channel attacks. If a node is frequently in active states (instead of idle states), there is high probability that the node is an aggregator node. Such leakage of information is highly undesirable because the

leaked information could be strategically used by adversaries in the accumulation phase of an attack.

Another emerging security concern with regards to traffic analysis attacks involves around guarding the source privacy. By performing a traffic analysis, the adversary can identify the source of event leading to location where the event occurred. This was described in detail in Chapter 2.

3.3.2.4.1 Countermeasures It is practically inefficient to prevent adversaries from identifying aggregator nodes because camouflaging traffic in sensor networks is power intensive. Consequently, the focus is on preventing adversaries from identifying valid aggregation cycles of aggregator nodes. One solution to counter such attacks is to have each aggregator node execute dummy operations that resemble the average power consumption curve observed during the normal operation of the aggregator node. This additional requirement on aggregator nodes, despite the fact that it would result in additional power consumption, can be met, because unlike ordinary nodes in a sensor network, aggregator nodes are typically equipped with longer battery lives owing to their integral nature and functionality with the sensor network.

Apart from simulating the power consumption of a genuine process execution, the two necessities that the execution of the dummy process must incorporate to be successful in thwarting the accumulation phase are to use a different dummy execution process each time or have a low repetition rate. This should help prevent the attacker from finding a pattern that would differentiate the execution of a dummy process from the normal execution of an aggregator node. The second requirement relates to the timing of the execution of the dummy process. Depending on whether there is a pattern to the timing of the execution of

a dummy process, an adversary may be able to identify and disregard the dummy process. For example, if an adversary is capable of identifying the presence or absence of a radio transmission, the attacker can disregard any power consumption curve computed during the absence of transmission signal. Similarly, if the dummy process is not executed every time the aggregator node receives a transmission, the attacker will be able to identify invalid transmission. Hence, to ensure the effectiveness of this scheme, the dummy process must be executed each time the aggregator receives a transmission as well as randomly during idle periods. The advantage of incorporating dummy processes in an aggregator is to minimize the ease of identifying transmission flow in a sensor network that can be used to identify the base station of the sensor network, which could be highly confidential in critical applications (such as military applications).

3.3.2.5 Timing Attacks

The timing attack involves exploiting the variance in execution time for different branches in the cryptosystem. This becomes all the more important in sensor networks subject to the slower processors used in the nodes. The slower processors will enhance even small difference in computation time over different branches. Timing attacks could also study the number of memory accesses and the time variance in doing the same.

3.3.2.5.1 Countermeasures One of the most relevant countermeasure for such attacks is using more clock cycles such that branching does not effect the execution time. Also, the memory access times should be standardized to be the same over all accesses. Since time as a resource is available in abundance in a sensor network (as compared to power), slowing down the access times by adding sufficient delay to normalize the access times should be the

priority.

3.3.2.6 Fault Analysis Attacks

Fault analysis attacks (also called fault induction attacks) are attacks in which useful information gets leaked out due to occurrence of fault in the cryptosystem. The faults may occur naturally or be induced by adversaries. Adversaries may inject faults in two ways. One is to give programs invalid input, which may cause buffer overflows. The other is to use equipment such as a laser pointer to illuminate SRAM (EPROM and EEPROM can also be modified) to flip some bits in memory [37]. Fault analysis attacks have been discussed in detail by Biham and Shamir [38]. The importance of checking cryptographic protocols for faults was detailed by Boney et al. [39].

3.3.2.6.1 Countermeasures To counter fault analysis attacks, use redundancy to catch injected faults. The idea is similar to N-version programming by Avizienis et. al. For certain critical function, it is recommended to deploy multiple implementations of the same function. Given an input, process it using the various different implementations and compare the outputs. A selection module could be incorporated to decide the valid output. The selection could be as simple as: accept the result only if $\tau\%$ of redundant modules have the same result output. Although sensor nodes have limited resources, critical regions usually comprise the crypt functions, which need to be secured.

3.3.2.7 Acoustic Attacks

There are two types of acoustic emissions that have been studied previously: acoustic emissions from keyboards [19, 20] and acoustic emissions from computing components such as

CPU and memory [40]. Acoustic emissions are produced by a keyboard when different keys are pressed and can be used to identify the keys being pressed with extra triangulation information. Acoustic emissions from computing components have been demonstrated by Shamir et al. to be exploitable [40]. As sensor nodes typically do not have keyboards, the acoustic emission from its exposed computing components is the real concern.

3.3.2.7.1 Countermeasures One practical and effective countermeasure for acoustic attacks is to encapsulate a sensor node in sound absorbing material. Another countermeasure is to introduce random acoustic noise of similar frequency to obfuscate acoustic emissions from sensor nodes.

3.3.2.8 Thermal Imaging Attacks

Thermal imaging attacks differ from acoustic attacks in that the emission being exploited is heat instead of sound. Such attacks often exploit the infrared images emanating from CPUs.

3.3.2.8.1 Countermeasures To counter thermal imaging attacks, one approach is to use a dual layered case with the inner layer a highly conducting surface and the outer layer made of a non-conducting material. When heat is generated from internal computing components, the inner, highly conducting surface will quickly dissipate the heat around. The outer layer prevents accesses to the temporary hot spots formed on the inner layer, thereby preventing any information leaked in the form of heat.

3.4 Related Work

Roosta et al. was the first to briefly discuss the possibility of side channel attacks (also referred to as tempest attacks) in sensor networks [17]. The study of side channel attack dates way back to early twentieth century when R.E. Priestley wrote about it in [41]. Okeya et al. demonstrated the feasibility of side channel attacks by passively monitoring the power consumption waveform of IC chips using an oscilloscope [42]. They presented differential power analysis attacks by selective forgery of MAC and demonstrated how several key bits may be extracted. Similarly, Kocher et al. showed that monitoring the electrical power consumption of a smart card running the DES algorithm was sufficient to retrieve the secret key [31]. Gratzer et al. successfully demonstrated a side channel attack with neither the knowledge of any information about the algorithms nor the microprocessor power consumption model [43].

Cryptographic techniques to counter side channel attacks on smart cards and IC chips have been proposed by Micali et al. in physically observable cryptography [44]. Alternately, theoretical approaches like building private circuits in a boolean context have been proposed to counter side channel attacks. Using this approach, any n-gate circuit that is allowed to leak up to 't' bits at a time, can be converted to a perfectly secure circuit of size $O(nt^2)$ with an increase in circuit size by factor of $O(t^2)$. The limitation of the technique is its weakness to probing attacks where the information retrieval is dependent on a limited number of physical wires. Ishai et al., in a follow up paper, proposed a method to detect tampering even in a fully compromised circuit followed by self destruction of the circuit [33].

Chevallier-mames et al. proposed a low cost solution in terms of both complexity and computational speed, which creates side channel atomic blocks where the whole code of a process appears as a succession of blocks that are indistinguishable by simple side channel

analysis [45]. This method has the advantages of being inexpensive and generic.

Batina et al. talked about fault attacks that reveal secret information by inserting faults into the device cryptosystem and concluded that clear box testing is necessary in the case of side channel attacks and testing cannot be limited to black box [46]. To gauge the effectiveness of the various strategies proposed to counter side channel attacks, evaluation needs to be performed at the field level. Kocher provided means for testing and evaluating how the various secure strategies to counter side channel attacks perform in practice [47].

Acoustic side channel attacks with sound emanating from a keyboard was studied by Asonov et al. [19]. This was further revised by Zhuang's group in [20]. Similarly, Shamir and Tromer provided a proof of concept of vulnerable acoustic emissions from computing components [40].

Most of the studies discussed here corresponds to systems in general. Among the few that have an architecture of the order of a sensor node is the smart card. The smart card can be considered to have similar limitations as the sensor node though it does not involve any distributed processing as in the case of sensor network, which brings in unique side channel vulnerabilities.

3.5 Summary

The lack of physical shielding and the deployment in open environments make sensor nodes particularly vulnerable to side channel attacks. In this chapter, a comprehensive study of side channel attacks on sensor nodes is presented with four key contributions. First, a three phase attack model applicable to side channel attacks on sensor nodes is introduced. The three phases are training phase, accumulation phase, and attack phase. Second, experi-

mental results on conducting electromagnetic leakage attacks on Tmote-sky sensor nodes is presented. The experiments were performed in an anechoic chamber to replicating environmental conditions of a rural or forest area (interference less). The results show the feasibility of launching side channel attacks on sensor nodes with a strong correlation between the electromagnetic radiation and the data processed by the sensor. Another observation was the non-degradation of signal over smaller distances. This gives the opportunity to the adversary to read the electromagnetic side channel without getting too close to the sensor node. Third, the obfuscation techniques are discussed and a process obfuscation method is proposed, which can be used as a countermeasure for a variety of side channel attacks. Finally, detailed taxonomy of side channel attacks on sensor nodes is presented and for each type of attack, guidelines and approaches to thwart the attack is provided.

Chapter 4

Event Data Propagation Prevention Attack

Sensor networks find application in different disciplines ranging from research to practical on-field military interests. Provided the vast scope, securing a sensor network is critical given the deployment scenario wherein the nodes may be left unattended over long durations of time. The sensor network can be under attack from different types of adversaries, some intentional (malicious user) and some unintentional like environmental conditions.

In this chapter, collaborative adversaries are considered whose goal is to segregate part of the network in order to prevent event reporting by either dropping or corruption of packets. Also, if the events are sporadic, the base station will not be able to differentiate between non-occurrence of an event and non-report of an event due to malicious activity. This is important in military applications such as detecting heavy artillery movement. There have been several studies to protect the sensor network from different kinds of adversaries trying to launch various types of attacks. Protocols like TinySec [14], SPIN [48], TinyPK [15], SERP [16] have been developed to provide security and to maintain integrity of the communication data. Pirzada *et al.* [49] use a trust model like a reputation scheme to identify the sinkholes and wormholes in the network. In this scheme, they guarantee that the packet reaches the base station using a trusted path, but do not detect the malicious nodes in network.

The research studies so far have provided solutions which work in an inside-out fashion, *i.e.* the onus of identifying such mal-intent is left to the sensor nodes. There have been numerous reputation based systems, which use this methodology attempting to solve the problem of intrusion detection and mal-behavior detection of sensor nodes [50–52]. Krontiris *et al.* design an IDS to specifically detect sinkhole attacks [53]. In [54], Ngai *et al.* provide a scheme to analyze packets to detect the intruder. Many of the existing approaches tend to prevent a specific attack type. Even the primary goal of an intrusion detection system is to detect an intrusion while identifying the attack type is secondary. An extension of the intrusion detection system is the intrusion prevention system, which is a real time reactive system to block malicious activities. In [55], Su *et al.* present an intrusion detection and prevention system. What is lacking is a proactive architecture that can detect a malicious attack while also being able to identify the type of attack. The threat model considered is a collaborative attack to segregate a region so as to disallow event packets in this region from reaching the base station. The adversary is a laptop class attacker which can perform four types of attack, namely sinkhole attack, selective forwarding, wormhole and sybil attack to achieve the same.

Dynamic Camouflage Event based Malicious Node Detection Architecture (D-CENDA) uses a multi-phase approach that includes camouflage event generation, planning the mobile-node tour, encoding the address in the packet, packet analysis by the base station, detection and verification of the malicious node. A camouflage event is a reputable event generated in response to a base station request. The camouflage event generator is a mobile-node which can be mounted on robot or an unmanned aerial vehicle. This mobile-node traverses the path decided by the base station and generates the camouflage events at regulated intervals of time. These events are called camouflage events since they mimic the real events, but

are not real events in the true sense. A simple solution is for the base station initiating a camouflage event from the node by sending a message, but a powerful adversary can overhear the communication and allow such event packets. The nodes which are in the sensing range of the camouflage event location will detect the event and report back to the base station. A lightweight address encoding scheme is provided wherein each node encodes the shortened relative address of the node from which it received the packet. Each node also maintains the information about the overheard packet transmissions by the neighbors. We present a bloom-filter [2] and a bit-array based scheme which are used for verification while branding a node malicious.

Performance of D-CENDA is compared with CHEMAS by Xiao *et al.* [12] and sinkhole intrusion detection algorithm (IDA) by Ngai *et al.* [54]. CHEMAS and sinkhole IDA provide protection against individual attack types and is seen that D-CENDA shows marked improvement in malicious node detection while having significantly less false positives. Moreover, D-CENDA is able to identify the type of malicious activity and is flexible to include other attack types. Additionally, results of D-CENDA in relation to the results from CENDA is presented.

The chapter is organized as follows. Classification of nodes and definition of metrics is discussed in Section 4.1. The attack model is presented in Section 4.2. As part of the architecture, design of the mobile route for the camouflage event generator is discussed in Section 4.3. This is followed by attack fingerprinting in Section 4.4. Then, security architecture is described in Section 4.5. Finally, the results are presented in Section 4.6, followed by the summary in Section 4.7.

4.1 Node Classification and Metrics Definition

This section presents the node classification and metrics definition.

Table 4.1: D-CENDA Table of Notations-1

Symbol	Definition
n	Number of nodes
f_i	Forwarding number of node i
f_{mean}	Mean of forwarding numbers
CI	Critical index
U_i	Unit areas having node i coverage
C_a	Set of nodes sensing over unit area a
CoI	Coverage inheritance of node i
b_i	Boolean value of node i is cut vertex
d_i	Feedback on usage of node i
CA	Cumulative Attack

4.1.1 Node Classification

Let n be the number of nodes and f_i is the count of the number of nodes to which node i forwards the packet. This is the forwarding number of node i. f_{mean} is the mean of the forwarding number of all nodes. At network initialization, the forwarding number of a node is set to be inversely proportional to its distance from the base station. An example is presented in Figure 4.1. Based on the forwarding number, the nodes are classified as follows:

1. Regular nodes: Nodes forward packets for two or less nodes including itself.

2. Routing nodes: Routing nodes forward packets for more than two nodes but less than the f_{mean}.

3. Backbone nodes: Nodes forward packets for greater than or equal to f_{mean}.

$$
\begin{aligned}
&Regular\ Nodes = \{n_i\} \forall\ i\ where\ f_i \leq 2 \\
&Routing\ Nodes = \{n_i\} \forall\ i\ where\ 2 < f_i < f_{mean} \\
&Backbone\ Nodes = \{n_i\} \forall\ i\ where\ f_i \geq f_{mean}
\end{aligned}
\tag{4.1}
$$

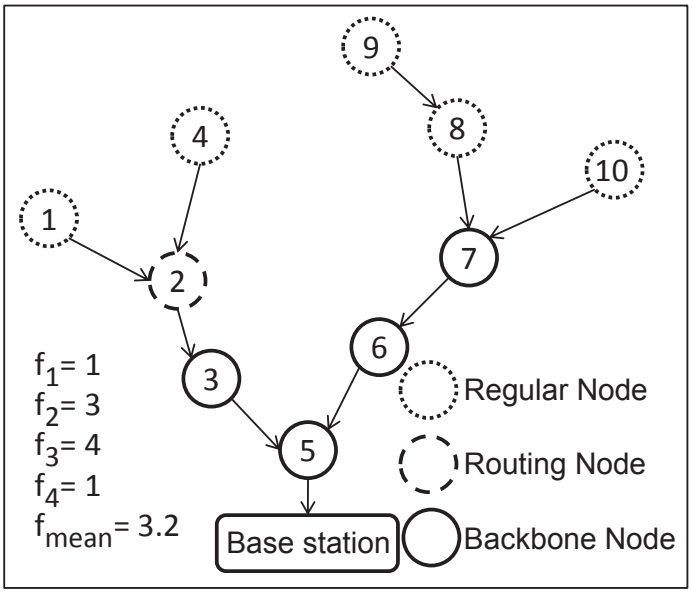


Figure 4.1: Node Classification

4.1.2 Metrics Definition

Critical Index (CI) is a per node metric, which is the availability of sensor coverage over an area. The sensing area is divided into m unit regions. Each node i senses over a set of unit areas called *Sensing Area*. U_i is a set of unit area's over which node i has sensing coverage. Each unit area will be monitored by a set of nodes called *Sensing Nodes* represented as C_a where a is the unit area identifier. The *Coverage Inheritance* (CoI_i) of a node i is defined as the average of C_a where area a belongs to the set U_i.

$$\boxed{CoI_i = \frac{\sum_a C_a}{|U_i|} \text{ where } a \in U_i} \tag{4.2}$$

The coverage inheritance is an inverse property i.e. higher the coverage inheritance of a node, lesser is its importance. It is an acquired factor because the value of this parameter is dictated by other peer nodes sensing the region.

Next, an active component called progressive feedback (d_i) is added. The progressive feedback is the dynamic component which changes based on the real time usage of a node. A node can be close to the base station but can remain redundant until another node fails and the packets need to be routed through it. Also, progressive feedback is different from forwarding number as it is based on the practical usage of the node in the past time periods and not a calculated theoretical parameter. The CI of a node is computed based on the $Forwarding\ Number$ and the CoI of the node. The CI of a node is calculated as follows:

$$\boxed{CI_i = \alpha f_i + \frac{\beta}{CoI_i} + \gamma b_i + \rho d_i \quad \text{Where } \alpha + \beta + \gamma + \rho = 1} \tag{4.3}$$

where α, β , γ and ρ are constant coefficients, b_i is a boolean variable identifying if node i is a cut-vertex. d_i is the usage of node i in the network over the last time period. Based on the node usage as a sensing unit or as a forwarding entity, the critical index gets updated in real time. If a node is a cut-vertex, the importance of the node increases manifold, as the loss of this node can partition the network resulting in some nodes to be unable to reach the base station. The constant coefficients are set based on the application.

4.1.2.1 Impact of Coefficients

An analysis is performed to study the impact of constant coefficients on the number of critical nodes selected. It was done to see the variance of critical nodes selected when compared to the critical nodes selected when each parameter was given the same weight of 0.25. The threshold value of critical index (normalized) is set to 0.25. It should be noted that a lower threshold value will result in higher number of nodes selected as critical nodes. The total number of nodes is 512. It was seen that initially 371 nodes were identified as critical. After 5 runs of simulation, two cases of analysis were performed with the weights as follows:

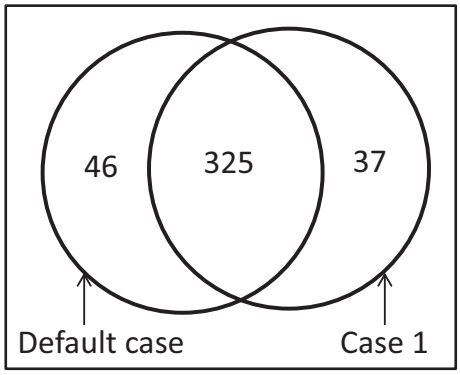


Figure 4.2: Critical Nodes Case 1

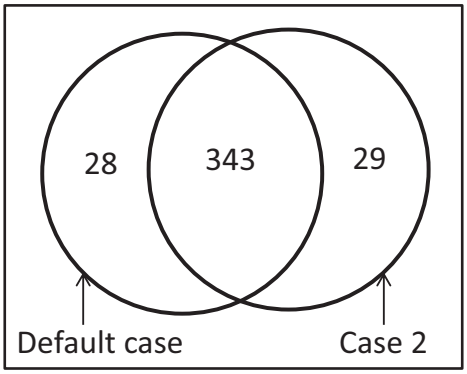


Figure 4.3: Critical Nodes Case 2

1. $\alpha = \beta = \gamma = 0.2$ and $\rho = 0.4$

   The resultant number of critical nodes after 5 periods of simulation is depicted in

Figure 4.2. It is seen that, of the 371 critical nodes only 325 nodes were reselected as critical nodes, but in addition 37 new nodes were selected as critical nodes. 46 nodes which were selected initially as critical nodes were not selected.

2. $\alpha = \beta = \gamma = 0.3$ and $\rho = 0.1$

   Similarly, number of critical nodes and its variance compared to the default is depicted in Figure 4.3.

It is seen that the changes in the weights impacts the critical nodes selected. Giving a higher weight to progressive feedback has its own pros and cons. As discussed earlier, a node may not be currently used, but can be significant in routing packets if another node which is currently heavily used goes down. Hence, depending on the application, it is important to give the proper weightage to each of the parameters.

4.2 Attack Model

The adversary is a laptop class attacker and on capturing a node, it has access to everything on the node. Also, the malicious entity would want to capture the nodes such that the minimum number of node captures result in the maximum damage. Being a powerful node, it is reasonable to assume that the malicious entity can overhear communication over a larger area compared to the sensor node and can make informed decisions about the nodes it wants to attack. The malicious entity captures a node and does one or more of the following attacks: Sinkhole attack, Wormhole attack, Sybil attack or Selective Forwarding attack. The goal of the attacker is to segregate a region such that event packets do not get reported from the region, but may allow other communication. The modus operandi of the threat model

is to increase the value of *Cumulative Attack* (CA), where cumulative attack is defined as follows:

$$\boxed{CA = \frac{\sum C_p}{|p|} \; Where \; C_p - \; critical \; index \; of \; node \; p} \tag{4.4}$$

The adversary wants to compromise a node and perform attack from the list above in order to achieve the goal of region segregation. The attacks can be composite in nature, *i.e.* occurrence of multiple different types of attacks simultaneously.

4.3 Mobile-Node Route Design

The mobile node is the camouflage event generator. The route followed by this node governs the location and time of the camouflage event which in turn determines the nodes that will detect this event. The design of an optimized route is critical due to the following reasons: A poorly designed mobile node route can cost the network dearly in the form of event detection by unwarranted nodes and having the network flooded by these packets. The second reason is the waste of time and energy of the mobile node to follow a non-prime path. The process includes a node selection preliminary step.

4.3.1 Node Selection

It is costly to protect all nodes. Under attack, it is required to be able to identify and protect the nodes which are critical to the network functioning. Hence, the nodes are prioritized based on their role and importance. This importance could be dictated by different factors like location, power, etc. Based on the classifications and definitions in Section 4.1, select a subset of nodes such that they satisfy the requirements and is depicted in Algorithm 1.

Table 4.2: D-CENDA Table of Notations-2

Symbol	Definition
CN	Set of critical nodes
MCN	Maximum coverage node
C_m	Set of unit areas
L_c	Camouflage event location set
$CovAreas$	Set of unit areas covered
T	Sensing range of node
τ	Critical index threshold
X_n	Set of all nodes
X_p	Set of privileged critical nodes
T	Sensing range of node

Algorithm 1: Node Selection

Input: $\{CovArea\} \Leftarrow C_m$, $\{CN\} \Leftarrow \phi$, $X_n = \{Nodes\}$
Output: Critical nodes in $\{CN\}$
for $x_i \in X_n$ *and* $CI_i > \tau$ **do**
  $\{CN\} \Leftarrow \{CN\} + x_i$
  $\{CovArea\} \Leftarrow \{CovArea\} - U_i$
while $\{CovArea\} \neq \{\phi\}$ **do**
  $MCN \Leftarrow x_i$ where $(((x_i \in X_n)$ and $(x_i \notin CN))$
    and $x_i = (max(U_n - \{CovArea\})))$
  $\{CovArea\} = \{CovArea\} - U_i$
  $\{CN\} = \{CN\} + MCN$

4.3.2 Route Design

M is the mobile route, $L_1, L_2..., L_c$ are the locations on route M where the event generation occur and c is the number of stops in the mobile route. If the routing paths of the packets is known, find the subset of nodes in X_n which should detect the event such that all nodes in CN either detect the event or are on the routing path of node detecting the event. Let this subset of nodes which detect the event be the set of privileged node represented by X_p. If the routing paths of the packets is unknown, the set CN is the set of privileged node represented as X_p. The goal is to find set of locations L_a in the field such that all nodes in X_p are at a distance of sensing range or less from at least one location in L_n. This is

presented in Algorithm 2. In the event of having multiple mobile vehicles, the set X_p is geographically partitioned into the number of mobile vehicles available and the mobile route can be planned for each of the partitions independently. The next step is to calculate the shortest path to cover the locations in set L_c. This is formulated similar to the traditional 'Traveling Salesman Problem' but with an optimization exception to allow the mobile node to revisit any prior visited location.

Algorithm 2: Camouflage Event Location Selection

Output: Set of event locations $\{L_c\}$
$\{L_c\} \Leftarrow \{\phi\}$
while $\{X_p\} \neq \{\phi\}$ **do**
  for $C_m \notin \{L_c\}$ **do**
    $SenseCov_m \Leftarrow Count(Distance(X_p, C_m) \leq T)$
  $MaxCovArea = Max(SenseCov_m)$
  $\{L_c\} = \{L_c\} + MaxCovArea$
  $\{X_p\} = \{X_p\} - \{Node\ X_i\ Sensing\ MaxCovArea\}$

4.4 Attack Fingerprinting

In this section the attacks are modeled based on the characteristics possessed by each attack type. A high level classification based on some of the parameters used in D-CENDA is shown in Figure 4.4. A detailed information about the attacks is presented in [56] and [17].

4.4.1 Sybil Attack

The basis of sybil attack depends on two factors, first the ease of acquiring different identities, and second, the amount of damage a node can inflict by acquiring the identity. In our system which is based on the node's response to the real-world-like camouflage events, a node will not be able to tarnish the reputation of another node since the neighborhood of the nodes

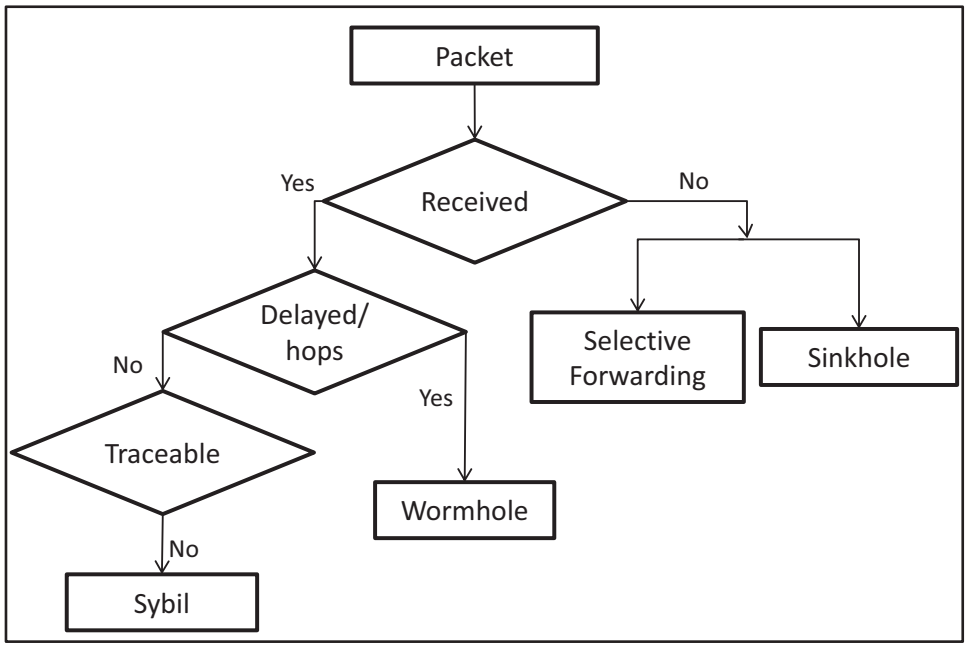


Figure 4.4: Attack Taxonomy

is set and the peers are not required to maintain reputation. The control of managing the reputation lies with the base station.

In D-CENDA, the sybil attack possesses characteristics similar to wormhole attack in which case, the non-verification of the trace-back path to the originating node indicates the presence of malicious activity.

4.4.2 Wormhole Attack

Lemma: It is not possible for the colluding nodes to do significant damage and thwart being identified in case of wormhole attacks in D-CENDA.

Successfully tracing back to the source under wormhole attack: Consider the scenario in Figure 4.5 wherein the source node s sends packet to the base station. The nodes 3 and 4 collaborate to launch a wormhole attack. In this case, node 4 will fake a relative address of a neighbor before forwarding the packet since it cannot specify that it received the packet

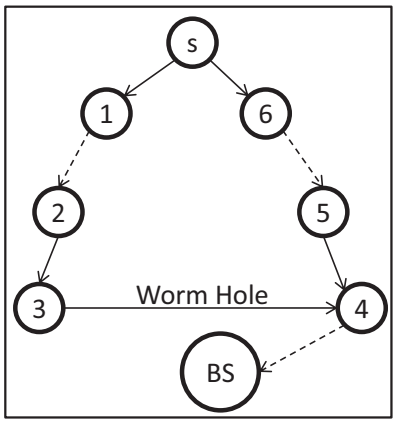


Figure 4.5: Wormhole

from node 3. For the base station to be able to trace-back to the source, it needs to satisfy the following. There needs to exist a valid path from node 4 to source with $h + 1$ number of hops where h is the number of hops from node 3 to the source. Let there exist a path P satisfying this condition. It needs to satisfy the per-hop condition such that relative addresses encoded in the packet from source to node 3 match that of a new path from node 4. Also, the effectiveness of a wormhole attack depends on how farther away the traffic is re-routed in the network, thereby increasing the resource consumption in the network. If it needs to satisfy the first condition *i.e.* to maintain the hop count, the collaborating attacking nodes within the network cannot reroute the packet across large distances.

4.4.3 Selective Forwarding Attack

Selective forwarding attack is a form which can be easily confused with sinkhole attack. Also, a bad channel leading to packet losses can imitate a selective forwarding attack. To differentiate this attack from packet losses due to bad channel or interference, some knowledge of the channel condition is required which is not in the scope of this thesis. But to differentiate it from a sinkhole attack, it is required to gather packet loss information over a larger period of

time. For this purpose, the dynamics of analysis for sinkhole attack and selective forwarding attacks are different. For selective forwarding attacks, the information collection period has to be longer compared to sinkhole attack. Hence, in D-CENDA while a sinkhole attack can be identified within a few rounds, identifying the selective forwarding attack requires to gather information over multiple rounds. In our simulation, to detect a selective forwarding attack the analysis of the data collected over 5 rounds is performed.

4.4.4 Sinkhole Attack

In sinkhole attacks, the node advertises a low cost path to the base station and may or may not evince itself to be within the neighborhood of a larger number of nodes than it actually is. The base station knows the location and neighborhood information of all the nodes and can trace the loss of packets to within a few hops of this malicious node. This attack type can be identified by analyzing the packets lost and then verified by querying the bloom filters/bit-arrays of neighborhood nodes in the identified region.

4.5 Secure Architecture

The secure architecture is a sequence of processes for proactively identifying the malicious nodes in the network. It begins with the initiator in the form of camouflage event generator and ends with a verifier using which the base station verifies the mal-intent of the node before qualifying it as a malicious node. It is a standalone module, which consists of the camouflage events, embedding path information, packet meta-analysis, and verification.

4.5.1 Camouflage Events

The events generated by the mobile nodes are called camouflage events. To the base station, the camouflage event possesses some distinct properties which are not characteristic of real world events. These properties are as follows:

1. The base station is aware of the location of the event occurrence (L).

2. The base station knows the precise time of the occurrence of the event (T).

3. With the knowledge of the location and time of event occurrence, the base station knows the set of sensor nodes that detect the event (SN).

The mobile node starts traversing the route designed by the base station. While traversing the path, it stops at the designated location and generates the camouflage event. The nodes sensing the event respond back to the base station. The sensor nodes cannot differentiate this event from an actual event and hence it is handled like a real world event. It needs to be emphasized because event type anonymity is crucial to this methodology. If a malicious node can differentiate this event from a real world event, it can treat just these events like a well-behaved node to escape detection. In D-CENDA, the onus of detecting malicious behavior is on the base station. The sensor nodes only need to record overheard packet transmissions by the neighbors.

The base station collects all packets that were generated and have traversed through the network. For each packet received by the base station in response to a camouflage event, there are two types of nodes involved. The event notifier node which reports the occurrence of the event and the intermediate nodes through which the packet traversed to reach the base station. At a higher level every packet received by the base station provides one bit information about each of the nodes involved in the delivery of the packet.

Depending on routing, there are two distinct cases that are considered. One in which the packet follows the same path for a set period of time. In this case the base station is aware of the path the packet traverses. Hence, when the packet reaches the destination, the base station knows the nodes which propagated the packet to it. In the second case, instead of following a particular route, the packet follows geographical routing. For the base station to know the path traversed by the packet, the intermediate nodes need to append additional information. In the absence of this additional information the base station can only make an educated guess about the intermediate nodes using the knowledge of the location of the nodes and their sleep cycles.

4.5.2 Embedding Route Information

In the absence of fixed route, a scheme is devised in which the nodes append the route information to the packet before transmitting over the next hop. This address information being attached to the packet can be of three types.

1. Absolute address: In this case the node appends the absolute address of the neighbor from which it received the packet. The drawback of this scheme is the amount of space it takes to represent the absolute node address.

2. Relative address: Taking advantage of the knowledge of the node distribution by the base station, a scheme is presented in which the nodes appends the relative address of the neighbor node from which it received the packet.

3. Shortened relative address: The direction of the flow of the event packets is towards the base station. Using this, the neighbor nodes are identified that should forward the

packet to the node and take into consideration only these nodes while encoding the neighbor address in the forwarded packet.

Absolute address is suitable if the number of nodes is small, resulting in shorter address. But the number of nodes in a sensor network can be large, making the absolute address longer. The length of the relative address is dictated by the number of neighboring nodes. To analyze the overhead disparity between the absolute address and relative addresses, simulations were run to study the distribution of the nodes based on the following parameters: number of nodes, area of field, type of distribution. The type of distribution can be uniform or non-uniform. In uniform distribution, the nodes are homogeneously spread in the sensing area, which leads to the nodes having a very even neighborhood resulting in relative address length close to the mean of all relative address lengths. If the distribution is non-uniform, for densely distributed areas, the nodes are programmed such that only a set of nodes are active at a point of time. This set of nodes can be mutually exclusive to non active nodes.

Using the availability of the topology at the base station, the relative address is further enhanced by using a shortened relative address. This is achieved by classifying the neighboring nodes as upstream and downstream neighbors. An upstream neighbor node is a valid node from which this node receives a packet to forward to the base station. The other nodes which should not send the packet to this node to forward to the base station are classified as downstream neighbor nodes. Since a node should be only receiving packets from its upstream neighbor nodes, the relative address can be shortened to only encode these neighbors and is called shortened relative address.

Table 4.3 displays the comparison between the absolute address length and the relative address length for different area sizes and different node densities. The absolute address

Table 4.3: Node Distribution

Node	Area(m^2)	Average Hops	AAL^*	RAL^{ψ}	$SRAL^{\mu}$
512	1000 X 1000	8	72	32	24
1000	1000 X 1000	8	80	34	26
2000	1000 X 1000	7	77	38	31
3000	1000 X 1000	7	84	38	31

* Absolute Address Length (bits)
ψ - Relative Address Length (bits)
μ - Shortened Relative Address Length (bits)

length is calculated as the average number of hops multiplied by the number of bits needed to represent the absolute address. Let us consider the case in which the area is 1000 x 1000, with 512 nodes (this gives us a coverage of four times unit density since 128 nodes are needed for unit density coverage while assuming sensing range of $50m$). It is seen that the average path length is eight hops and the average neighborhood is 14.12. Also, on an average four bits are needed to represent the relative address of the neighbor (average neighborhood $\leq$ 16). Along with the average path length of eight hops, 32 bits are required to represent the path from the source to the destination.

4.5.2.1 Shortened Relative Neighbor Address

The address and representation of the neighbor A for node B is based on two factors. The absolute node identifier of A and the number of neighbors of node B. The relative addresses are assigned in the increasing order of neighbors node identifier. For node 23 in Figure 4.6, the relative address and shortened relative address of the neighbor nodes are listed in Table 4.4. The relative address can be represented using four bits while the absolute address can require up to seven bits. The neighbors which forward packet to a node are the upstream neighbor of the node and all other neighbor nodes are downstream nodes. The downstream neighbors all have shortened relative address of 0, while the upstream neighbors

have an incremental address starting from 1 based on their absolute address. Using this classification the address encoding length can be further reduced. It is seen to have address length shortening (up to 25%) when using the shortened relative address.

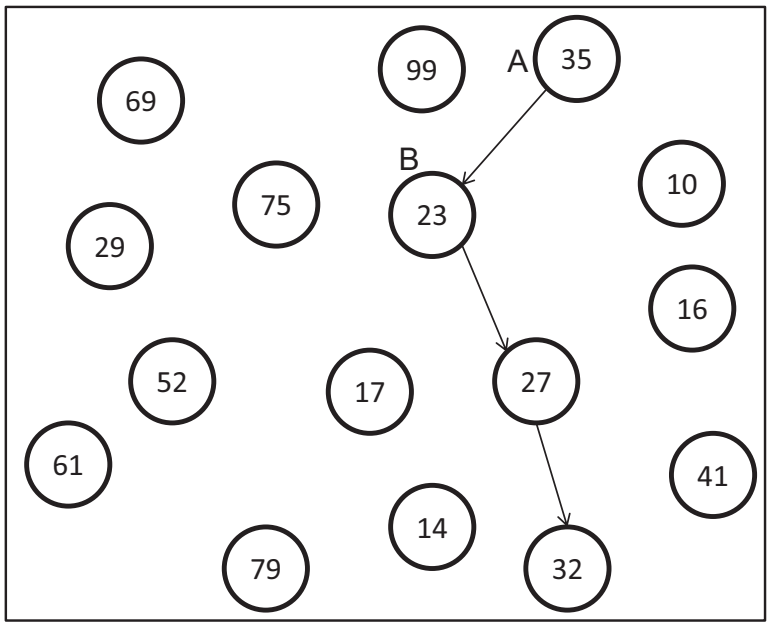


Figure 4.6: Packet Route

Table 4.4: Node 23 Neighbor Address

Aid	Rid	SRid
10	0	1
16	1	0
17	2	0
27	3	0
35	4	2
75	5	0
99	6	3

Aid - Absolute Node identifier
Rid - Relative Address
SRid - Shortened Relative Address

4.5.2.2 Encoding the Path Address

When a node receives a packet it appends the relative address of the node from which it received the packet and encrypts it before forwarding it over the next hop. If node o received the packet from node m, the encryption is shown below, where $[data]_{PK_o}$ is the encryption of $data$ with the key of node o.

$$\boxed{\begin{array}{c} [Address||RA_{m|o}]_{PK_o} \quad Where\ A||B\ -\ Append\ B\ to\ A \\ RA_{m|o}\ -\ Relative\ address\ of\ m\ w.r.t\ o \end{array}} \tag{4.5}$$

Table 4.5: Address Encoding

Node 23		Node 27		Node 32	
Aid	*SRid*	*Aid*	*SRid*	*Aid*	*Rid*
10	1	14	0	14	1
16	0	16	1	27	2
17	0	17	2	41	3
27	0	23	3		
35	2	32	0		
75	0	41	0		
99	3				

Aid - Absolute Node id
SRid - Shortened Relative Node id

$$
\begin{array}{c}
[Addr]_{PK35} \\
\downarrow \\
[[Addr]_{PK35}||10]_{PK23} \\
\downarrow \\
[[[Addr]_{PK35}||10]_{PK23}||11]_{PK27} \\
\downarrow \\
[[[[Addr]_{PK35}||10]_{PK23}||11]_{PK27}||10]_{PK32}
\end{array}
\tag{4.6}
$$

In Figure 4.6, the event is detected by node 35 and it is transmitted to the base station via nodes 23, 27, 32. Table 4.5 lists the shortened relative address for the three nodes receiving the packet. Each node before forwarding the packet over the next hop, encodes the relative address information in the packet as follows. Node 23 will encode the relative address of node 35 which is 2 before transmitting to node 27. Node 27 will encode the relative address of node 23 which is 3 before forwarding the packet to node 32 and so forth. This is depicted in Equation 4.6.

4.5.2.3 Decoding the Path Address

When the base station receives the packet, it knows the sender of the last hop of the packet. In Figure 4.6, the last hop forwarder is node 32. The base station also knows the neighbors of node 32. On decoding the packet using the key specific to node 32, the base station knows the relative address of the node which forwarded the packet to node 32. Looking up the table, the base station identifies that the packet was received from node 27. Recursively, the

base station can trace the complete path back to the sender. This is represented as follows:

$$\boxed{\begin{array}{c} [EncryptedAddress]_{PK32} \Longrightarrow [EncryptedAddress_1 \mid\mid 10] \\ \downarrow \\ [EncryptedAddress_1]_{PK27} \Longrightarrow [EncryptedAddress_2 \mid\mid 11] \\ \downarrow \\ [EncryptedAddress_2]_{PK23} \Longrightarrow [EncryptedAddress_3 \mid\mid 10] \\ \downarrow \\ Node35 \end{array}} \tag{4.7}$$

4.5.3 Packet Meta-analysis

Let X_i be the set of packets that are generated as a result of a camouflage event. Let R_i be the set of packets that were received by the base station and L_i is the set of packets that were not received.

$$\boxed{R_i \subseteq X_i, \quad L_i \subseteq X_i, \quad R_i \ + \ L_i = X_i} \tag{4.8}$$

The base station maintains records of characteristics for the different attack types for each node. This record consists of a set of parameters which are updated by the base station whenever a packet is generated for a camouflage event. As a packet is received or not-received, the record for the nodes gets updated accordingly.

A set of four parameters is maintained for each node; packets generated, packets received, packets lost, packets garbled. Packets generated is a parameter which keeps track of the packets generated by the node in response to the camouflage event. Packet received

is a parameter of the node which represents the number of packets successfully received when the node was either the source of the packet or an intermediate node forwarding the packet. Packet lost parameter of a node tracks the packets that were lost and is statistically determined to see what path the packet should have taken to reach the base station. Using this information, the intermediate nodes are also marked for the packet lost. The packet garbled parameter tracks the number of packets which were successfully received, but cannot be traced back to the source.

4.5.3.1 Packets Received

These are the packets which are received by the base station in response to a camouflage event. The hop count and the total time elapsed between the camouflage event and the time of reception of the packet are calculated. This is possible because of the availability of the temporal information about the occurrence of the event. The base station checks if the number of hops and the time of delivery is within the deviance limits for the particular node. If either of the two parameters lie beyond the threshold limits, a path analysis is performed.

If the number of hops is within the limits (called Hop-Validity) and if the time elapsed is also within deviance (Time-Validity), the nodes packet received parameter is marked. If either of hop-validity or time-validity fails, a path analysis is performed for the packet. The first step in path analysis is to do a trace-back to the source. Under certain circumstances like a wormhole attack, the trace-back to the source may not be successful. If the trace-back is successful, a per-hop analysis is performed for the packet received. In a per-hop analysis, each pair of consecutive intermediate nodes are considered and verified to see if they adhere to the upstream-downstream requirement.

Let P_i be the packet sent by node N_s to report an event E. m is the number of hops

and t is the time taken by the packet to reach the base station. h_e and t_e are normalized expected number of hops and time.

$$\boxed{Hop\ Validity_i = \frac{m - h_e}{h_e} \qquad Time\ Validity_i = \frac{t - t_e}{t_e}} \tag{4.9}$$

If either of the two validity fails, the packet is traced back to the source based on the encoded relative addresses. If relative address is used and trace back to the source is successful, for each hop of the packet, let node x_t be the transmitter and node x_r be the receiver of the packet. Check if node x_r is a downstream neighbor of node x_t. If it is not x_r, mark node x_t. If the trace-back fails, mark the packet garbled parameter similar to the packet lost parameter. This is presented in Algorithm 3. If shortened relative address is used, and if any of the neighborhood address is 0, query the node which received the packet from the downstream neighbor and mark that node.

The table gets populated over a period of time and we perform a short term analysis and a long term analysis. Each of these analysis caters for a particular type of attack. The short term analysis is done to identify wormhole, sybil and sinkhole attacks while a long term analysis is performed to identify a selective forwarding attack. The long term analysis has sufficient information that helps us to differentiate between the sinkhole and selective forwarding attacks. Also, it helps us differentiate between a dead node and a malicious node performing a selective forwarding attack. The reasoning behind this is the fact that to identify a selective forwarding attack, it is needed to gather information over longer durations as compared to other attack types.

Algorithm 3: Node Marking

Input: $N \rightarrow$ *all nodes*, $R_i \rightarrow$ *received packets*
$L_i \rightarrow$ *lost packets*, $x_n \rightarrow$ *nodes on path of packet* P_n
for $P_i \in R_i$ **do**
  if $((HopValidity_i) \,||\, (TimeValidity_i))$ *fails* **then**
    Initiate Source Trace-back
    while $(n_i \neq N_s) \,||\, (TracebackAdd! = \phi)$ **do**
      $p = NumNeighbors(n_i)$
      $RelAddLen = sizeofbitlength(binary(p))$
      $RelAdd = RelAddLength\ LSB\ of\ TracebackAdd$
      $n_i = Lookup \rightarrow RelAdd(n_i)$
      $TracebackAdd = TracebackAdd - RelAdd$
    if *Source Trace-back == SUCCESS* **then**
      for $(Node\ n \in x_n)$ **do**
        if $n \in$ *path of* P_i **then**
          if x_r *is downstream neighbor of* x_t **then**
            $x_t \rightarrow SuspectNode$
    else
      Intermediate Node Identification $(Node\ n \in x_n) \rightarrow Path\ of\ P_i$
      $Mark\ PacketGarbled(n)$
for $P_i \in L_i$ **do**
  Intermediate Node Identification
  for $(Node\ n \in x_n) \rightarrow Path\ of\ P_i$ **do**
    $Mark\ PacketLost(n)$
  Marked Node Analysis
  for $(Node\ n \in N)$ **do**
    if $PktLost(n) \geq 1$ **then**
      $u \in UpstreamNeighbor(n)$
      if $PktLost(n) > \sum PktLost(u)$ **then** $n \Rightarrow suspectnode$
    if $PktGarbled(n) \geq 1$ **then**
      $u \in UpstreamNeighbor(n)$
      if $PktGarbled(n) > \sum PktGarbled(u)$ **then**
        $n \Rightarrow suspectnode$

4.5.3.2 Packets Lost

With the knowledge of the occurrence of the events, the base station expects a set of packets from a group of nodes. If there is malicious activity, there may be loss of packets. For each packet lost the base station is able to trace-back the packet loss to within a subset of nodes. This is depicted in Algorithm 3.

4.5.3.3 Verification

Verification is the process performed by the base station before finalizing the node as malicious. To pinpoint the node which was responsible for the packet loss from within the subset of nodes, each node maintains a simple but efficient data structure which works on the principle of bloom filter and stores overheard packet transmissions. Two methods are presented to maintain the overheard packet transmission.

1. Bloom filter: Each node maintains multiple counting bloom filters. These are used to mark the overheard packet information during different intervals of time. The nodes overhear the packet transmissions of their neighbors and mark information into the bloom filter using the relative address of the neighbor. The number of bloom filters maintained by each node will decide the granularity of the information stored. For example, information maintained in a single bloom filter over a period of time t will give us less accurate description of events compared to 10 bloom filters maintained by the node for each of $t/10$ time intervals.

2. Bit-array: It is seen in Section 4.5.2, each node will have a maximum of 32 active neighbors when the node itself is active. If a node has a higher neighborhood density, the

sleep cycles are programmed such that a node has a maximum of 32 active neighbors. Since the total number of neighbors are few, the second method is to use bit-arrays to store overheard packet transmission. Use of bit-array simplifies the amount of computation required at a sensor node to gather and store the data. Additionally, when the node is queried by the base station, it is a simple lookup operation by the sensor node to respond to the query. Also, converting a bit-array into a counting bit-array will require replacing the individual bits by multiple bit cells. Although it is not the most optimal method, but it is the most simplistic method, particularly considering a energy constrained sensor node.

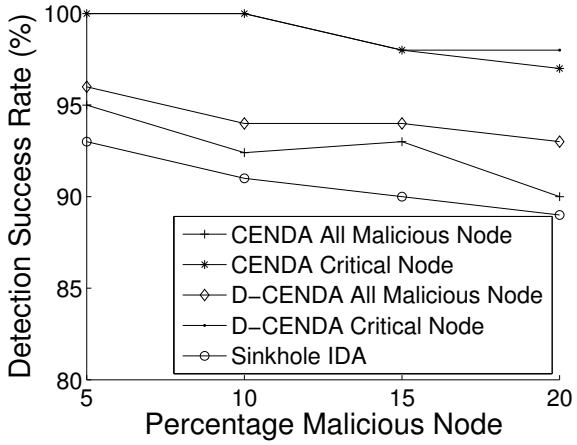
The base station can query the neighboring downstream nodes to check if they overheard the packet transmission by the node over a particular interval of time. As for the response to the query, the architecture provides two options. One, the queried node responds back to the base station with its bloom filter or bit-array representation and the base station processes it to decipher the information. In the second case, the node processes the data structure to extract the data and replies back with only the required information. The two methods have their own advantages and disadvantages. In the first method where the sensor node returns the entire structure as a response, the base station might be saved from making repeated queries to the same node about its different neighbors. Another significant advantage of this method is the discreteness with regards to the node under investigation. The drawback is the increased packet size. In the second method, the communication cost is less, but the base station will need to query the node about a specific neighbor and cannot be discrete like in the first method. Hence there is loss of anonymity and it can be qualifying factor in certain military applications.

For un-received packets over a period of time, the base station can trace the packets loss to a subset of nodes from the information it gathers for every un-received packet. The base station queries multiple neighbors (this number can be configured) of these nodes to check if they overheard a packet being transmitted by their neighbor at a particular time interval. The responses are analyzed to check the presence of suspect nodes. The nodes are aggressively marked as suspect, hence this verification process plays a important role in keeping the false positives low while detecting malicious node. It will be seen in the results in Section 4.6, that the percentage of the false positives considering critical nodes is low (6.5% in sinkhole attacks, 8% in selective forwarding attacks, 9.75% in Sybil attacks, and 10.5% in wormhole attacks). The low false positive rates indicate that the exoneration of the non-malicious suspect nodes is high, displaying the higher probability of successful verification by the base station.

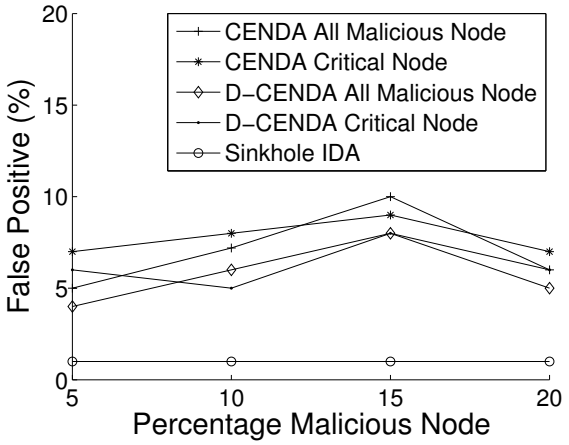
4.6 Simulation Results

The simulations were performed over an area of 500x500m, having 125 nodes so as to have a unit coverage over the entire field with each node having sensing range of $25m$. The critical index threshold is 0.25 and the four parameters $(\alpha, \beta, \gamma, \rho)$ were each set to 0.25. The simulation is performed in Matlab. There were two varying factors in the attacks. First, the number of malicious (compromised) nodes was varied as 5%, 10%, 15%, 20%. The compromised nodes were randomly selected but the critical nodes were given a higher priority to be under attack considering a worst-case scenario with a sophisticated attacker. Second, the attack type of the compromised nodes were again randomly assigned. The base station gathered data over multiple runs. The results of D-CENDA is presented while keeping the

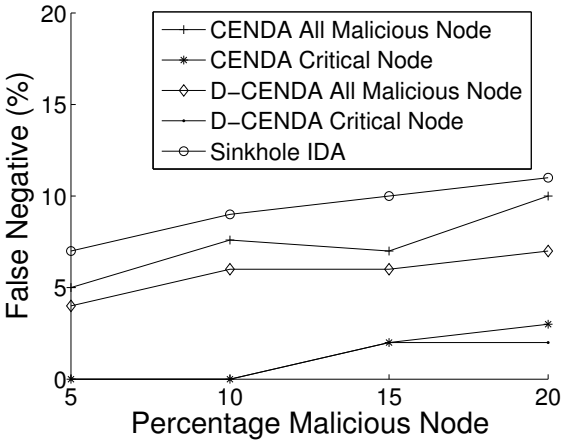
results from CENDA for a quick comparison and also include the results in sinkhole intrusion detection algorithm by Ngai *et al.* [54] and CHEMAS by Xiao *et al.* [12] where relevant.



(a) Sinkhole Detection



(b) Sinkhole False Positive



(c) Sinkhole False Negative

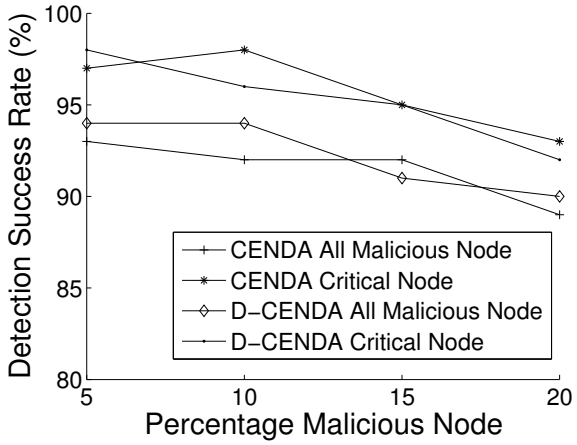
Figure 4.7: Sinkhole Attack

Figure 4.7a depicts the detection success, when all the malicious nodes are performing sinkhole attack. Even though the success rate of identifying the malicious node was not 100%, the subset of malicious node which are critical were identified accurately. There are two reasons for this, the critical nodes have connectivity to the base station and the camouflage events are generated to cover the critical nodes. It is also seen that the success

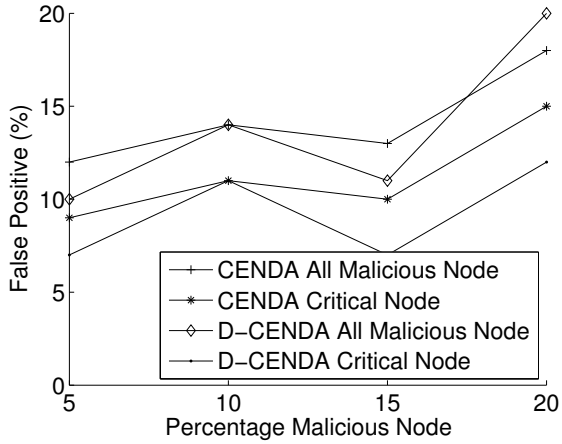
rate of detecting all malicious nodes is higher for D-CENDA compared to CENDA. This is a result of having dynamic selection of critical nodes which results in selection of some new non-critical malicious nodes for detection purpose that will not occur in CENDA. The results from [54] (sinkhole IDA) have been included for comparison, and it is seen that D-CENDA outperforms sinkhole IDA with respect to malicious node detection success rate.

The false positive rate when only considering the critical nodes is slightly higher as compared to the false positive rate when the entire set of malicious nodes was considered. Again, as the number of malicious nodes increased there is an increase in the false positive rate. This is because a node is aggressively marked as suspect to get lesser false negatives, with assurance that the verification discards the false positives as depicted in Figure 4.7b. Even in false positives accuracy an improvement is seen in D-CENDA. The false negative rate when considering the critical nodes was much lower compared to the false negative rates of the whole malicious nodes set (Figure 4.7c).

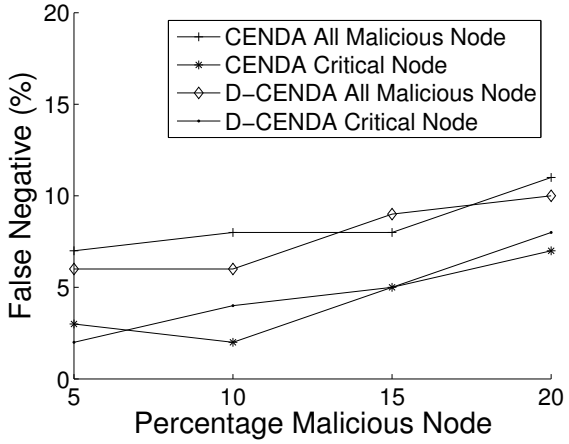
The next set of results comprise the malicious nodes all performing selective forwarding attack. The results are presented when the nodes randomly dropped 30% of the packets. It was seen that this drop percentage affected the number of rounds the camouflage event generator needed to make, to get accurate results. Lower the drop percentage, higher was the number of rounds needed by the mobile-node, but again lower drop percentage means a less severe attack. The overall performance of D-CENDA was better than CENDA for all the cases except when the percentage of malicious nodes was 15%. This was happening because the dynamic selection of critical nodes changes the critical nodes which are under observation. As mentioned earlier, identifying a selective forwarding attack required us to gather data over multiple runs and the change in nodes under observation within those runs resulted in some undetected malicious nodes. This was seen to happen for critical nodes in



(a) Selective Forwarding Detection



(b) Selective Forwarding False Positive

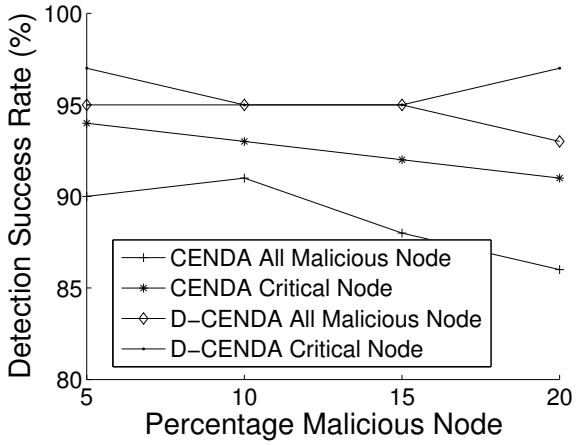


(c) Selective Forwarding False Negative

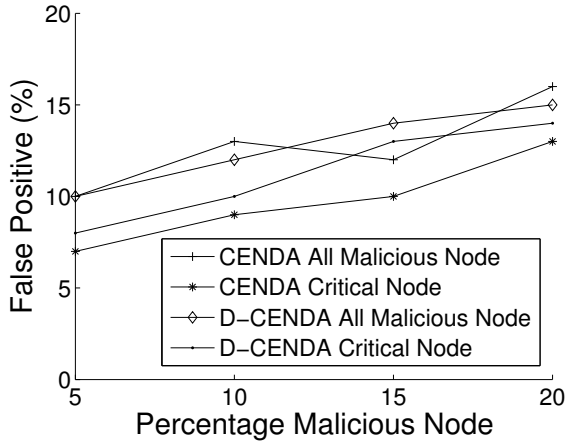
Figure 4.8: Selective Forwarding Attack

the 5% and 20% cases. The results are presented in Figures 4.8a, 4.8b, 4.8c and are similar to sinkhole. Unlike in sinkhole, the analysis to detect selective forwarding attack is performed over 5 rounds by the mobile-node. In other words, it was slower to catch a malicious node performing selective forwarding attack.

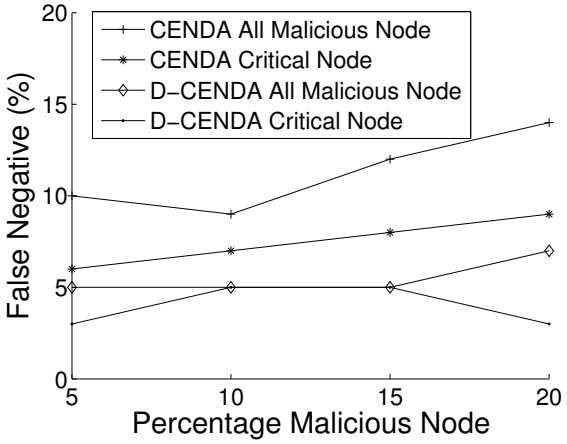
Figures 4.9a, 4.9b, 4.9c represents the detection success, false positives and false negatives for the network under sybil attack. The assumption was that at a point in time, a node can acquire the identity of another node, but does not have access to cryptographic information



(a) Sybil Detection



(b) Sybil False Positive

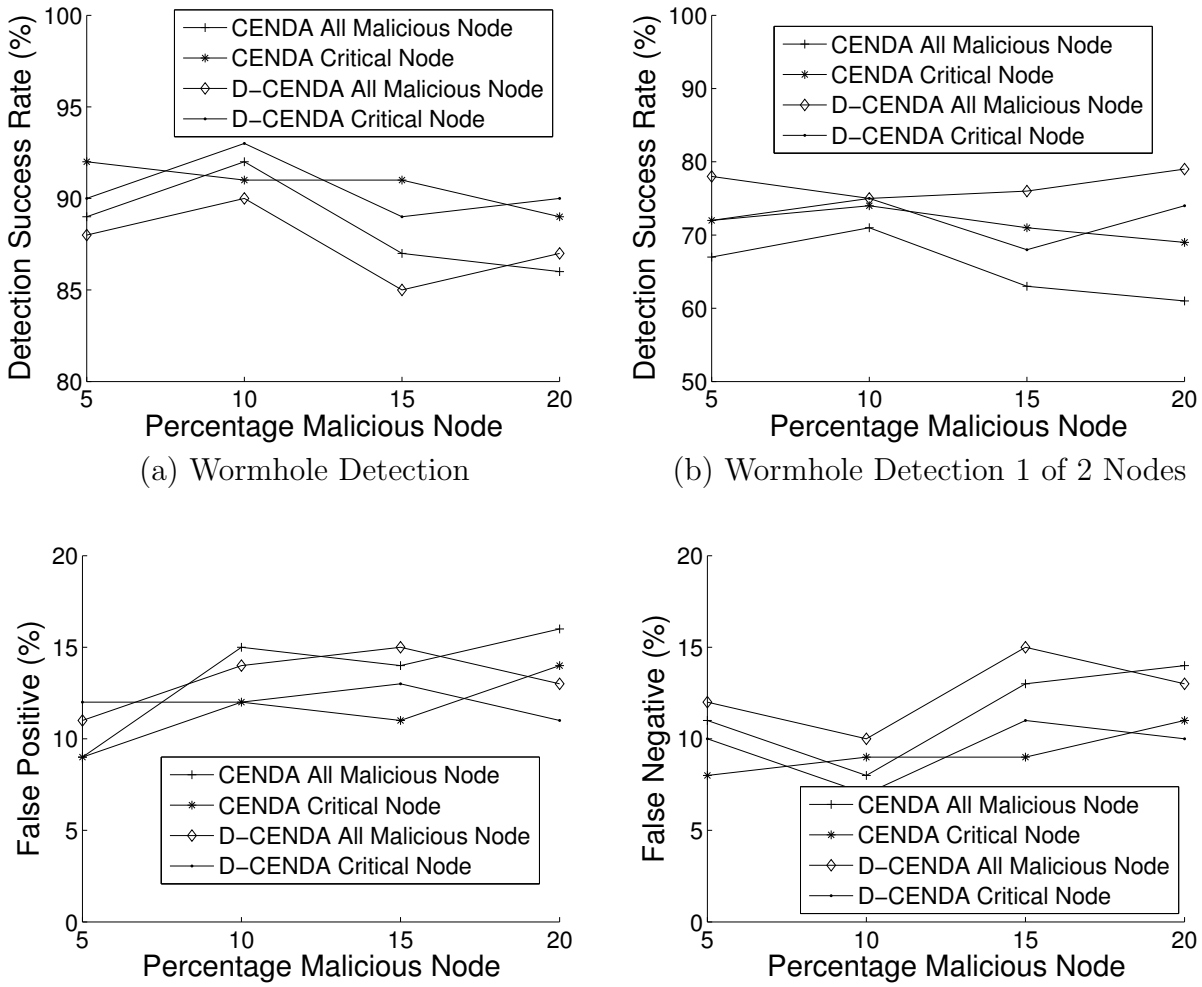


(c) Sybil False Negative

Figure 4.9: Sybil Attack

of the node. When comparing the performance of D-CENDA with CENDA, D-CENDA performed better in most cases and no worse than CENDA in the rest.

In the next simulation, all randomly selected malicious nodes launch a wormhole attack. A wormhole attack is different from other attacks due to the fact that a single attack will have at least two colluding nodes. In wormhole attack, randomly selected nodes collude with each other and launch the attack by directing the traffic to the other side of the network. The results for the wormhole attack is presented in Figures 4.10a, 4.10c, 4.10d. When



(a) Wormhole Detection

(b) Wormhole Detection 1 of 2 Nodes

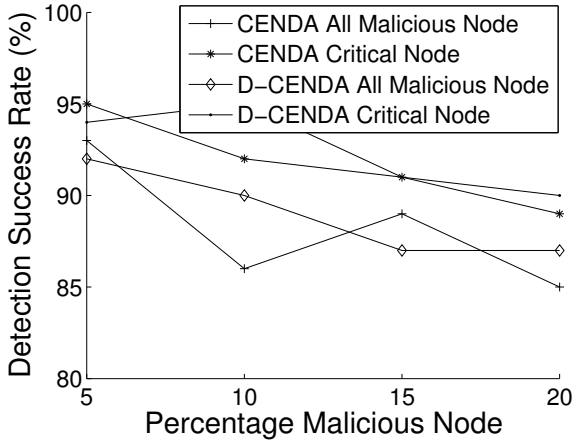
(c) Wormhole False Positive

(d) Wormhole False Negative

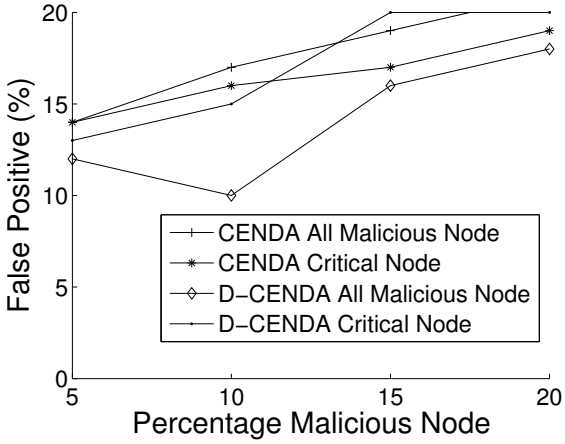
Figure 4.10: Wormhole Attack

considering all malicious nodes, CENDA outperformed D-CENDA, and while considering only the critical nodes the performance was comparable. Under the circumstance when just one of the node is directing traffic to another part of the network, the model had difficulty in identifying the other colluding node since it was only directing traffic to the base station and was not doing anything malicious per se. This is depicted in Figure 4.10b. When only one of the nodes is launching a wormhole attack, D-CENDA outperforms CENDA. This can be accounted to the dynamic selection of critical nodes which results in looking out for the

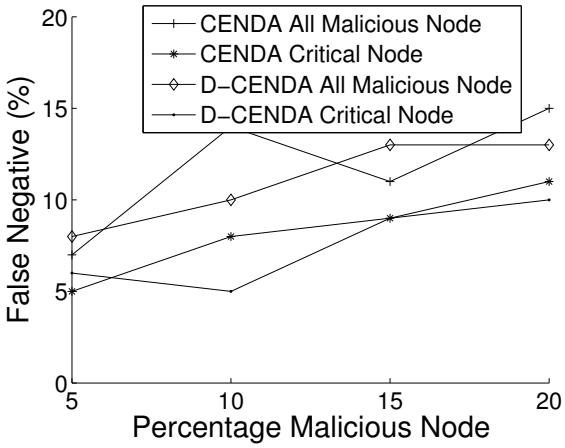
more active malicious nodes.



(a) All Attacks Detection



(b) All Attacks False Positive



(c) All Attacks False Negative

Figure 4.11: All Attacks

The next set of simulations involved malicious nodes having equal distribution of all the four attack types. The results for the same are shown in Figures 4.11a, 4.11b, 4.11c. In short term analysis, *i.e.* when analyzing each round of camouflage event packets, it was difficult to differentiate the sinkhole attack from selective forwarding. Analyzing the camouflage packets over multiple rounds produced sufficient information to demarcate the same. The results provided in Figures 4.11a, 4.11b, 4.11c are for five rounds of camouflage events.

On comparing the results of D-CENDA with CENDA, it is seen that the results are improved while giving significant overhead savings by using a shortened relative address. With dynamic adaptation on the selection of critical nodes, the nodes on high usage paths are better protected. Hence, those nodes which are not selected as critical initially are also covered for malicious node detection. Additionally, usage of progressive feedback in D-CENDA impacts the critical nodes selected while incorporating real time information as seen in Section 4.1.2.1. This further improves the 'all malicious node' detection in D-CENDA. Additionally, the speed and computational benefit of using a bit-array over bloom filter can be seen, although it consumes a little more memory.

The performance of D-CENDA is compared with the results from CHEMAS by Xiao *et al.* [12] and with intruder detection algorithm by Ngai *et al.* [54]. The simulation results in Figures 4.7a, 4.7b, 4.7c indicate that the performance is at par with the two methods while having the advantage of being able to detect and differentiate multiple attack types at the same time. Also, with CHEMAS [12], the success of detecting the malicious nodes drops off drastically with the increase in the number of malicious nodes. For CHEMAS, in a 400 node network, the detection success percentage (in parenthesis) for the number of malicious nodes 1(99%), 2(97%), 3(95%), 4(94%), 5(92%). It should be also noted that D-CENDA is a proactive architecture, hence the probability of catching the malicious node without any legitimate event packet loss is higher. The verification of the packet transmissions from the neighbor node helps in drastically reducing false positives in the case of sinkhole and selective forwarding attacks. Additionally, D-CENDA can prevent region segregation malicious attacks. With regards to conserving energy, based on the requirement or attack type anticipated, the verification part of the system can be turned on/off. As a by product, the system is able to detect any dead nodes due to energy exhaustion or due to environmental

conditions.

4.7 Summary

D-CENDA is a proactive architecture to detect malicious nodes in the sensor network. It can be used in complementary to an existing system and is controlled by the base station. A mobile-node based camouflage event generator scheme is used to overcome the problem of region segregation by malicious nodes trying to prevent event reporting packets from reaching the base station. This is important in case of sporadic events, wherein the base station cannot differentiate between non-occurrence and non-report of events. The camouflage event based malicious node detection system uses a bloom-filter based verification procedure before labeling a node malicious. A node-classification scheme based on the role and importance of the node in the network is provided. A light weight path marking system using shortened relative addresses to indicate the path traversed by the packet to reach the base station is presented. Using our shortened relative address gives up to 25% address length shortening compared to a relative address and over 60% address length shortening over using absolute address.

The effectiveness of the system when the network is under different attacks such as sinkhole attack, selective forwarding attack, sybil attack and wormhole attack is examined using simulations and the results demonstrate this. Additionally, composite attacks of the prior mentioned four attack types are also analyzed.

D-CENDA can identify close to 100% critical nodes under sinkhole attack, while identifying more than 93% of all compromised critical nodes in selective forwarding and sybil attacks. The false positive rate (incorretly assuming a node as malicious) is high because

we aggressively mark nodes as a suspect with assurance that they will be released during verification. D-CENDA can identify one of the two collaborating nodes under wormhole attack. The other collaborating node in wormhole attack was only receiving the traffic from its colluding partner and transmitting it to the base station, hence making it difficult to identify it as malicious. If both nodes are directing traffic to different parts, we had 89% detection success. Additionally, compared to existing systems, which identify attacks of single type, D-CENDA can differentiate attack of four types and in future this can be further enhanced to identify other attack types as well.

Chapter 5

Attacks on Cognitive Radio Network

In the chapters so far we have been primarily looking at data delivery security or providing privacy to the data generating device. With the data having been secured, we will look at improving the spectrum utilization of a special type of network called cognitive radio mobile ad-hoc network (CR-MANET) in the presence of malicious users. The basic premise behind increasing spectrum utilization is to have the cognitive devices make good decisions even when under attack by malicious users. In cognitive radio network, the primary users are the licensed users that have the prerogative to use the spectrum. The sole use of the spectrum by the primary user has led to its significant underutilization. This prompted the FCC to allow the secondary users to access the licensed spectrum in an opportunistic manner, but without interfering with the primary user. The secondary users equipped with a cognitive radio are allowed to use the spectrum only when it is idle, *i.e.* the spectrum is not currently occupied by a primary user. In order to not interfere with the primary user, it is important for the secondary users to correctly identify spectrum occupancy by the primary user. This is a complex problem that is made difficult by signal fading and hidden terminal problems faced by the secondary user. Akyildiz *et al.* detail in [57] how the multi-hop architecture, dynamic changing topology, and the absence of a central authority further adds to the complexity of efficient functioning of CR-MANET. Another problem with being mobile is the limited amount of energy available to the device. The devices in CR-MANET run on power supplied by a battery, which can last from a few hours to a

few days depending on the usage. Communication is one of the more resource intensive operations by a CR-MANET device, which has resulted in the use of low power radios to conserve energy which limits the transmission range. Hence, a solution for reliable spectrum sensing should be energy efficient and have low communication overhead. As a recourse, distributed cooperative spectrum sensing (DSS) to identify the presence of primary user signals is gaining momentum because of its robustness against hidden terminal problem and signal fading.

Reliable spectrum sensing and accurate data fusion for decision making are two important requirements for the distributed cooperative secondary user network to function optimally. Reliable spectrum sensing can be achieved by using a cooperative system collecting data from many devices, while accurate data fusion is required to segregate erroneous and malicious data and utilize only valid information in decision making. Invalid data can be spread by the devices under three circumstances: First, due to device malfunction; such a malfunction is called byzantine failure and by nature it occurs due to arbitrary failure of components. Second, due to signal fading, or the device facing a hidden terminal problem. Third, false spectrum information can be spread by devices with malicious intent. Such an attack is called spectrum sensing data falsification attack (SSDF) [58]. The intent behind SSDF is to force the data collector into making an incorrect decision about the primary user spectrum occupancy. A malicious user may do this to either disrupt the network or with selfish motives so that it can utilize the network for personal gains when the data collector makes an incorrect decision.

An incorrect spectrum occupancy decision can cause two types of errors: one is a false positive wherein the spectrum is identified to be occupied, but in reality the spectrum is vacant. This error causes the spectrum to be non-utilized leading to reduction in throughput

of the secondary user network. The other type of error is the false negative in spectrum detection, i.e. the spectrum is detected to be vacant while the primary user is transmitting over the spectrum band. This error can lead to the secondary user signal colliding with the primary user signal causing interference which is strictly prohibited and can lead to strong ramifications. Hence, it is important to accurately identify the primary user spectrum occupancy.

A network of mobile devices augments problems for distributed cooperative spectrum sensing. One reason being the continuously changing neighborhood due to device mobility. Another being the limited number of inputs available for attaining the final decision. Additionally, some of these inputs will be inaccurate due to the problems specified earlier or could be incorrectly reported on purpose by malicious users. A malicious users can hide its mal-activity under the changing neighborhood, hence it is important to be able to validate the inputs being used in the data fusion. We present three solutions to this problem. The first solution is 'Multi-fusion based Distributed Spectrum Sensing' (MFDSS). The second solution is called 'Recursive Neighbor Validation' (ReNVaS) and the third solution is called 'Tight Medoid Clustering' (TMC).

The organization of the chapter is as follows: The problem is defined in Section 5.1. Section 5.2 presents the multi-fusion based distributed spectrum sensing methodology which includes the framework and analysis (Section 5.2.2.1) and the results (Section 5.2.2.4). The recursive validation and clustering schemes are discussed in Section 5.3. Related work follows as Section 5.4 and finally the summary is presented in Section 5.5.

5.1 Network Model and Problem Statement

We first describe the network model and define the problem based on this model. The network consists of two types of devices: the primary user and the secondary users. There is one primary user that is licensed to use a particular spectrum. This primary user has a powerful transmitter having a transmission range of up to 100 kilometers. The primary user is represented as N. The secondary users are mobile ad-hoc devices that want to form a network (MANET) with other mobile ad-hoc devices and opportunistically use the primary user spectrum. The secondary users in the network are represented as n_x where x uniquely identifies each secondary user. The primary user is powerful and cannot be compromised. Compared to a primary user, the secondary users have limited resources such as energy, transmission range etc., and are prone to compromise. Also, being a MANET, the secondary user network is spread over a few square kilometers.

The secondary users are equipped with a cognitive radio, which allows them to sense the spectrum for the presence of the primary user signal. This is achieved by measuring the RSS. If the spectrum is identified to be vacant, the secondary user can utilize the spectrum. The secondary users being mobile and resource constrained, face the hidden terminal problem, signal fading or byzantine failures which could cause the secondary user to make an inaccurate spectrum occupancy detection. To overcome this, the secondary users use distributed cooperative spectrum sensing which could introduce inputs from malicious secondary users launching spectrum sensing data falsification attacks (SSDF). The malicious user launches a SSDF attack to convince other secondary users into making an incorrect decision about primary user spectrum occupancy.

The errors (intentional and unintentional) in reported signal (RSS) can be of three types:

signal measurement error, intentionally modified signal measurements reported by malicious users, and incorrect measurement reported by devices due to byzantine failures. The devices measure the signal in the spectrum which includes an error that can be caused by signal fading, hidden terminals, or inferior sensors. The error $e/2$ is defined as a maximum measurement error (in percentage) by a device, hence every device has a measurement error between $-e/2\%$ and $e/2\%$. The measured signal which is reported by device n_x is represented as $s_x(t)$. Let $s^*(t)$ be the actual signal strength. The malicious devices report a signal measurement with a maximum malicious modification of $f\%$ to the measured signal. Hence, in addition to the error in the measured signal, the malicious devices modify the reported signal as follows:

$$\boxed{s_x(t) = s_x^*(t) \pm (s_x^*(t) * ((0 : e/2) \pm (0 : f/2)))} \tag{5.1}$$

A device undergoing byzantine failure will report a signal which can have up to 100% error in it, but unlike a signal measurement reported by a malicious device, it is unintentional. It can produce incorrect measurement with error greater than 100%, but in such cases those inputs can be easily identified as outliers and removed from computation. Hence, we assume the byzantine failure devices to have errors up to 100%.

$$\boxed{s_x(t) = s_x^*(t) \pm (s_x^*(t) * (0 : 50\%))} \tag{5.2}$$

The goal of the malicious devices is to manipulate the data inputs so as to force incorrect decision making by neighboring devices. If $g_*(t)$ is the final decision by the neighbor $*$, the goal of the malicious device is defined in Equation 5.3.

$$\boxed{\begin{array}{l} \forall i \; where \; i \in \{neighbor\}, \; g_i(t) = 1 \; when \; PU = 0 \\ \exists i \; where \; i \in \{neighbor\}, \; g_i(t) = 0 \; when \; PU = 1 \end{array}} \tag{5.3}$$

Goal: The network goal is to correctly identify the primary user spectrum occupancy under signal fading, hidden terminal problems, byzantine failures, and in the presence of malicious secondary users.

5.2 Multi-Fusion based Distributed Spectrum Sensing

MFDSS includes three steps, namely, sensing data fusion, reputation propagation and fusion, and decision fusion. Unlike existing distributed schemes where a device presents only a local decision to its neighbor, in MFDSS the device transmit the actual measured sensing data to its neighbor. Collection of the actual data instead of a binary decision from its neighbor allows the data collector to pre-process it prior to data fusion. MFDSS presents the ability to detect extreme outliers and remove them before further processing. It is our understanding that to catch byzantine failure you require information with much higher granularity than having a binary decision information. Also, compared to centralized approaches, the amount of data inputs is limited in CR-MANET, hindering the use of statistical methods requiring large inputs. The effect of erroneous information due to SSDF attacks that escape detection during outlier detection are suppressed using a reputation scheme. During this, the remaining cohesive observations are weighed by their reputation and are fused to make a local decision about the presence of primary user signal. This is the data fusion step. This local decision data is exchanged with the neighbors and fusion of these decisions is performed to reach the final decision. This step is the decision fusion. Although this entire process occurs locally at

one hop radius requiring only one hop communication, it incorporates sensing information from two hop neighbors, thereby encompassing inputs from a larger area. The devices are mobile leading to changing network topology and changes in neighborhood. To cater to the changing dynamics, we present a reputation propagation and fusion method based on the reputation of devices and the timestamp of the last update of the reputation. This has three benefits: First, it provides an opportunity for devices to make robust decisions using propagated reputation of new neighbors. Second, the reputation of a malicious device is propagated which reduces its likelihood to hide its malicious activities under the changing neighborhood. Third, it helps maintain the freshness of the reputation information.

5.2.1 Architecture and Working Model

Multi-Fusion based Distributed Spectrum Sensing (MFDSS) is a semi-global sensing data collection scheme. The data collected in MFDSS is considered semi-global in regards to the CR-MANET secondary user network and not with respect to the primary user network. This data is fused and the intermediate resultant local decision is exchanged with neighbors for further fusion to make a final decision. The data collected from neighboring devices may contain incorrect or malicious elements that need to be fixed. We consider two important aspects while dealing with incorrect data. First, remove any extreme points in the data set using outlier detection. This takes care of incorrect data resulting from byzantine failures. Second is to suppress any remaining malicious data caused by a SSDF attack using a reputation mechanism. The reputation of the devices are maintained locally, but utilizing the inherent feature of mobility of the devices, the reputation is propagated opportunistically. MFDSS is a three step process consisting of sensing data fusion, reputation propagation and fusion, and decision fusion. Table 5.4 lists the notations used in MFDSS.

Table 5.1: MFDSS Table of Notations

Symbol	Definition
$s_m(t)$	Sensing results of device m at time t
$l_m(t)$	Data fusion result at device m
$d_m(t)$	Local data fusion decision of device m
$g_m(t)$	PU spectrum occupancy decision at m
r_{mi}	Reputation of device i at m
PET	Primary energy threshold
$F_1[\,]$	Sensed energy data fusion
$F_2[\,]$	Decision inputs fusion
MRI	Malicious reputation information propagation
FRI	Freshness of reputation information metric

5.2.1.1 Sensing Data Fusion

The secondary user sense the environment for the presence of primary user signal which can be of three types: energy detection, cyclostationary feature detection, or matched filter detection [59]. We use energy detection. The reason to use energy detection is because the secondary devices are not resource abundant in CR-MANET and the other methods require larger amount of resources.

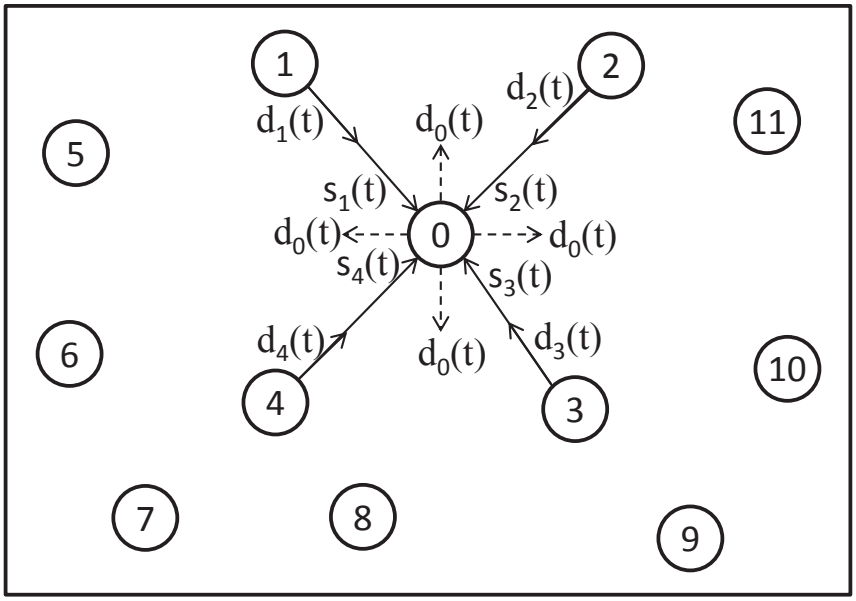


Figure 5.1: Multi-Fusion based Distributed Spectrum Sensing

Every secondary user collects the current sensed information from its immediate neighbors. This is depicted in Figure 5.1 which has 12 secondary users. For clarity, we only present one device performing the local data fusion, while in practice every secondary user performs this operation. There are no separate decision centers and every device acts as a decision center for itself. The secondary user 0 collects the sensed information at time t, namely $s_1(t)$, $s_2(t)$, $s_3(t)$, and $s_4(t)$ from its neighbors 1, 2, 3, and 4 respectively. There are two steps in the local decision making process. First, the secondary user needs to identify any outliers if present. Second, the collected data is weighed into using the reputation of the neighbor and used in the local decision making process.

During outlier detection and data fusion, the availability of the reputation value is a criteria used by the device to decide whether the sensing input should be included in outlier detection and data fusion. If the reputation of a neighboring device is not present, the device is put in an incubation period during which the inputs are not incorporated in the decision making process, but the device is given a reputation value based on their input and the final decision. This incubation period impacts a device during initial network joining, but plays a significant role in discouraging an adversary from taking different identities trying to avoid bad reputation or performing an attack similar to sybil attack. For example, if device 0 did not have reputation information about device 2, the input $s_2(t)$ will be excluded from outlier detection and data fusion, and its decision $d_2(t)$ excluded from decision fusion (described in Section 5.2.1.2) during the incubation period. This value $s_2(t)$ will be used as a basis for updating the newly initialized reputation value of device 2 at device 0. It should be noted that the device 2 is unaware whether its sensed result is used at device 0. This is because, device 0 may not have reputation of device 2, but device 2 could have received reputation information about device 0 through reputation propagation as discussed later in the section.

The length of the incubation period is selected randomly by the device and it ranges between 1 and ϕ time periods.

5.2.1.1.1 Outlier Detection The first step after collection of the energy samples from the neighbors is to remove the extreme outliers. Although a reputation based mechanism is good to suppress bad data, a device which has a good reputation but has failed recently can survive feeding incorrect information till its good reputation wears out. Till then it adversely impacts the local decision of the data collectors. Hence, outlier detection is required to remove any extreme values before the data fusion. A detailed study of outliers in statistical data is presented by Barnett *et al.* in [60].

There are two ways to handle outliers in a data set. One is to accommodate a modified value of the outliers. The second way is to discard it from future computations. In a cognitive radio network when neighboring devices detect and report PU signals which are far different, it is better to discard these extreme outliers since the measurement of energy values by immediate neighbors should not be too different. This can be done because we only discard the extreme outliers while the effect of any mild outliers present will be suppressed in further processing. To identify outliers we want a robust yet simple procedure to detect them. If the number of outliers can be estimated and the contamination proportion (defined as the ratio number of incorrect results to total sample size) is less than 0.21, we can use sample kurtosis and significance tables provided by D'Agostino *et al.* [61] as in Equation 5.4. This value of 0.21 only considers device failure rate as in this step we are looking at catching incorrect inputs due to byzantine failures. The extreme outliers are generally caused by byzantine failures and if the failure rate of the devices is known, the number of extreme outliers can be estimated. This works well for small sample sizes [60] which is characteristic

of our secondary user model.

$$\boxed{T = \frac{n\sum_{i=1}^{n}(s_i(t) - \tilde{s(t)})^4}{\left[\sum_{i=1}^{n}(s_i(t) - \tilde{s(t)})^2\right]^2}} \tag{5.4}$$

If we cannot estimate the number of outliers we propose to use the modified z-test made famous by Iglewicz *et al.* [62] wherein the median and the median absolute deviation are considered instead of the mean and the standard deviation respectively.

$$\boxed{\begin{array}{c} MAD(t) = \tilde{u} \;\; where\ u = \{|s_i(t) - M(t)|\} \\ Z_i(t) = 0.6745 \cdot \frac{(s_i(t) - M(t))}{MAD(t)} \end{array}} \tag{5.5}$$

For the collected data $s_i(t)$, the median (M) and median absolute deviation (MAD) are calculated as in Equation 5.5. The observations with absolute Z_i score greater than 3.5 [62] are considered as outliers and will be discarded from further computations. Both the methods are computationally inexpensive and suits well to function in a distributed CR-MANET. A simplistic overhead non-intensive outlier detection works because in this stage we are trying to catch only extreme outliers primarily caused by Byzantine failures.

5.2.1.1.2 Data Fusion Incorporating Reputation The remaining observations that survived the outlier detection are used to compute the local decision for the presence of the primary user signal. The reputation of the individual neighbor as maintained in the data fusing device is used to weigh the data. R_m is the reputation set maintained at a secondary user m. $R_m = \{r_{mi}, t_{mi}\}$, where r_{mi} is the reputation of secondary user i maintained by secondary user m, t_{mi} is the latest time period when r_{mi} was updated. If $s_1(t)$, $s_2(t)$, ..., $s_x(t)$ is the final set of observations and r_{m1}, r_{m2}, ..., r_{mx} are neighbor devices corresponding

reputation values belonging to a set R_m. The weights for each observation are calculated as in Equation 5.6.

$$\boxed{\begin{aligned} w_{mj} = & \frac{r_{mj}}{\sum_{i=1}^{x} r_{mi}} \Rightarrow \sum_{i=1}^{x} w_{mi} = 1 \\ & l_m(t) = \sum_{i=1}^{x} s_i(t) \cdot w_{mi} \end{aligned}} \tag{5.6}$$

The weighted mean and standard deviation for energy samples is calculated. The hypothesis to test for presence or absence of primary user occupation can lead to two types of errors with each type having a cost associated with it. The first type of error occurs when spectrum is identified as occupied by PU, when in reality the spectrum is unoccupied. This error reduces the efficiency of spectrum utilization by the secondary user but does not impact the primary user. The second error is the local decision being a vacant spectrum while the primary user is occupying the spectrum. This error is critical since it can interfere with the spectrum usage by the primary user which is strictly prohibited by the FCC. With this in consideration, we develop the hypothesis utilizing two primary energy thresholds (PET).

$$\boxed{\begin{aligned} H_0 : & \ (l_m(t) < PET_2) \ \ OR \\ & \quad ((PET_2 \leq l_m(t) \leq PET_1) \ AND \ (\sigma < EDT)) \\ H_1 : & \ Otherwise \end{aligned}} \tag{5.7}$$

EDT is the energy deviation threshold. The two primary energy thresholds are PET_1 and PET_2 where $PET_2 < PET_1$. The two primary energy thresholds will vary vastly depending on the distance of the CR-MANET from the primary user location and are closely bound to the error in the sensing measurement. The PET_2 is a hard threshold below which the

spectrum is identified to be unoccupied, but to be certain and to provide resiliency to non-interference with the PU, if the spectrum energy is estimated to be borderline (between PET_1 and PET_2), we verify the deviation in the inputs before marking the spectrum vacant. This is required to prevent a few mild outlier inputs impacting and producing a false-negative final decision. The final decision on the occupancy of the primary user is defined by the null hypothesis indicating the absence of PU occupancy ($d_m(t) = 0$) and is depicted in Equation 5.7.

5.2.1.2 Decision Fusion

The local decisions computed by each device is broadcast to its neighbors. After collecting the local decisions, each device performs a fusion over it to reach the final decision. MFDSS does this to incorporate the measurement by devices over two hops as compared to just one hop. This gives the devices inputs' over a larger area of sensed measurements. Figure 5.2 depicts an example scenario of how this is established. In the example, $F_1[x]$ is a data fusion function over energy measurement inputs x as depicted in Section 5.2.1.1 and is a fusion process using Equations 5.6, and 5.7. Similarly $F_2[x]$ is fusion over the local decision inputs x. $g_m(t)$ is the final decision on the occupancy of the spectrum by the primary user by device m at time t.

Consider the case of device 3 in the example depicted in Figure 5.2. It is immediate neighbors with devices 2 and 4 while it is two hop neighbors with devices 1 and 5. The first data fusion on energy samples by three devices is given in Equation 5.8. The result is exchanged by the neighboring devices and the final result calculated by device 3 is shown in Equation 5.9.

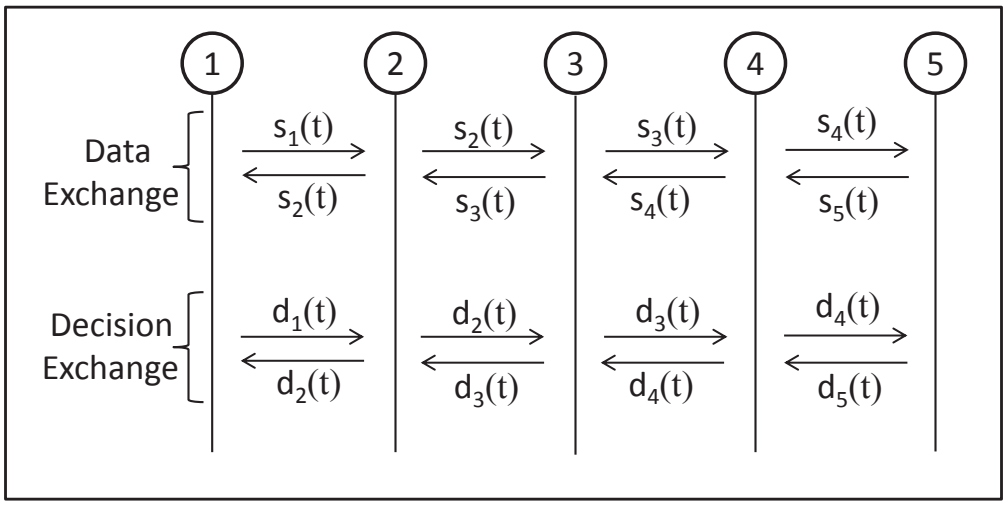


Figure 5.2: Decision Fusion Example

$$\boxed{\begin{aligned} d_3(t) &= F_1[s_2(t), s_3(t), s_4(t)] \\ d_2(t) &= F_1[s_1(t), s_2(t), s_3(t)] \\ d_4(t) &= F_1[s_3(t), s_4(t), s_5(t)] \end{aligned}} \tag{5.8}$$

$$\boxed{\begin{aligned} g_3(t) &= F_2[d_2(t), d_3(t), d_4(t)] \\ &\approx F_2[F_1[s_1(t), s_2(t), s_3(t), s_4(t), s_5(t)]] \end{aligned}} \tag{5.9}$$

It is seen that the final data fusion occurs over two hops giving us an area coverage of up to 9 times that achieved by a solo device taking decisions independently. When considering a MANET which is spread over a few square miles, this additional primary user signal sensing reach is significant especially in overcoming signal fading and hidden terminal problems. Additionally, it must be noted that all communication occurs between one-hop neighbors only. In MFDSS, to keep it light-weight and simple, we use majority fusion for making the final decision as in Equation 5.10.

$$g_m(t) = \begin{cases} 1 & if \ \sum(d_x) > (count(d_x) - sum(d_x)) \\ 0 & otherwise \end{cases} \tag{5.10}$$

Quantitatively, this decision making process is robust as it considers much larger set of inputs from devices. Let each device have a neighbors. In a 2 hop radius the number of neighbors are $(a \cdot (a-1))$. If the neighbors have b common neighbors, the total number of devices in the final decision process is $(a \cdot (a-1)) - (a \cdot b)$. Simulating a 200 device network spread uniformly over 2000×2000 area with a communication range of $250m$, the average number of one hop neighbors is 9 and the average number of two-hop neighbors is 23. We show in Section 5.2.2, how the additional inputs from the two hop neighbors along with reputation propagation makes MFDSS robust.

5.2.1.3 Reputation Management

Robust maintenance of reputation in a CR-MANET is as important as the actual sensing data collected. There has been a large amount of research on reputation management in MANET, P2P networks and also at the application layer such as in eBay. The primary goal of reputation mechanisms in MANET and P2P systems is to gather reputation information involving packet forwarding or delivery which is a visible and measurable entity. For example, a packet forwarded is overheard by another device which can vouch for the device forwarding packets (watchdog mechanism). In case of cognitive radio devices, the reputation is awarded based on the reported values of the spectrum measurement which cannot be easily verified as this value is affected by the environment. Our reputation propagation scheme inherits characteristics from the transitive property of Eigen-trust algorithm in P2P networks [63] where if a device i trusts a device j, i will trust the devices j trusts; and the trust management

using maturity based model [64]. The principles of maturity based model is used in the reputation propagation.

With regards to reputation, there are two types of devices, the assessor device and the assessed device. The assessor device maintains the reputation value of the assessed device. It should be noted that in MFDSS every device is an assessor of reputation of every other device. Reputation management consists of two parts. One, the reputation calculation wherein the reputation of the assessed device is calculated based on its input data and the final decision by the assessor device. The other is the proactive propagation of reputation.

Algorithm 4: Reputation Calculation

```
if s_i(t) < PET_2 then
    d_i(t) = 0
else
    d_i(t) = 1
if s_i(t) ∉ outliers then
    if d_i(t) == g_m(t) then
        r_mi = r_mi + 1
    else
        if d_m(t) == g_m(t) then
            if g_m(t) == 0 then
                r_mi = r_mi - 1
            else
                r_mi = r_mi - 2
else
    r_mi = r_mi − 1
time_mi = t
```

5.2.1.3.1 Reputation Calculation The reputation of the neighborhood devices are updated as depicted in Algorithm 4. m is the device-id of the assessor device and i is the device-id of a neighbor. $s_i(t)$ is the value of the report from the neighbor device i. $d_m(t)$ is the local sensing decision and $g_m(t)$ is the final decision at device m. Let r_{mi} be the current reputation of device i at device m. A device reporting an outlier value is penalized

by reducing its reputation value by 1. A device reporting a value corresponding to non-occupancy of the spectrum by PU, but the fusion result is PU presence, it is penalized more for the false-negative input compared to a false-positive. A false-negative on PU occupancy can lead to interference with the PU spectrum usage. Such interference is unacceptable given the FCC guidelines while compared to a false-positive which may only reduce the opportunity of the SU to utilize a vacant spectrum.

5.2.1.3.2 Reputation Propagation The secondary users are mobile and the neighborhood changes frequently. The new neighborhood of the user will consist of some secondary users that were neighbors at an earlier time period and some new users who become neighbors for the first time. Additionally, the reputation information of some devices could be obsolete and may need to be updated. To improve the efficiency, we use a reputation propagation mechanism in which a device will import the reputation information from a neighbor if the reputation of the neighbor as seen by this secondary user is above the reputation threshold (τ) and the reputation value is fresh i.e. the reputation was updated in the last γ time-periods. On receiving the reputation information, device m performs a fusion operation on all the reputation sets it received. The fusion operation performed is depicted in Equation 5.11.

$$\boxed{\begin{aligned} r_{mi} &= \langle r_{pi} \rangle \quad | \quad ((t - t_{pi} < \gamma) \text{ AND } (r_{mp} > \tau)) \\ t_{mi} &= \lfloor \langle t_{pi} \rangle \rfloor \quad | \quad ((t - t_{pi} < \gamma) \text{ AND } (r_{mp} > \tau)) \end{aligned}} \tag{5.11}$$

5.2.2 Analysis and Results

In this section we analyze the ability of MFDSS to withstand different attacks and handle incorrect information propagated due to byzantine failures or hidden terminal problems. It

is generally accepted that incorporating any security feature in a CR-MANET will incur some overhead. Hence, we analyze the additional overhead due to the scheme and present its justification. The security analysis involves studying the impact of the different attack scenarios as well as the ability of the devices to withstand byzantine failures. For this purpose we simulated CR-MANET in Matlab. The secondary user network is spread over an area of 2000×2000 $meters$ and the number of secondary users varied as 50, 100, 150, and 200. The devices are loosely time synchronized and stay at a location for 10 time units before they move to a randomly selected location within a distance of 250 $meters$. We run the simulation for 1000 time units. The presence of the primary user follows a beta distribution [65]. This gives us a realistic model to study the malicious activity given the primary user presence following a real-world distribution. We first present the MFDSS analysis, then overhead analysis followed by parameter analysis, and finally the simulation results.

5.2.2.1 MFDSS Analysis

There is research which considers the usage of cognitive radio networks in a mobile ad-hoc network but without considering the mobility aspect of the devices. The aspect of not considering mobility while solving a CR-MANET problem is a very strong assumption which is not acceptable since mobility of devices leads to changing neighborhood. Any form of a reputation system in a continuously changing topology will fail as it allows the malicious devices to hide behind the changing neighborhood. Hence, we first analyze the probability of malicious devices to hide behind the changing neighborhood.

5.2.2.1.1 Malicious Reputation Information Propagation The total number of secondary devices is set to 50, 100, 150, and 200, while there is only one malicious device. Every

device except the malicious device moves a random distance up to 250 *meters*, but the malicious device always moves 250 *meters* to get as far away from the current neighborhood as possible. This is done for the cases with traditional reputation maintenance systems and compare it against MFDSS.

Figure 5.3 presents the proportion of neighbors having reputation information about the malicious devices called Malicious Reputation Information (MRI). We define MRI as the percentage of neighboring devices aware of the correct reputation of the malicious device. The MRI metric indicates how easily a malicious device can hide in a neighborhood with a lower value representing higher success. It can be clearly seen that in a CR-MANET with mobility, propagating the reputation of the devices tremendously reduces the possibility of the malicious devices hiding behind the changing neighborhood. Figure 5.4 depicts the same scenario as MRI, but shows the number of devices in the network that are aware of the presence of the malicious device in the network. It shows the availability of malicious device's reputation information at other devices. It can be seen that MFDSS does very well compared to a non-propagating reputation scheme.

5.2.2.1.2 Freshness of Reputation Information The freshness of reputation information (FRI) metric is important since a device may have received bad reputation for incorrect sensing due to signal fading or hidden terminal problems and not due to any malicious behavior. Unlike a static network, since the devices move in CR-MANET, the issues due to hidden terminal and signal fading can change and the devices may be able to attain valid sensing results thereby improving their reputation. In reputation non-propagation schemes, a device may contain the reputation but it could be stale. An incorrect stale reputation is worse than having no reputation at all. Figure 5.5 presents the percentage of average

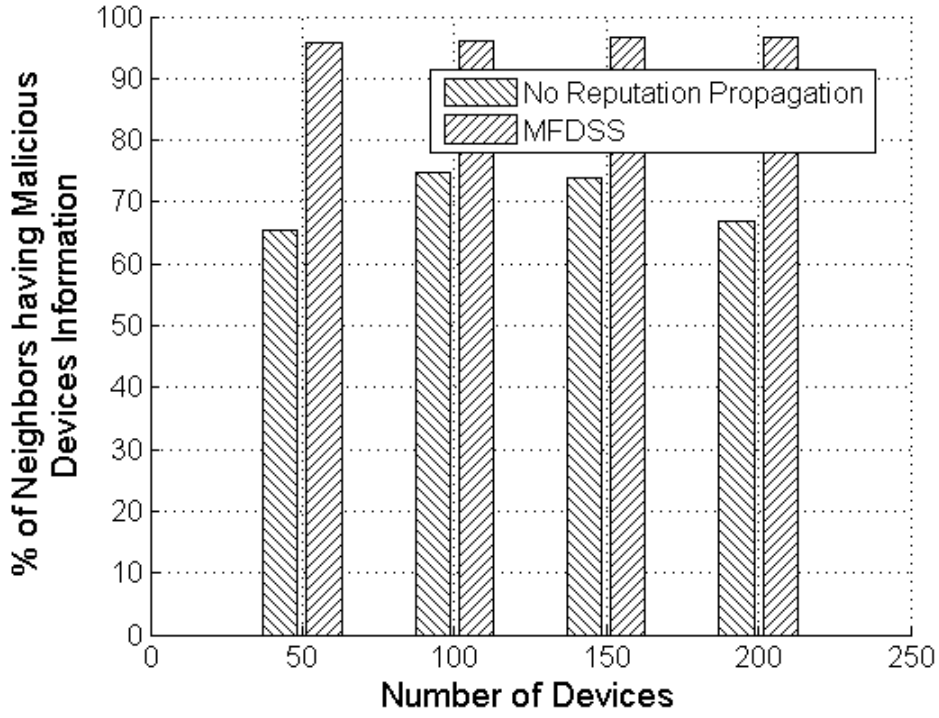


Figure 5.3: Malicious Neighborhood Information

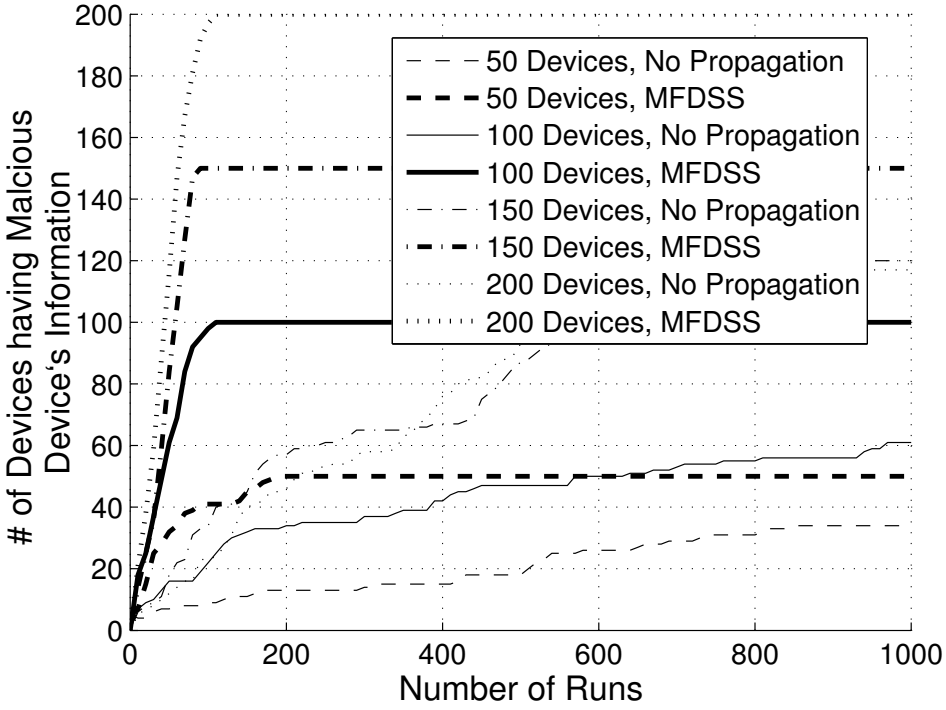


Figure 5.4: Malicious Device Information

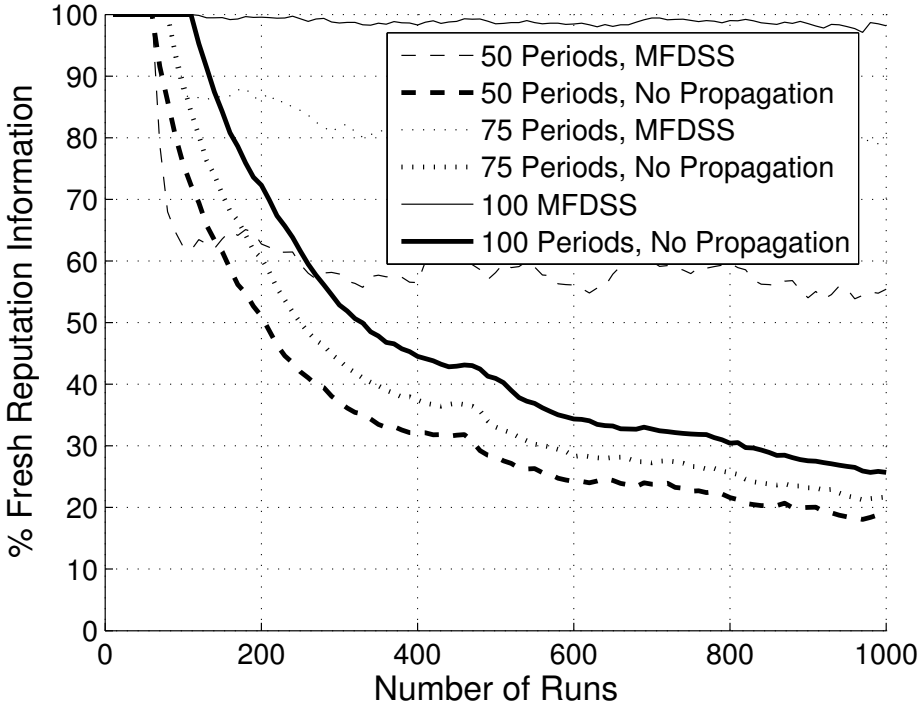


Figure 5.5: Freshness of Reputation Information

FRI metric in the network with MFDSS reputation propagation scheme as compared to a reputation non-propagation scheme. There are three freshness values considered 50, 75, and 100 units. When considering 50, if a device has reputation information which is older than 50 time units, then it is considered stale. We can see from the figure that in a reputation non-propagation scheme, the overall freshness of the information deteriorates very quickly. While considering information up to 100 time periods old as fresh, in MFDSS more than 95% of reputation information is fresh.

5.2.2.1.3 Malicious Neighborhood Probability The multi-fusion model covers a larger area that adds significantly to the resiliency of distributed sensing under device failures and attacks. Consider a network of n devices. Let p be the percentage of malicious devices. The total number of malicious devices is $\frac{p \cdot n}{100}$. In a simple majority scenario, if the

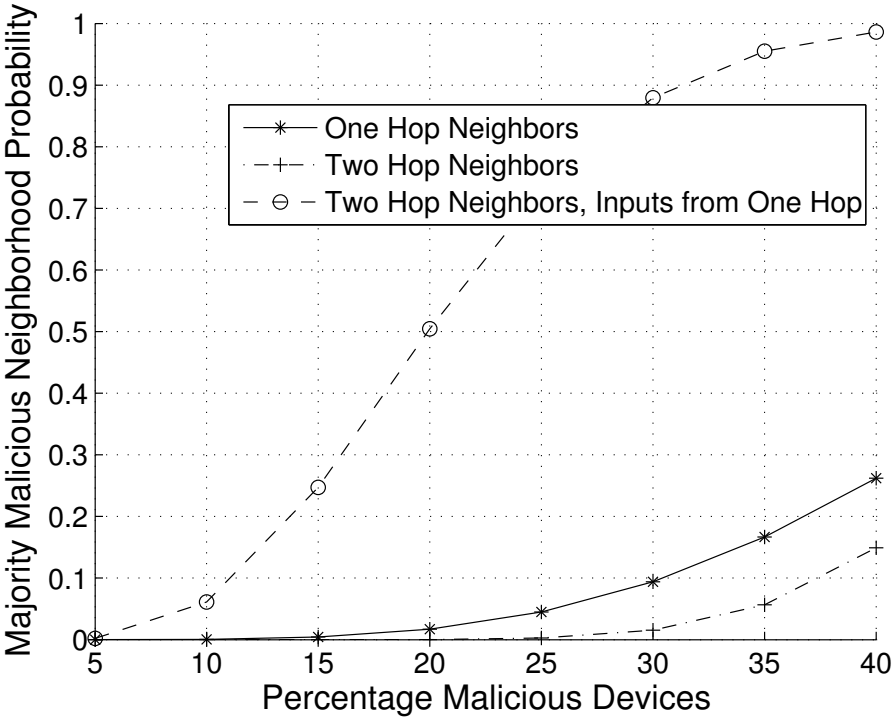


Figure 5.6: Malicious Neighborhood Probability

number of malicious devices are greater than half the inputs for fusion, it results in a wrong decision. Let s be the average neighborhood size, then the probability of having greater than $s/2$ malicious devices in the neighborhood is $\sum_{m=\frac{s}{2}}^{s} \frac{\binom{\frac{n \cdot p}{100}}{m} \cdot \binom{n-\frac{n \cdot p}{100}}{n-m}}{\binom{n}{s}}$.

The results for a network with 200 devices spread over an area of 2000 × 2000 meters with a communication range of 250 *meters* was simulated and found to have an average neighborhood of 9 devices and a two hop neighborhood of 23 devices. To have a complete influence when considering a majority fusion, there needs to be greater than half the devices that are malicious. So, when considering one hop neighborhood, the number of malicious devices should be 5 or more and when considering two-hop neighborhood, it should be 12 or more. Figure 5.6 presents the results of probability of occurrence of malicious devices in either one-hop or two-hop neighborhoods.

The plot for one hop neighbors in Figure 5.6 shows the probability of finding 5 or more

malicious devices in a one-hop neighborhood for a random distribution of devices. The two hop neighbors plot shows the probability of finding 12 or more malicious devices in a two hop neighborhood. We see that considering a two-hop neighborhood definitely provides a higher resiliency against malicious SSDF attacks. The devices are all mobile and although we do not consider a collaborative attack model, it is interesting to consider if the malicious devices can collaborate with other malicious neighbors and the malicious devices could all move in to one hop neighborhood so as to gain a majority strength and dupe a good device into making a wrong decision. We calculate the probability of finding enough malicious devices within a two hop neighborhood so as to be able to gain a majority in a one-hop input scenario and see that any model based on just collecting inputs from one hop neighbors will deteriorate drastically as the number of total malicious devices in the system increases. This result is plotted as 'two hop neighbors, inputs from one hop' in Figure 5.6.

5.2.2.2 Overhead Analysis

In overhead analysis we consider the additional overhead required by MFDSS on top of the regular overhead needed for a practical functioning of CR-MANET. This would encompass the following steps leading to additional overhead.

5.2.2.2.1 Sensing Data Exchange and Fusion In the most basic CR-MANET, where each device senses and makes decisions by itself, there is a possibility for the device to make a wrong decision due to the following factors: incorrect sensing due to fading, hidden terminal, etc. Hence, the basic CR-MANET for comparison with MFDSS uses distributed spectrum sensing, but only considers fusion of the decisions from the immediate neighborhood devices. When considering the decision instead of actual sensed value under byzantine failure of

the devices, the device is still correct 50% of the cases i.e., if a device randomly responds with a PU-present or PU-absent decision without any malicious intentions, it is still correct 50% of the times. This becomes critical when PU presence is reported as PU-absent. The probability of catching and marking a result due to byzantine failure is high in MFDSS since we do not lose granularity of information collected which would happen when reducing it to a binary value. It should be noted that when considering just this step, there is no additional overhead.

5.2.2.2.2 Reputation Data Exchange and Fusion Correct and up to date reputation management of the devices is important for proper functioning of the model and this incorporates overhead in the form of storage space, exchange of reputation information, and the fusion process. In our model, we consider storing two parameters (reputation and timestamp) for each device with reputation parameter taking a byte while the timestamp is 2 bytes long. The device-id is 2 bytes long. Hence, in a n device network, each device may need to store at worst $5 \cdot (n-1)$ bytes of reputation information. Even if the number of devices increases, this storage space only increases linearly. This is not an issue for mobile devices with laptop or a tablet PC form factor, but when considering resource constrained devices like smartphones, GPS, personal digital assistant, etc., this increase in storage space requirement can be detrimental, but still workable since even a 1000 device network (considered large for CR-MANET) only requires $5KB$ of storage space.

The next factor to consider is reputation propagation and the overhead involved. The number of times the reputation set gets exchanged depends on the mobility of devices. We would like to note that the reputation set is only exchanged between one hop neighbors, hence a device reaching a new neighborhood can broadcast its reputation set to its neighbors.

Additionally, the reputation set can piggyback the sensing data packet as a device will only require this reputation set before performing a sensing data fusion, thereby eliminating the overhead for reputation propagation.

5.2.2.2.3 Decision Exchange and Fusion The research studies so far consider only one fusion, be it the sensing data fusion or decision fusion. MFDSS gathers information over a much larger area, thereby providing resiliency against hidden terminal and signal fading. The overhead involved in performing such a fusion can be considered additional and when compared to a single fusion model it is double. When comparing the overhead with a centralized model, where each device responds back to the base station (central authority), it is still less because, even as the network grows, the exchange of information happens only between one hop neighbors in MFDSS. In the centralized approach the number of hops the packet travels is governed by the path length which grows with network size. Also, each device consumes the same amount of energy, hence having fairness in energy consumption compared to the centralized approach in which the devices on the high traffic path will expend more energy forwarding packets. All the three exchanges are broadcast over one hop only. Hence, not considering the actual time the transmitter has to transmit (depending on the data length in the packet), we can assume the three transmissions to consume energy of the same order (let this be E_X). Also, the sensed data transmission and reputation set transmission can be achieved in the same packet. This gives us the total transmission overhead of $2 \cdot E_X$, which is twice that of basic CR-MANET.

Table 5.2: Energy Measurement Range for Different Error Rates

Error (%)	Primary User Absent	Primary User Present
10%	$[0, 1.1 * \epsilon]$	$[0.9 * \epsilon, 2 * \epsilon]$
20%	$[0, 1.2 * \epsilon]$	$[0.8 * \epsilon, 2 * \epsilon]$
30%	$[0, 1.3 * \epsilon]$	$[0.7 * \epsilon, 2 * \epsilon]$
40%	$[0, 1.4 * \epsilon]$	$[0.6\epsilon, 2 * \epsilon]$

5.2.2.3 MFDSS Parameter Analysis

We performed simulations with varying parameter values to study their impact on the performance of MFDSS. There are two primary energy threshold values in MFDSS. Primary energy threshold-1 (PET_1), which is the higher cut-off parameter is used to ascertain the presence of the PU signal. Primary energy threshold-2 (PET_2) parameter is instrumental in deciding the vacancy of the spectrum and plays a more critical role than PET_1 since an occupied spectrum if resolved to be unoccupied by the secondary user can result in interference with the PU which is not acceptable. Hence, we study the impact by varying the PET_2 values while keeping all the other parameters constant. PET_1 was set a value of $1.1 * \epsilon$, and an error rate of 10%, where ϵ is the energy threshold. The values for PET_2 was varied as follows: $0.8 * \epsilon$ $(\epsilon - 2)$, $0.9 * \epsilon$ $(\epsilon - 1)$,ϵ, $1.1 * \epsilon$ $(\epsilon + 1)$ wherein, when the value of PET_2 is $1.1 * \epsilon$ it is equal to PET_1. The number of malicious devices was also varied as 10%, 20%, 30%, 40% and the results are plotted in Figure 5.7. The malicious devices respond with random energy value within the error bounds for the sensing data input and since we are studying the impact of PET_2, the decision input from the malicious device is the decision fusion result. It can be seen that MFDSS performs the best when the PET_2 value is $\epsilon - 1$ which can be attributed to the error amount 10% in reporting the values by the neighborhood devices.

To study the performance correlation between the error in signal strength measurement to the primary energy threshold values we varied the PET_2 values as $0.7 * \epsilon$ $(\epsilon - 3)$, $0.8 * \epsilon$

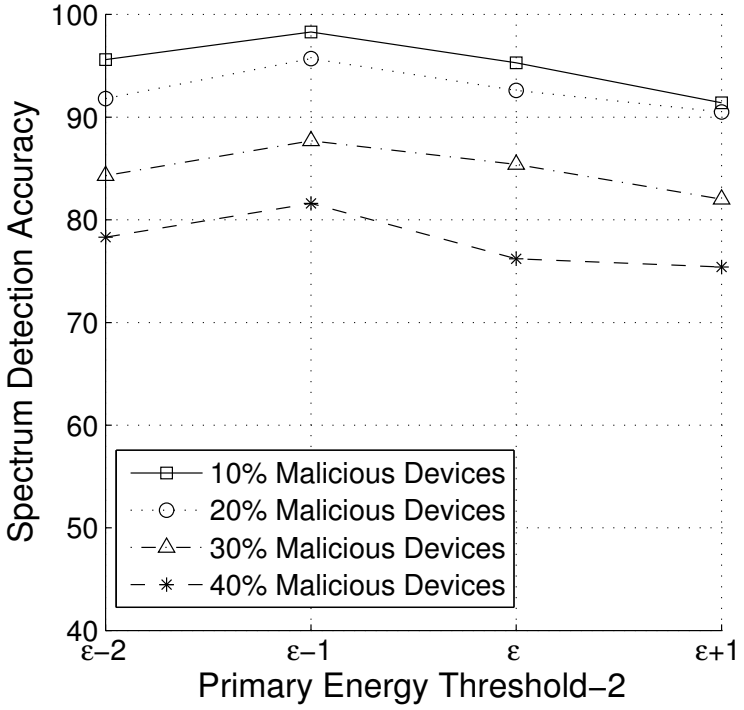


Figure 5.7: Primary Energy Threshold Sensitivity

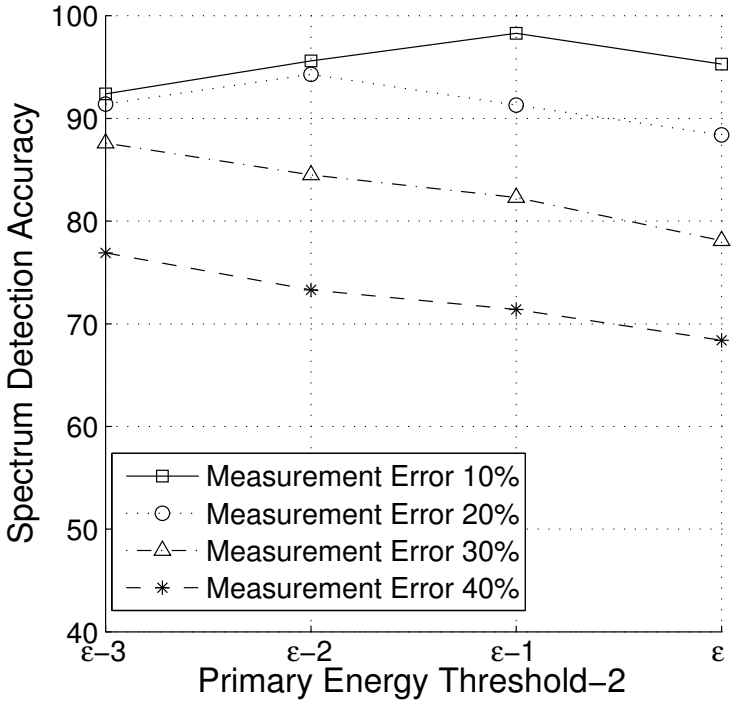


Figure 5.8: PET with 10% Malicious Devices

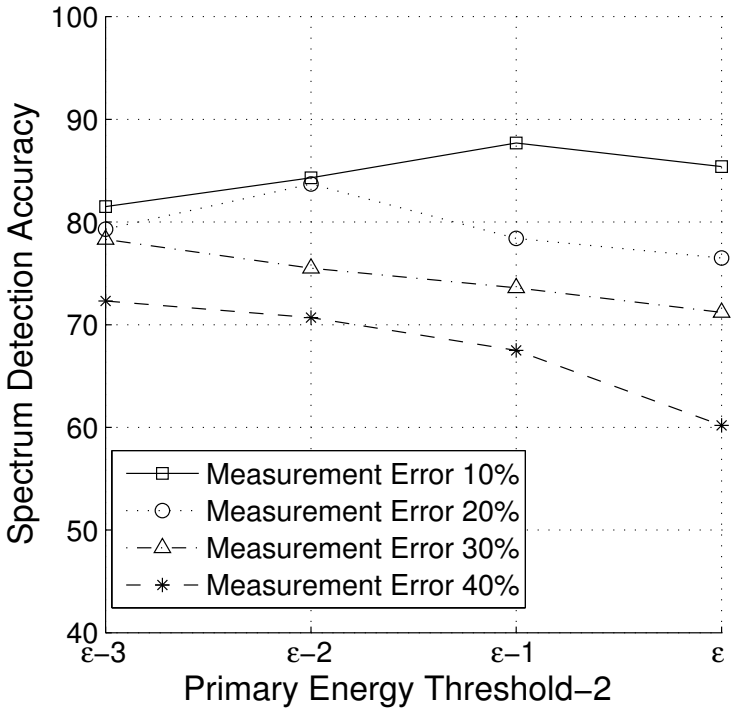


Figure 5.9: PET with 30% Malicious Devices

$(\epsilon - 2)$, $0.9 * \epsilon$ $(\epsilon - 1)$,ϵ (PET_1 value set at $1.1 * \epsilon$) while varying the error rate in the measurement as given in Table 5.2. We performed it for different amounts of malicious devices (10%, 20%, 30%, and 40%) and saw a similar trend in all four cases. We present results for 10% and 30% malicious devices as depicted in Figures 5.8 and 5.9 respectively. It can be seen that the primary energy threshold value has a strong correlation with the measurement error and is not dependent on the amount of malicious devices. For example, in Figure 5.8, we get the most accurate spectrum sensing for PET_2 value of $0.8 * \epsilon$ when the measurement error is 20%. Hence, if the amount of measurement error can be estimated (out of scope of this thesis) which is generally dependent on the enviromental factors and the measuring sensor, the PET_2 value can be set accordingly.

5.2.2.4 Results

We simulate the functioning of MFDSS under different attack scenarios. The number of devices is 200, and the devices move every 10 time-periods to a random destination within 250 meters.

Table 5.3: Malicious Attack Scenarios

Sensing Data Input	Decision Input
Report energy value of PU signal absence	PU absent (0)
Report energy value of PU signal absence	PU present (1)
Report energy value of PU signal absence	Data fusion result
Report energy value of PU signal presence	PU absent (0)
Report energy value of PU signal presence	PU present (1)
Report energy value of PU signal presence	Data fusion result
Report random energy value	PU absent (0)
Report random energy value	PU present (1)
Report random energy value	Data fusion result

MFDSS incorporates three fusion phases: data fusion, decision fusion, and reputation fusion. The malicious devices could present an incorrect input to the data fusion phase, or the input to decision fusion could be with malicious intent. The different combination of attacks studied is depicted in Table 5.3. The table features two columns, the first column (sensing data input) corresponds to the sensed input provided by the malicious devices to its neighbors. This can be of three types, namely, PU signal always absent, PU signal always present, and a random value. The second column or the decision input corresponds to the decision value a malicious device shares with its neighbor and it can be PU always absent, PU always present, or the actual data fusion result.

The first set of results we present compares the performance of the outlier detection strategy when strong outliers are present due to Byzantine failures. In the Figure 5.10 we present two cases based on if the byzantine failure rate can be estimated. For this purpose

we varied the Byzantine failure rate between 2% and 10% in increments of 2%. It is seen that the sample Kurtosis method performs slightly better than the modified z-test when the estimate can be done while the modified z-test performs better in the case when the Byzantine failure rate cannot be estimated. For the remaining of the results, we assume that the byzantine failure cannot be estimated and hence use modified z-test.

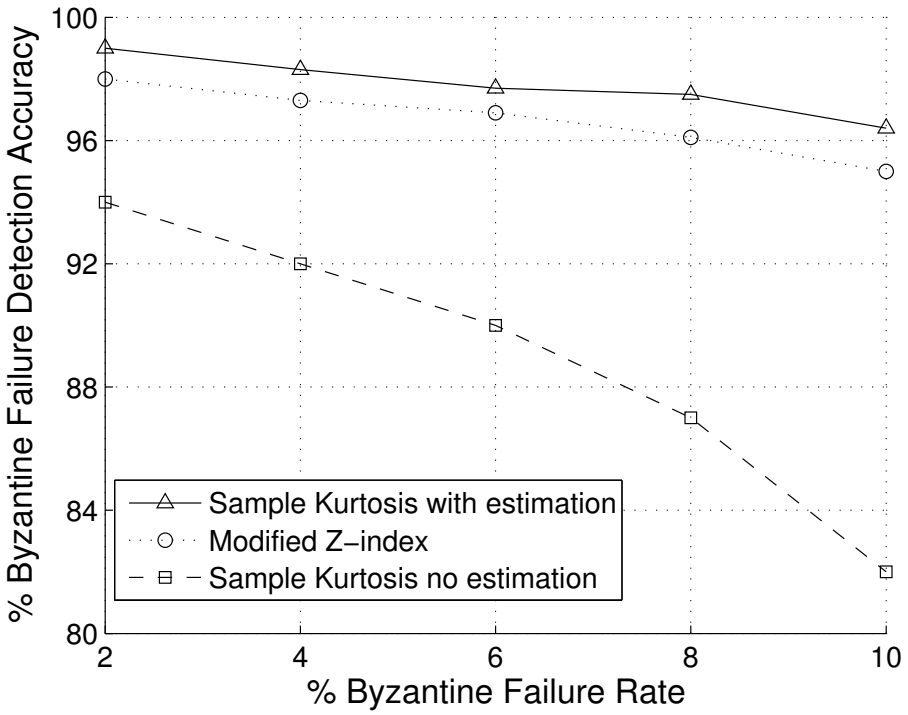


Figure 5.10: Byzantine Failure Outlier Detection

The number of malicious devices is varied between 10% and 50%, with up to 2% of all devices undergoing byzantine failure. A device undergoing byzantine failure could be either a good or a malicious device. A malicious device also performs a sybil attack and changes its identity with 30% probability every 100 time periods. The simulation is run for 1000 time periods and the devices move randomly every 10 time periods. The results of the simulation is presented in Figure 5.11. We also did some simulations with number of malicious devices under 6%, but do not present the results as the detection accuracy is upwards of 99% and

close to 100% for such cases. The results show that when the malicious users are 30% or lower, we have better than 83% detection accuracy and after that as the number of malicious users increase, the detection accuracy reduces but gracefully with having better than 65% accuracy even when the number of malicious users is 50%. The detection accuracy starts degrading faster when the number of malicious devices are greater than 50%.

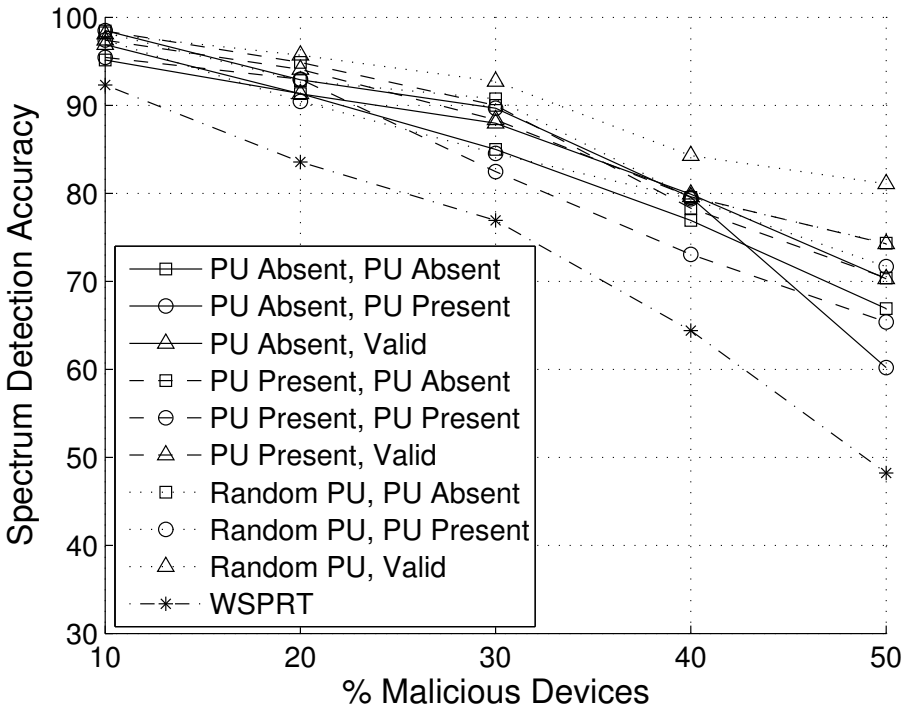


Figure 5.11: Spectrum Detection Accuracy

The Figure 5.11 shows the results of MFDSS compared with WSPRT model by Chen *et al.* [66, 67] and we see that MFDSS outperforms WSPRT by a wide margin. When the number of malicious devices are greater than 30%, we see a drastic drop in performance of WSPRT. This is because in its current form, WSPRT is not completely suited to a CR-MANET. WSPRT calculates apriori probability for miss detection and false positives based on the device location and uses the path loss model depending on the environment which may not be accurate. Second and a more important factor is the reputation of the device and

number of samples required. Due to the mobility of the devices, in WSPRT the secondary user does not have enough opportunity to collect the samples from devices about which it had reputation information, thereby impacting the results tremendously. The third factor impacting the performance of WSPRT was the ability of the malicous device to change identity to appear as a new device and take advantage of non-availability of reputation information with the SU device in WSPRT. These factors are addressed in MFDSS which shows in the resiliency of results especially when the number of malicious devices are high (close to 50%) and degrades gracefully beyond that.

5.3 Recursive Validation and Clustering for Distributed Spectrum Sensing

In the recursive validation method every device maintains a virtual cluster of all the devices with which it has been neighbors with in the past. The set of devices in the virtual cluster are used to validate the input from the neighborhood devices before they are incorporated in the decision cluster. In the tight medoid clustering scheme, the inputs from the neighbor are recursively clustered to find the tightest bound set of inputs and these get added to the decision cluster. Since the neighborhood changes frequently, the medoid clustering scheme may not incorporate any reputation information. The data in the decision clusters are fused to make a decision about spectrum occupancy. We present two fusion algorithms: non-unified data fusion and unified data fusion. Results for two types (based on different dynamism) of networks is presented to verify the performance of the model. ReNVaS performs well in a network with low churn but the performance falls when the network memberships changes frequently. Non-unified and unified fusions using ReNVaS and TMC give better performance

Table 5.4: ReNVaS Table of Notations

Symbol	Definition
n_x	Secondary user x
$s_x^*(t)$	True signal strength at device x at time t
$s_x(t)$	Sensing results of device x at time t
e	Measurement error
f	Malicious device signal modification
l_x	Neighborhood list of device x
VC_x	Virtual cluster of x
DC_x	Decision cluster of x
VL_i	Validity List of i
∇	Min neighbors for validation

in a high churn network with unified fusion keeping the false negatives in spectrum detection to a very low value.

5.3.1 Recursive Validation and Clustering Schemes

The basic idea is to provide a low cost yet highly efficient cooperative scheme to make a correct decision in a completely distributed manner under poor environmental conditions, presence of malicious devices, and byzantine failures. We present two device input selection strategies that will be used in the decision making process. The first method is called recursive neighbor validation scheme (ReNVaS) and the second method is called tight medoid clustering (TMC). ReNVaS uses past neighbor information to form a virtual neighborhood cluster that is used to validate the current inputs. The tight medoid clustering strategy uses the partitioning around medoid clustering algorithm recursively in order to find the tightest bound set of inputs. The resultant set of device inputs from the two methods is called the decision cluster. Following this, we present two data fusion methods utilizing the decision cluster to make an accurate decision in a distributed manner.

5.3.1.1 Recursive Neighbor Validation Scheme

In a CR-MANET, the devices are mobile resulting in continuously changing neighborhood. This presents situations in which many devices in the neighborhood are either unfamiliar or have not been a neighbor in a long time resulting in the information provided by these devices to be unreliable. To solve this unfamiliarity scenario we present forming virtual neighbor clusters. The purpose of members of this virtual neighbor cluster is to provide validation to the inputs.

5.3.1.1.1 Virtual Neighbor Cluster The virtual neighbor cluster is a device cluster set maintained at each device that comprises of the neighborhood devices with which the device came in contact with during its movement through the network. This virtual neighbor cluster set continuously changes due to the mobility of the devices. The virtual cluster at a device is represented using a m-tuple where each entry represents a set of information about each device in the virtual cluster. A virtual cluster set at a device x is represented as follows:

$$\boxed{\begin{aligned} C_x =& (c_{x1}, c_{x2}, c_{x3}, ..., c_{xm}) \\ & Where\ c_{xm} = (d_m, r_m, t_m) \end{aligned}} \tag{5.12}$$

where d_m is the device-id of device m, r_m is the reputation information of device m as seen by this device, and t_m is the time-stamp of the last update about the reputation of device m.

Maintaining the virtual neighbor cluster The devices are mobile and they move around in the network area. As the device navigates through the network it comes in contact with a number of devices. These devices with which it comes into contact can belong to one of the two types. The first type is of devices which are new neighbors and are coming into the

vicinity of each other for the first time. When this happens it adds a new tuple to its virtual cluster as follows:

$$\boxed{\begin{aligned} C_x =& C_x + c_{xm} \\ & Where\ c_{xm} = (d_m, 1, CurrentTimestamp) \end{aligned}} \tag{5.13}$$

The second type is when it comes across devices which were neighbors in the past. These devices already have an entry in the virtual cluster and it is updated as follows:

$$\boxed{\begin{aligned} C_x[m] =& (d_m, r_m + (-1)^q, CurrentTimestamp) \\ & where\ q = 0\ if\ s_m(t) = FinalDecision \\ & q = 1\ Otherwise \end{aligned}} \tag{5.14}$$

Over the period of time there will be memberships in the virtual cluster that become stale. Those stale tuples are not discarded immediately but they will be utilized for validation only if the timestamp is within a set number of past time periods.

5.3.1.1.2 Decision Cluster The decision cluster is a set of inputs collected at a device based on the reports that it receives from the neighboring devices. The purpose of forming a decision cluster is to shortlist a subset of device inputs which will be used to perform the fusion and make a decision on the primary user presence. All devices sense the spectrum for primary user signal. This primary user signal measurement is exchanged with the neighboring devices. So, along with its own signal measurement every device has the inputs from its neighbors.

We have two types of devices: the requester device and the validated device. The requester device is the device that is currently collecting the inputs from all the neighbors, and

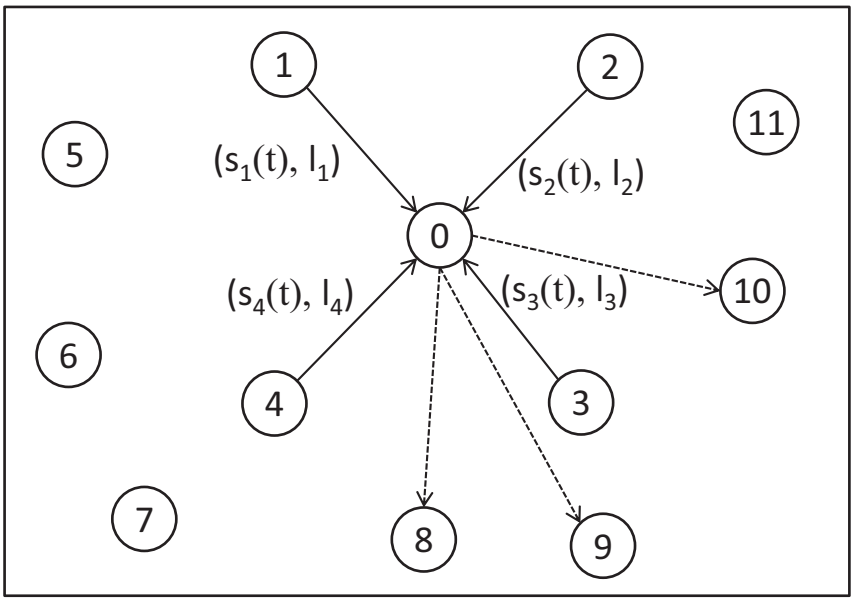


Figure 5.12: Recursive Neighborhood Validation

the validated devices are its neighbors considered one at a time. In the recursive neighbor validation scheme each input report is validated using the neighbor reports belonging to the validated device which are in the virtual neighbor cluster of the requester device. In the first step, the requester device request inputs from its neighbors. The neighboring device provide their primary user signal energy measurement inputs along with the device information (device-id's) about their immediate neighbors. On receipt of this information, the requester initiates a dialog with the devices in the neighbor list provided by the validated device provided they satisfy the following conditions: The device should be in the virtual neighbor cluster of the requester device and the reputation and last time of update of the device should be within the parametric threshold values as seen by the requester device. A malicious node could do a selective forwarding attack and may try to only forward those neighbor device information whose measurements concur with its own measurement. There is a good possibility that those devices whose measurements concur with the validated device are malicious and have poor reputation information at the requester device or have no

reputation. In either case, such devices will not be polled to provide their measurements to validate the device input. Hence, such a selective forwarding attack will fail.

Consider the network shown in Figure 5.12. There are 12 devices and each of the device will be performing the formation of decision cluster. For ease of understanding, we present the decision cluster formation for device 0. It receives the spectrum sensing information from all neighbors and the neighborhood set of each neighbor respectively. In this example, device 0 receives the following information from device 3: $(S_3(t), l_3)$ where $l_3 = (0, 8, 9, 10)$.

Algorithm 5: Recursive Neighborhood Validation

```
Input: n → all secondary users
for n_x ∈ n do
    Sense spectrum for PU signal measurement (s_x(t))
    Exchange s_x(t) and l_x(self neighborhood list) with neighbors
    for each neighbor n_i of n_x do
        NumRecursion = 1
        FailValidity(n_i) = 1
        while ((FailValidity(n_i)) AND (NumRecursion <= RecursionThreshold)) do
            for each n_y ∈ l_i do
                if n_y ∈ VC_x then
                    if n_y is good then
                        VL_i = VL_i + n_y
            // CheckValidity(n_i, VL_i) in Algorithm 6
            if CheckValidity(n_i, VL_i) then
                DC_x = DC_x + n_i
                FailValidity(n_i) = 0
            else
                FailValidity(n_i) = 1
                NumRecursion = NumRecursion + 1
                l_i = l_i + NumRecursion-hop neighbors of n_i
```

Upon receiving the information, device 0 requests the sensing information from those devices in l_3 for whom there is information in the VC_0 and the reputation information as maintained is above a set threshold level and the tuple is not stale. If the conditions are met, the requester device asks for the energy measurement input and its neighborhood list

from the device. In the example, it will be from devices 8, 9, and 10 (device 0 being itself). This process is repeated for all the devices in the neighborhood list provided by the neighbor device. This subset of devices which satisfy the conditions are used to substantiate the input from the validated device. It is particularly important to use the neighborhood information to validate each input because devices which are further apart may see different readings due to signal fading or obstruction. If the input is validated, it is added to the decision cluster. If the input is not validated, a second round of inputs is gathered to validate the input from the neighbor which encompasses 2-hop neighbors. This process is repeated until the input is validated or we reach the recursion threshold. The recursion threshold is the maximum number of recursions performed to validate a neighbor. The process of decision cluster formation and input validation is presented in Algorithms 5, and 6.

Algorithm 6: Check Validation

Input: n_i $\rightarrow$ validated device
$VL_i \rightarrow$ Validating list for device i
$\bar{M}_i$ = Mean of $(s_i(t), s_{\{VL_i\}}(t))$
σ_i = Standard Deviation of $(s_i(t), s_{\{VL_i\}}(t))$
if $(s_i(t) < \bar{M}_i + \sigma_i)$ *AND* $(s_i(t) > \bar{M}_i - \sigma_i)$ **then**
  return true
else
  return false

5.3.1.2 Tight Medoid Clustering

In the tight medoid clustering (TMC) scheme, the requester devices asks for energy measurements from all its neighbors. When the neighbors respond with their measurements, they also include the measurements from all their neighbors along with a qualifying confidence factor corresponding to each neighbor measurement. If the device has the reputation information maintained as part of ReNVaS, this value becomes the confidence factor, otherwise

the confidence factor is 1. The availability of reputation information is not mandatory and TMC can function even without it. Upon collecting the information, the requester device creates a union of all the inputs while removing the duplicates and taking an average of the confidence factors of those duplicates. The presence of duplicate measurements is used for cross verification of intention of devices reporting them. This is further elaborated in Section 5.3.1.3. For the resultant set of inputs the requester runs a recursive partitioning around medoid (PAM) clustering on these inputs to partition them into sets. It starts by setting the number of clusters to 1 and recursively increases the number of cluster as long as it satisfies the following condition: The largest cluster has at least τ number of inputs. τ is the min-cluster-size parameter which depends on the estimation of the number of malicious devices in the network. τ is set to be inversely proportional to the number of malicious devices. If the number of malicious devices cannot be estimated, τ is set to $m/2$ to accommodate the worst case scenario of having up to 50% of the devices being malicious. We assume the maximum number of malicious devices to be 40% of all devices. This is depicted in Algorithm 7.

5.3.1.3 Data Fusion

In this phase, the data in the decision clusters formed previously are synthesized to reach a final decision. In order to do so, the device needs to decide which of the two decision clusters to use for data fusion. We present two methods for the decision making process. The data fusion of the values in the decision clusters follows the following hypothesis. If δ is the energy threshold for the presence of primary user signal, null hypothesis indicating primary user

Algorithm 7: Tight Medoid Clustering

Input: n_i $\rightarrow$ requester device; $l_i \rightarrow$ neighbor list of n_i;
$S_i = \{$(deviceid, measurement, ConfidenceFactor, NumMeasurements)$\}$
while $n_m \in l_i$ **do**
  while $n_p \in l_m$ **do**
    if $n_p \in S_i$ **then**
      Update S_i for deviceid n_p as (n_p, measurement + $s_p(t)$, ConfidenceFactor + c_p, NumMeasurements + 1)
    else
      Add an entry for n_p to S_i
while $n_p \in S_i$ **do**
  measurement = measurement/NumMeasurements
  ConfidenceFactor = ConfidenceFactor/NumMeasurements
a = 1
validMaxCluster = 1
while *validMaxCluster* **do**
  KM_i = k-mediods(S_i, a)
  if $\exists(maxClusterSize(KM_i)) > \tau$ **then**
    DC_i = LargestCluster(KM_i)
    a = a + 1
  else
    validMaxCluster = 0
return DC_i

signal is as follows:

$$\boxed{\begin{aligned} H_0 &: \frac{\sum(S_{\{DC_x\}}(t))}{|DC_x|} > \delta \\ H_1 &: \frac{\sum(S_{\{DC_x\}}(t))}{|DC_x|} \leq \delta \end{aligned}} \tag{5.15}$$

In CR-MANETs, due to the mobility of the devices, it is possible that the devices do not have sufficient reputation information about other neighborhood devices to make a proper validation of the input so as to be included in the decision cluster. This can happen under the following conditions: a highly dynamic network in which the devices joining the network and attrition of the devices from the network happens at a very high rate. Second, every device after joining the network requires a stabilization period during which it is acquiring network parameters in the form of reputation of other devices. Third, a device can be at the edge

of the network having topological inhibitions from having sufficient neighbors to validate it. In such cases, a good device does not get a fair chance of getting validated using the recursive neighbor validation scheme. The tight medoids clustering scheme uses statistical measurements to reduce the size of the decision cluster depending on the estimation of the amount of malicious devices present. If an estimate cannot be made, the worst case scenario of having up to 50% of devices being malicious is assumed. Both the methods have their drawbacks which can be overcome by using a combination of the two methods.

Every device gathers the sensing information from its neighbors. This collection of information requires two communication sequences over one hop. In the first sequence every device collects the information about the neighbors and in the second sequence the devices exchange the sensing information of each of their neighbors. When a device collects inputs from its neighbors and forwards those inputs to its neighbor, a malicious device can easily modify the values its neighbors presented before forwarding it to the requester device. To overcome this problem, the requester device cross verifies the input from one neighbor against the inputs from other neighbors on the measurements from a common neighbor. For example, in Figure 2.1, device 0 is the requester device. Devices 2 and 3 share a common neighbor 10. During cross verification if device 2 and 3 report different values of energy measurement for device 10, it raises a red flag and the inputs from the devices 2 and 3 are used unless they can be verified using other neighbors. We studied how many devices share neighbors and the results are presented in Table 5.5. This was done for different number of secondary devices (50, 100, 150, 200). The results are an average over 50 topologies. It is seen that the number of uncovered devices (a device which does not have any common neighbor with another device in the requester device's neighborhood) keeps reducing drastically as the device density in the network increases. In the example in Figure 2.1, device

Table 5.5: Covered Neighbors

# devices	total neighbors	covered neighbors	% uncovered
50	100.8	89.2	11.5
100	418.9	411.9	1.6
150	954.6	951.9	0.2
200	1687.4	1686.3	0.06

1 does not have any common neighbor with any of the other neighboring devices of device 0. Hence, it will be classified as an uncovered neighbor. A network with 200 devices spread over 2000 by 2000 area has 99.94% of neighbors covered with at least one common neighbor which can be used for cross verification. If a device is uncovered, its input is not used, hence deterring a device from reporting incorrect neighborhood results. This verification process will be a deterrent against sybil attacks as well. This works because it is a non-collaborative attack model.

5.3.1.3.1 Non-unified Decision Making Upon receiving the inputs, the requester device (i) runs a check as presented in Algorithm 8 to decide on the scheme to use for the formation of the decision cluster. This gives us the final decision $g_i(t)$. For using ReNVaS, the minimum number of neighbors that should be available to validate is given by ∇. ∇ is set to half the average neighborhood size.

5.3.1.3.2 Unified Decision Making The second fusion method for primary user spectrum occupancy detection uses both strategies (ReNVas and TMC) to decide on the occupancy. The devices make the preliminary decisions for the two strategies and combines them to make the final decision as follows: $g_i(t) = (D_r \vee D_m)$, where D_r is the result of recursive validation and D_m is the result of tight medoid clustering in device i at time t. This allows for the stringent check of the primary user spectrum occupancy and if the spectrum is iden-

Algorithm 8: Non-unified Decision Making

Input: n_i $\rightarrow$ requester device
$l_i \rightarrow$ neighborhood list of n_i
$l_i[\,] \rightarrow$ Two hop neighborhood list of n_i
foreach $n_i \in n$ **do**
    MedoidClustering = 0
    foreach $n_x \in l_i$ **do**
        if *(*$|VC_i \bigcap l_x| < \nabla$*) AND (*$|VC_x \bigcap l_x[\,]| < 2 * \nabla$*)* **then**
            MedoidClustering = MedoidClustering + 1
    if *MedoidClustering* $> \nabla$ **then**
        MedoidClustering() `// Algorithm 7`
    else
        RecursiveValidation() `// Algorithm 5`

tified to be occupied by either of the two strategies, it results in the final decision to be of occupied. This can lead to some additional false positives but it reduces the probability of any false negatives as we see in the next section. This can be further optimized to reduce the false positives, but is currently out of the scope of this thesis and is good to investigate in future.

5.3.2 Analysis and Results

It is generally accepted that any security feature incorporation in a protocol will add some amount of overhead. In this section we first analyze the amount of additional overhead generated by the scheme. Next, we perform security analysis by simulating the CR-MANET in Matlab. Two types of networks are simulated. One type of network is a stable network and the other is a network with fast changing network memberships (dynamic network). The secondary user network is spread over an area of 2000 x 2000 meters and the number of secondary users are 200. The CR-MANET devices move around in the network following the random waypoint model. Each device picks a random location within 250 meters off its

current position and moves to this new location. It stays in that location for a set timeperiod before repeating the process. Every device moves in this manner independent of other devices. This simulation is run for 1000 timeperiods and each device senses the spectrum for the presence of primary user signal. We simulated four levels of signal measurement errors, namely, 5%, 10%, 15%, and 20%. For example, if the signal measurement error is e%, the device measurement will range between $(s(t) - s(t) * (e/2)\%)$ and $(s(t) + s(t) * (e/2)\%)$ where $s(t)$ is the actual signal. The number of malicious devices was varied as 10%, 20%, 30%, and 40% of all the devices in the network. The malicious device reported signal measurement with a distortion of upto 50% of the measured signal i.e. $s(t) \pm s(t) * (0 : 25\%)$. In another set of simulations, we added dynamism to the network membership by adding 10% new devices every 100 timeperiods and 10% of existing devices left the network at that time. 2% of random devices undergo byzantine failures.

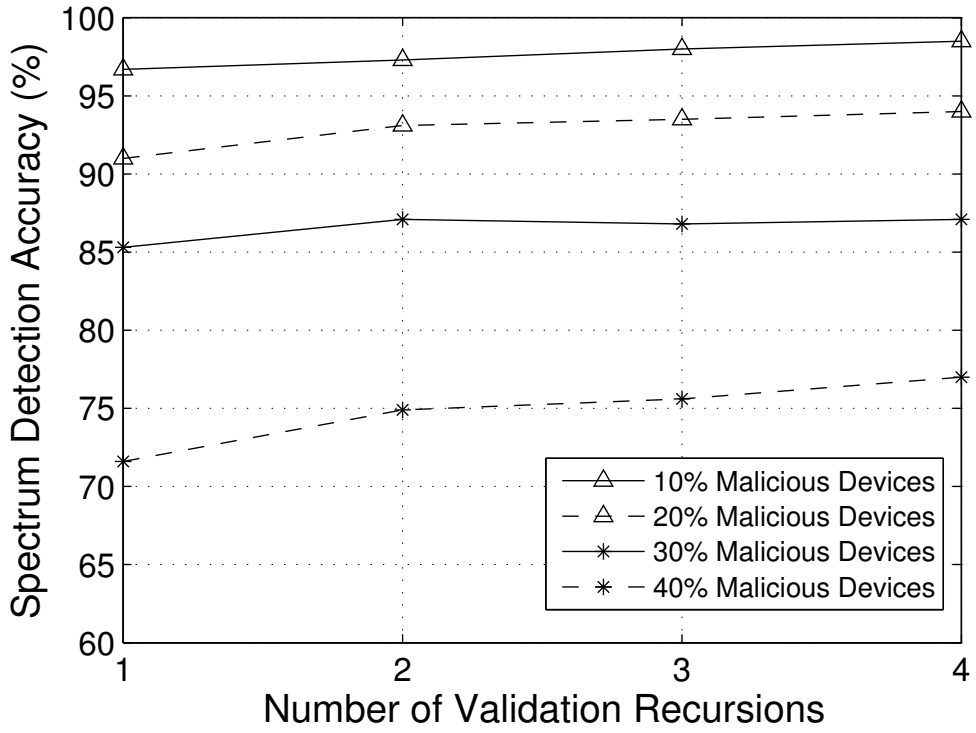


Figure 5.13: 5% Measurement Error

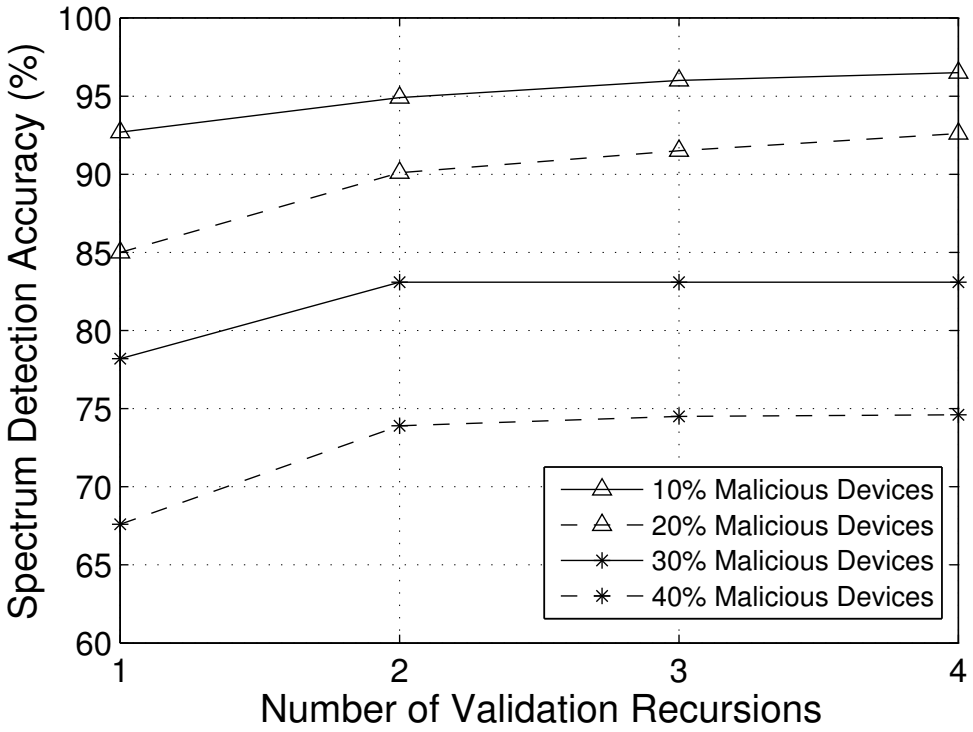


Figure 5.14: 10% Measurement Error

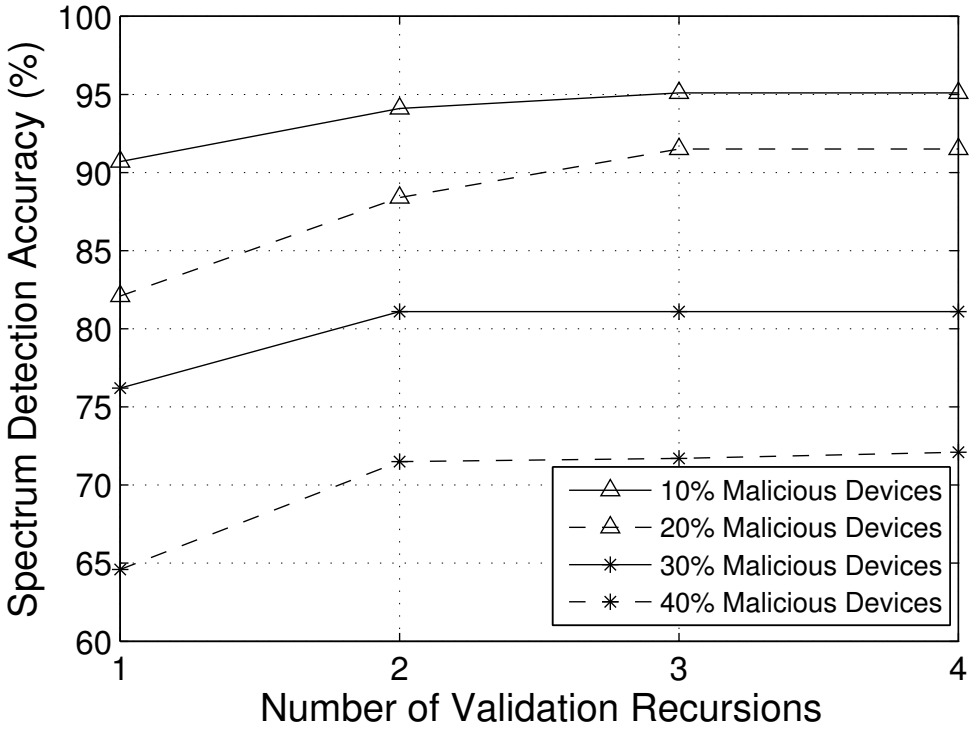


Figure 5.15: 15% Measurement Error

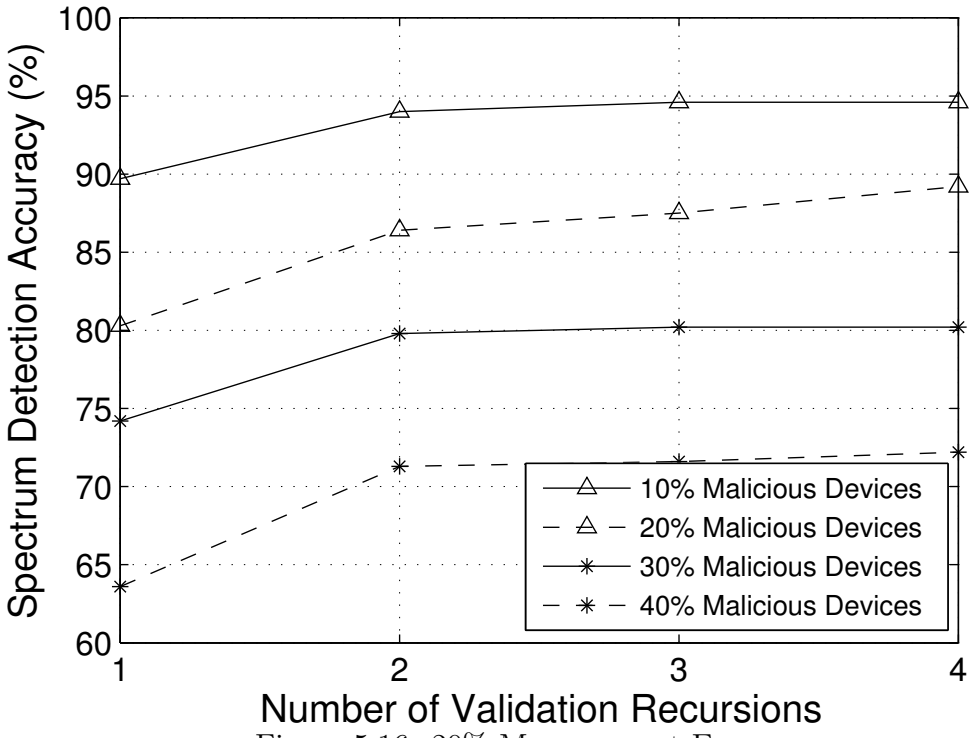


Figure 5.16: 20% Measurement Error

5.3.2.1 Overhead Analysis

In the traditional CR-MANET, every device senses the spectrum and takes a decision on primary user occupancy. This makes the secondary user network extremely vulnerable to different problems like signal fading, hidden terminal or byzantine failures and the device will ultimately make incorrect decisions. Hence, for comparison purposes we consider the traditional CR-MANET to be performing a cooperative spectrum sensing with the one-hop neighbors and utilizing the results by performing a simple average fusion.

In ReNVaS, the formation and storage of the virtual cluster information has the following overhead. For every device we store 3 parameters of information (device-id, reputation, and timestamp). The reputation is 1 byte long, while the device-id and timestamp are 2 bytes long. In the scenario where a device has a virtual cluster size of $n-1$ (has information about every other device in the network), the total storage space taken is $5 * (n-1)$. Unlike some

networks such as sensor networks, a CR-MANET is generally a mid-size network with a few thousand devices giving us a storage overhead of $5 * j$ KB where $1 \leq j \leq 10$.

Our model requires the devices to exchange the information twice over one-hop, hence the amount of transmission overhead compared to a traditional CR-MANET is double. However, when you compare the transmission overhead to a centralized cognitive radio network, it is much less and more fair. This is because in this model every device expends the same amount of energy while in a centralized model, the devices on high traffic path end up spending more energy. After the collection of information, if the recursive validation approach is used, the communication overhead will grow depending on the number of recursions used to validate the neighbor input. A device can cap the number of recursions and we will see in the results that the most cost effective solution is when going up to two recursions to validate the devices.

Let c be the neighborhood size of a device. v is the validation recursion number and ρ_{v-1} is the cumulative number of devices validated till recursion $v-1$ where $\rho_0 = 0$. The number of hops of communication for validation recursion at each recursion is calculated as $2 * v * (c - \rho_{v-1}) * c^{v-1}$. Total communication overhead for w recursions is calculated as shown in Equation 5.16.

$$\boxed{\sum_{v=1}^{w} 2 * v * (c - \rho_{v-1}) * c^{v-1}} \tag{5.16}$$

We see that as the number of recursions increases, the overhead increases significantly. Hence, we need to balance the number of recursions of validation based on the overhead. In the next section we will see that having 2 recursions gives the most cost effective solution while having 3 recursions gives us slightly more accurate results but at a higher cost. We do

% Malicious	Malicious majority neighborhood (%)
10	0.1
15	0.7
20	2
25	4
30	9
40	20

Table 5.6: Malicious Majority Neighborhood

not analyze the computation overhead since it is negligible compared to the communication overhead.

5.3.2.1.1 Malicious Neighborhood Analysis The attack considered is a non-collaborative attack and we assume the total number of malicious devices is less than 40%. But since the devices are mobile, and due to uneven distribution, the number of malicious devices in a neighborhood will be different. Table 5.6 shows the percentage of devices whose neighborhood (including themselves) has a malicious majority when the total number of devices is 200. When having up to 25% malicious devices, the number of devices having a malicious majority neighborhood is 4% and of those close to 2% were malicious themselves. Hence, of the good 75% devices, only 2% were affected by malicious majority which can be also seen in the results.

5.3.2.2 Results

In this section we present the results from our network simulation as described earlier. Figure 5.13 presents the results for ReNVaS when the sensing measurement error is 5% for different number of malicious devices present in the network. Similarly, Figures 5.14, 5.15, 5.16 present the results for signal measurement errors of 10%, 15%, and 20% respectively. This set of results presented is for a stable network without changes in the network membership

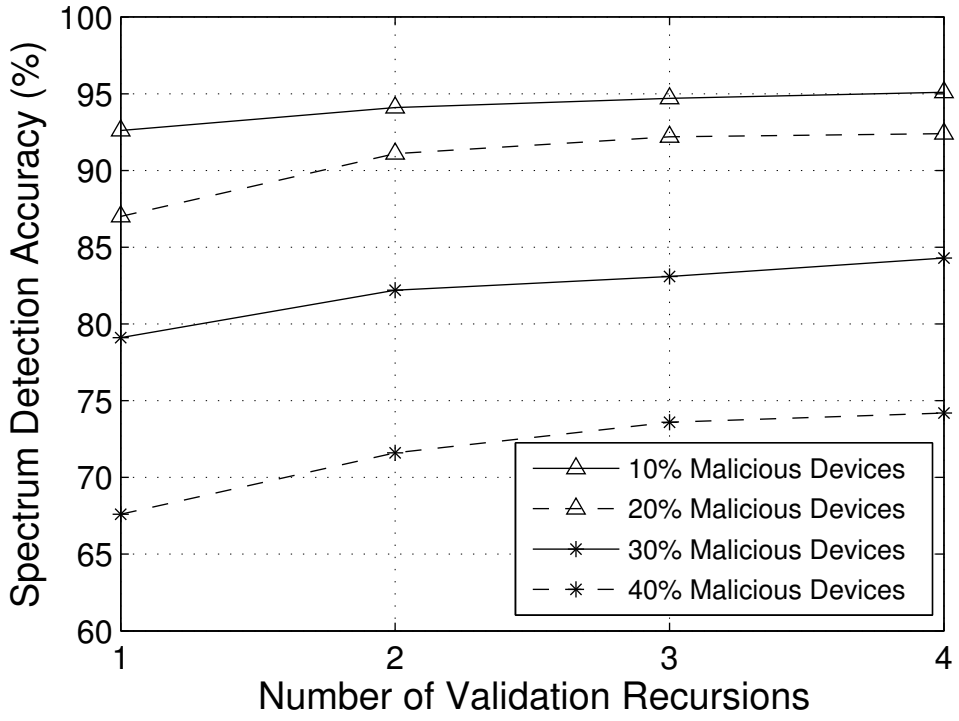


Figure 5.17: Recursive Validation- Dynamic Network 5% Measurement Error

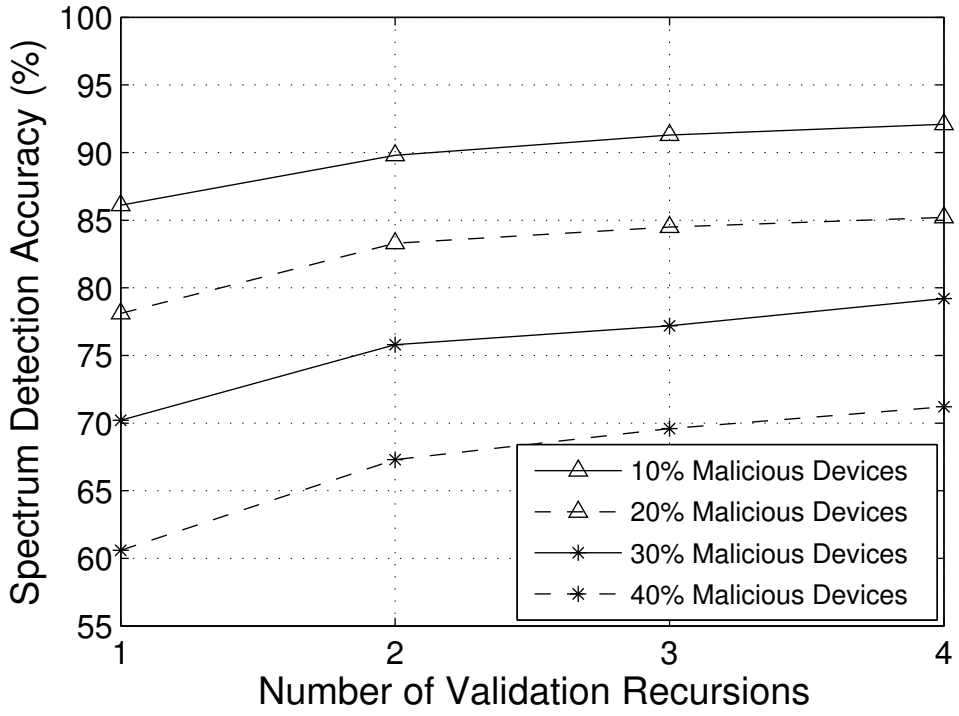


Figure 5.18: Recursive Validation- Dynamic Network 20% Measurement Error

for the duration of simulation. We see that the signal error measurement does not have as much impact compared to the amount of malicious device.

A separate set of network simulations were performed for the more dynamic network described earlier. The new devices joining the network do not have any reputation information of existing devices in the network and this impacts the overall performance. The degradation in overall performance happens because enough devices cannot be validated. This can be seen in Figures 5.17 and 5.18, where we present results for 5% and 20% measurement error respectively. Comparing the results in Figure 5.17 with the results in Figure 5.13, we see a significant drop in the primary user detection accuracy with changes to network membership. This happens because, with additional dynamism the overall performance of recursive validation falls due to its dependence on reputation information which is unavailable at some new devices.

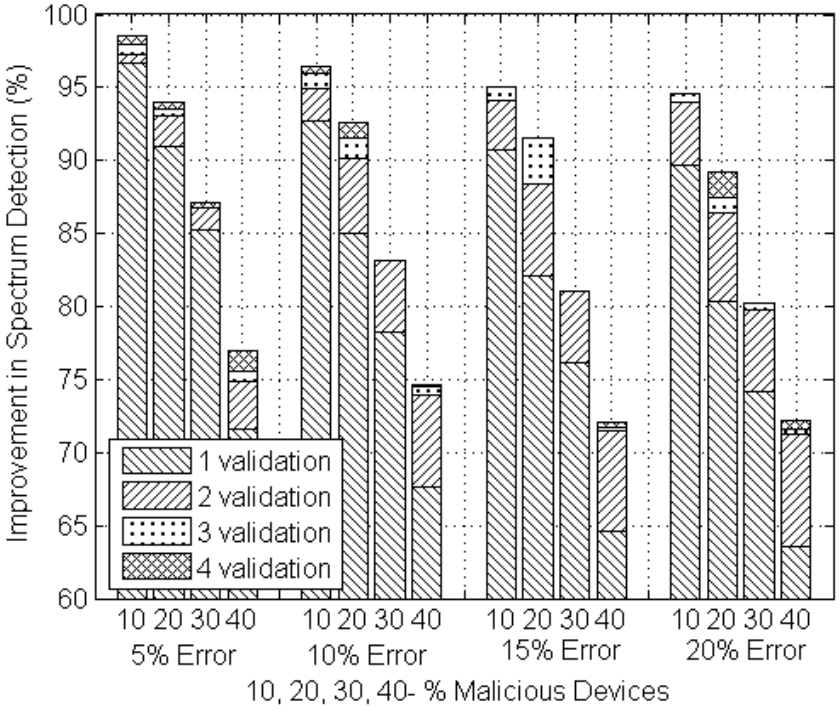


Figure 5.19: Spectrum Sensing Accuracy Improvement for Recursive Validation

Figure 5.19 shows the improvement in performance over different recursions for different

error rates in the measurement. Each bar represents the improvement over different number of validations for number of malicious devices (10%, 20%, 30%, and 40%) for the four measurement error rates of (5%, 10%, 15%, and 20%). It can be seen that the most improvement happens when going from one round of validation to two rounds of validation. This seems to hold true especially when the number of malicious devices is high (20%, 30%, and 40%). Beyond that, for increased number of validations, we only see a slight improvement. Also, as deduced in the overhead analysis, this additional improvement from having 3 or 4 rounds of validations has an associated cost which cannot be always justified.

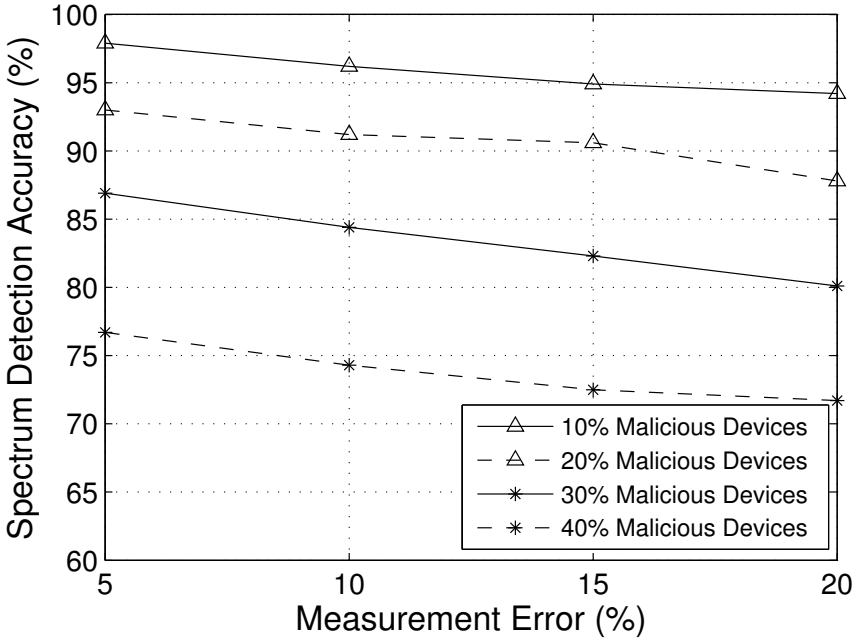


Figure 5.20: Spectrum Sensing Accuracy for Non-unified Fusion

The next set of results in Figures 5.20, and 5.21 show the performance of non-unified decision making scheme for the dynamic network described earlier. Based on the deductions from Figure 5.19, the recursion threshold for device validation was set to 2. Figure 5.20 presents the spectrum detection accuracy which gives us better than 70% detection even when the number of malicious devices is 40% and the measurement error rate of 20%. Figure 5.21 presents the incorrect decisions along with the amount of false positives and false negatives.

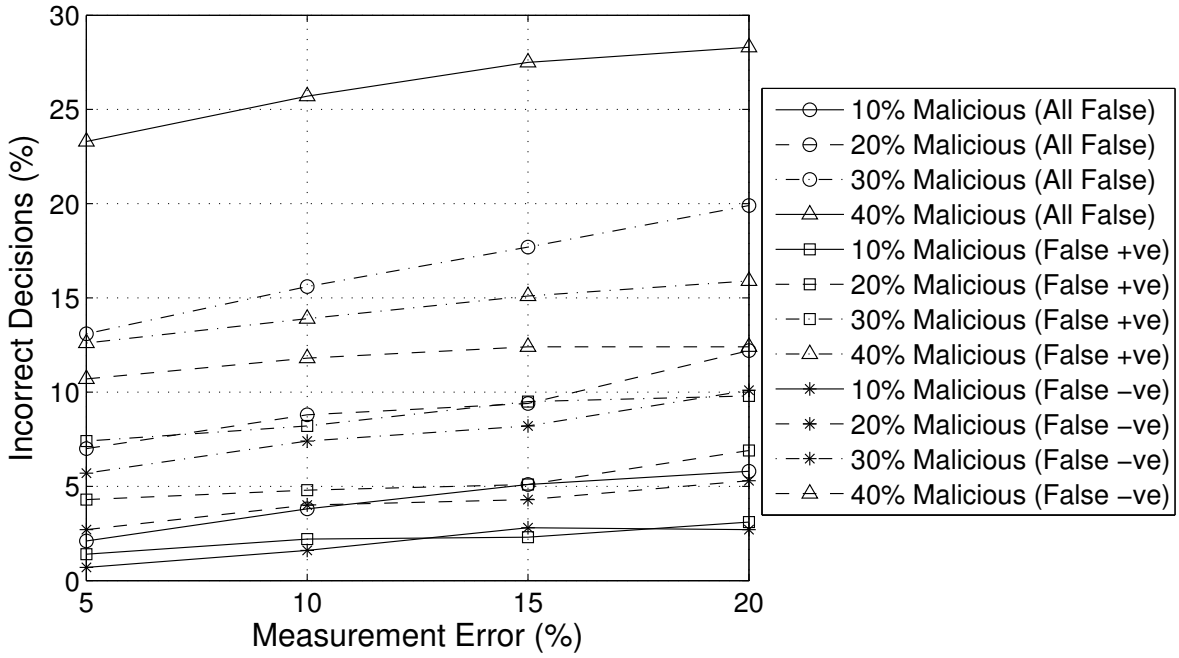


Figure 5.21: Incorrect Decisions for Non-unified Fusion

A false positive is when the spectrum is found to be occupied while the primary user is not using it. A false negative is when the spectrum is decided by the secondary user to be unoccupied while the primary user is using the spectrum. In the non-unified data fusion method the number of false positives and the number of false negatives occur at the same rate. A false positive result only reduces the secondary user throughput whereas a false negative decision will cause interference with the primary user.

The results in Figures 5.22 and 5.23 are for the unified decision making scheme and is compared to the performance of WSPRT by Chen *et al.* [66]. The spectrum detection accuracy of the unified scheme is similar to the one of non-unified decision making, but the false negative rate is far less than the false positive rate as shown in Figure 5.23. This happens because when either of the two decision cluster schemes (ReNVaS or TMC) conclude that the spectrum is occupied, the unified decision is that of spectrum occupied. It results in some additional false positives but the amount of false negatives reduces considerably. We also see

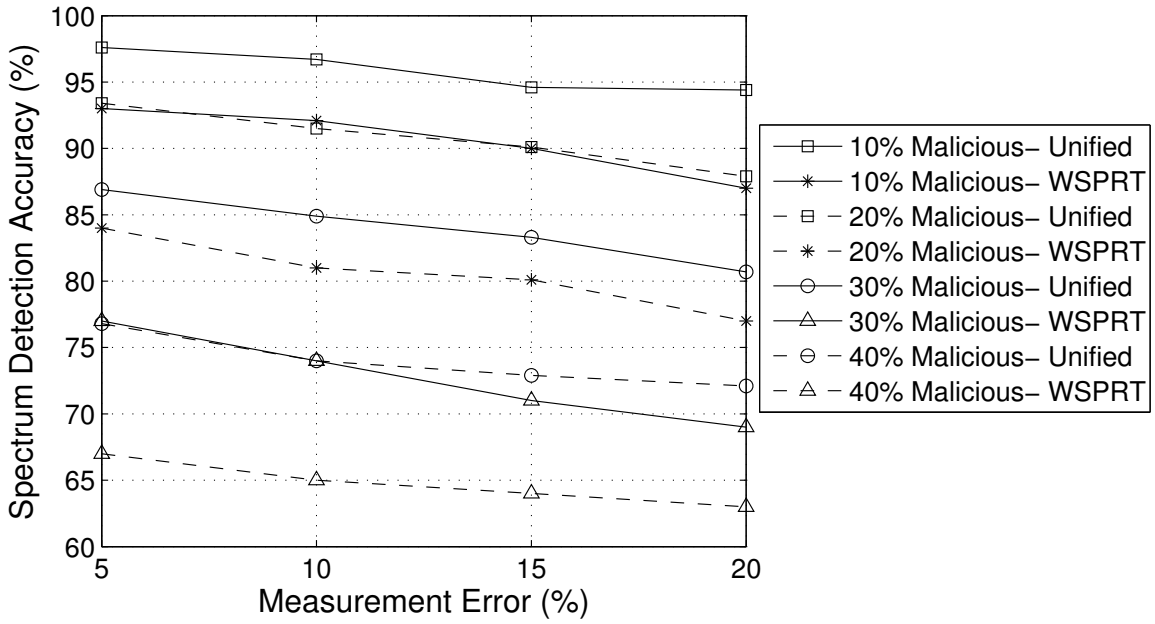


Figure 5.22: Spectrum Sensing Accuracy for Unified Fusion Vs WSPRT

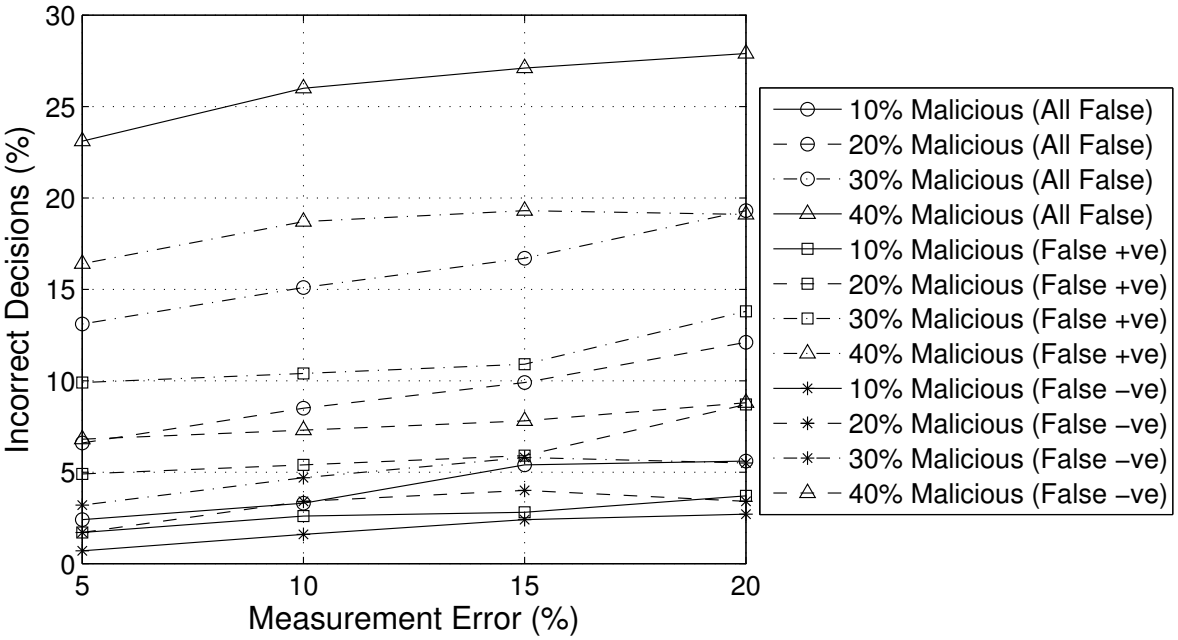


Figure 5.23: Incorrect Decisions for Unified Fusion

that the performance of unified scheme is on an average 10% better than WSPRT. One of the reasons is the mobility of devices causing imperfect input parameters for WSPRT. This happens because WSPRT calculates the apriori probabilities for miss detection and false positives based on the device location.

Unified and non-unified fusion using ReNVaS and TMC work well in a network with slow mobility and the neighborhoods are set for brief periods of time. This is because, recursive neighbor validation approach requires to communicate over multiple hops depending on the recursive cycle. If the network has fast changing neighborhoods, the performance of ReNVaS and TMC is affected, since enough information cannot be gathered in a short period of time for ReNVaS to function optimally. Under such circumstances, the unified and non-unified fusion (using ReNVaS and TMC) approach falls back to just using tight medoid clustering, which is not as robust as MFDSS using reputation. In a fast mobility network, MFDSS performs better due to its quicker decision making. This is because, in MFDSS we require one hop communication broadcast (twice) to exchange data so as to collect enough information to make a decision. Additionally, this also impacts the overhead of the two mechanisms. MFDSS has modest constant communication overhead whereas ReNVaS has increasing communication overhead which is governed by the number of recursions required to validate the neighbor. Hence, if communication overhead is a concern, MFDSS is better suited than ReNVaS and TMC. If accuracy of spectrum sensing has higher priority compared to overhead and the network mobility is not fast, unified fusion utilizing ReNVaS and TMC is better suited. In a fast paced mobile network, MFDSS performs better and the performance degrades gracefully as the number of malicious users grow beyond 40%.

5.4 Related Work

Compared to a wireless network, a cognitive radio network introduces new security challenges. In addition to the security concerns with the wireless network, a cognitive radio network needs to also consider the attacks of the following types: primary user emulation attack, spectrum sensing data falsification attack (SSDF), and lion attacks to name a few. The research community is gearing up to addressing these challenges with a lot of studies governed towards these specific attack types. Jayaweera et. al. have done significant amount of work in extending the research in cognitive radio networks [68–71].

The SSDF attack was coined by Chen *et al.* in [58] where they defined it as the security threat to distributed spectrum sensing by malicious secondaries transmitting false spectrum sensing data. There can be three cases when the secondary transmits wrong data. One, when there is byzantine failure and the secondary is unable to gather correct information. Second, when the secondary is maliciously spreading false data to harm the network. The third is when a malicious secondary transmits false data with selfish motives to use the spectrum for its personal gain by transmitting that a primary user is occupying the spectrum.

Wang *et al.* present a reputation based system to calculate the suspicion level of a device and assigns trust and consistency [72]. They remove untrustworthy users in the primary user decision making process, thereby making it robust. One of the drawbacks in this system is the assumption of presence of only one malicious secondary user. In an SSDF attack, there generally is two or more devices which may or may not be collaborating to make maximum use of the attack. Another drawback is the requirement of a centralized base station which collects all the information to make a decision. Although, such a system conforms to the IEEE 802.22 requirement, it may not work with an ad-hoc network requiring distributed

decision making.

Zeng *et al.* [73] present a trusted node assisted cooperative spectrum sensing to identify malicious nodes and remove the sensing reports based on reputation while making the decision on the presence of a primary user. The requirement of the trusted nodes and a centralized system make it infeasible to use such a system in an ad-hoc network. An outlier detection technique is used by Kaligineedi *et al.* [74]. In this paper, the authors pre-filter the data by removing the outliers based on a simple quartile based outliers detection mechanism. Additionally, similar to the reputation based schemes, a trust value is calculated and the higher trusted nodes are rewarded in the form of bigger weights in the primary user presence decision.

Another class of studies use some kind of outlier detection to discard inputs which are very different. A variation of this class of solutions is to use some kind of validation of the inputs before utilizing them in decision making process. Kaligineedi *et al.* [74] pre-filter the inputs using a quartile based outlier detection mechanism and use a trust based system similar to the ones described earlier to provide weights to the inputs in the final decision making process. Min *et al.* [75] present a centralized approach but uses a location based shadow fading correlation to validate the inputs before using them in the decision making process. This is a centralized approach and the secondary devices are assumed to be stationary, hence maintaining the neighborhood. Due to a fixed location and fixed neighborhood they get an opportunity to use consistent measurement parameters while validating the inputs which is not feasible considering changing location and continuously changing neighborhood in case of CR-MANETs.

Hyder *et al.* [76] present an adaptive reputation-based clustering scheme which uses the partitioning around medoid clustering. Equal size clusters are formed using the previous

sensing history and current reputation of the devices. Since it is a centralized system and the devices are stationary, a reputation based system is feasible. Applying the same to a distributed system with less number of inputs available and without consistent reputation information is infeasible. In ReNVaS we propose the use of device reputation to only decide whether the device input is used to validate a neighbor input. The manipulation of this reputation by a malicious device does not impact the outcome directly like it does in the existing approaches where reputation is factored into providing weights in the data fusion. Secondly, with a highly dynamic network a reputation system will fail due to the lack of availability of sufficient reputation information. Hence, a singular reputation or a trust based system will not be always feasible in a CR-MANET. Our approach provides a fall back under those circumstances and we device a recursive partitioning around medoid based clustering to weed out any outliers. The result is a tight bound cluster whose size is inversely proportional to the estimation of the number of malicious devices.

The study by Chen *et al.* is the first to present a distributed spectrum sensing scheme in CR-MANET without a centralized base station for data fusion and decision making [66]. They make an assumption that signal fading is not significant and takes the local decision as sufficient. Additionally, the reputation mechanism is local and does not incorporate any global information into it. Another drawback is the time or the number of samples it takes for the weighted sequential probability ratio test to reach a decision. MFDSS does not make the assumption of signal fading and takes into consideration the semi-global decision while awarding reputation at the local neighborhood. We collect the actual sensed values instead of the decision at the first phase which is important to weed out any outlier data generated due to byzantine failure or with malicious intent. Also, we incorporate a reputation propagation mechanism which drastically reduces the number of samples a node will need to collect to

reach a decision as we are able to award higher weighting factor to inputs from good devices. The ability of forming a quicker decision helps implement MFDSS in a highly mobile ad-hoc network. Additionally, MFDSS is more robust as it incorporates inputs from neighbors over 2 hops, so it is comparatively more secure from a small concentration of malicious devices which can negatively impact the decision when considering smaller input sizes. The same is true for ReNVaS and TMC as we will see in results and analysis in Section 5.3.2.

5.5 Summary

In this chapter we study the problem of improving spectrum utilization by a cognitive radio mobile ad-hoc network in the presence of malicious users. The goal is to identify the primary user spectrum occupancy in the presence of malicious users launching spectrum sensing data falsification attacks (SSDF), incorrect sensing data collected due to signal fading or hidden terminals, and byzantine failures. We present three solutions, namely multi-fusion based distributed spectrum sensing (MFDSS), recursive neighborhood validation scheme (ReNVaS), and tight medoid clustering (TMC) to overcome the problem.

The features of the multi-fusion based distributed spectrum sensing are as follows: First, we present a sensing data exchange scheme with outlier detection followed by its fusion utilizing weighted reputations. This step helps in identifying any outlier data input due to byzantine failure. Second, to prevent a malicious user from hiding under changing neighborhood, we present a reputation propagation and fusion scheme to gather valid and maintain fresh reputation of devices. In addition, an incubation period is introduced to discourage malicious devices from changing identity to wear off negative reputation. Third, a decision fusion scheme is presented which utilizes the reputation of devices to suppress the input

provided by malicious users. A unique feature of MFDSS is the implicit consideration of input from devices from a much larger area, thereby discounting the impact due to concentration of malicious devices withing a small region, even though the exchange of information occurs over one hop only. Fourth, we present a security and overhead analysis to corroborate the valid functionality of the model under different attack scenarios and present the MFDSS primary energy threshold analysis. MFDSS shows greatly improved performance compared to existing solutions, while preventing the malicious device from hiding under changing neighborhood. When the malicious devices are less than 30%, MFDSS has better than 83% spectrum detection accuracy. As the number of malicious devices grows larger, MFDSS is seen to degrade gracefully.

The core features of the recursive neighbor validation and clustering schemes is as follows: We present the formation and maintenance of a virtual cluster of past neighbors which is used for validation of inputs from the current neighbors. For a dynamic network with high device attrition, we present a recursive algorithm using partitioning around medoids clustering to find the tightest bound set of signal measurement inputs. These device inputs from the two schemes (ReNVaS and TMC) form a decision cluster. We present two strategies for data fusion of the decision cluster generated by ReNVaS and TMC, namely, non-unified fusion and unified fusion. The performance of ReNVaS is studied under two types of network conditions, one being a stable network while the other having continuously changing network memberships (dynamic network). It performs well under stable network conditions, while in a dynamic network the performance is affected due to the lack of availability of reputation information. Performance of ReNVaS and TMC are further studied under the non-unified and unified fusion in dynamic network conditions. Non-unified and unified fusion show improved performance with both having better than 70% detection accuracy even when the

number of malicious devices is high (40%) and at high error rate (20%). The difference being in non-unified fusion we have similar amounts of false negative and false positive rates, whereas in unified fusion we have significantly smaller number of false negatives compared to false positives, which is good as it agrees more with the FCC regulations.

To conclude, the results show that unified fusion using ReNVaS and TMC performs the best with great accuracy and has the smallest number of false negatives in a slow mobility network. In case of a fast mobility network, the performance of unified and non-unified fusion is not as good as MFDSS. This happens because ReNVaS requires multiple recursions to acquire sufficient inputs so as to validate the inputs from its neighbors. This may not be possible at high mobility and unified/non-unified fusion fall back on TMC for the decision cluster which is not as robust as MFDSS.

Chapter 6

Conclusion

In this thesis we researched security and privacy in wireless sensor networks and cognitive radio mobile ad-hoc networks. Sensor networks are unique because they can be inconspicuously deployed in the environment to sense for event occurrence. They help gather information in inhospitable environments and are one of the important developments in wireless networks. While a sensor network allows us to monitor event occurrences, there is another form of wireless network called cognitive radio mobile ad-hoc networks (CR-MANET), which senses the spectrum for occupancy by the primary user. When the spectrum is identified to be vacant, the CR-MANET device can opportunistically use the spectrum. This helps in improved spectrum utilization. While there are significant advantages with the advancement in the wireless technology, it brings its own security challenges. The sensor devices have a small form factor, which is advantageous for it to be inconspicuous, but introduces limitations such as limited battery, slow CPU, memory, etc.; while the CR-MANET devices are mobile which requires them to get their power from energy constrained batteries.

The sensor devices should be able to report the detected event information in a timely manner. This event information is bound to the location of the event detecting device. Hence, if the event detecting device is identified, it can lead to the event location being disclosed to an adversary, which can be harmful to the network goal. It is imperative to secure the privacy of the event detecting sensor device. In this thesis we present maintaining source privacy under eavesdropping and node compromise attacks (SPENA) which uses a

one-way hash chain based keying mechanism to hide the source information. Dynamically selected intermediate nodes on the path to the base station reconstruct the packet such that the adversary cannot correlate between the incoming and outgoing packets. The analysis and results indicate a superior performance in maintaining source privacy with modest a overhead.

The sensor network is deployed in the open and generally without supervision. The devices have limited physical shielding, which exposes them to side channel attacks. We performed a comprehensive study of different side channel attacks feasible on the sensor networks, while presenting countermeasures to prevent the same. A proof of concept is presented with experimental results to show the feasibility of electromagnetic radiation leakage attacks using readily available equipment. We also present a technique called process obfuscation which protects the sensor devices from a number of side channel attacks.

Another problem in sensor network occurs if the event generated packets do not reach the base station. The base station will not be able to differentiate between the loss of sporadic event packets and non-occurrence of events. The adversaries' goal here is to segregate the network region such that event packets from a particular region do not reach the base station and the adversary can carry out its malicious activities without being caught. We present a proactive solution called dynamic camouflage events based malicious node detection architecture (D-CENDA) utilizing camouflage events to detect such malfunctioning holes in the sensor network. In D-CENDA, the base station uses the spatial and temporal information of the camouflage event to detect the malicious node. In addition to identifying the malfunctioning nodes, we are also able to proactively identify the type of attack by the adversary causing the malfunction.

For a CR-MANET to function optimally, it should be able to accurately detect spectrum

occupancy by the primary user. Incorrect detection can lead to either spectrum under-utilization or interference with the primary user, which is strictly prohibited by the FCC. Accurate spectrum detection is hindered by signal fading, low sensitivity of cognitive radios, hidden terminal problems, byzantine failure of devices etc. To overcome these problems, a distributed cooperative spectrum sensing is considered, but it faces the problems of node mobility and malicious users. The malicious users launch an attack called spectrum sensing data falsification (SSDF), in which they encourage neighboring devices into making incorrect decisions about spectrum occupancy. Such incorrect decisions could prove fatal to a CR-MANET if it results in the secondary user transmission interfering with the primary user transmissions. To overcome this problem, we present three solutions namely multi-fusion based distributed spectrum sensing (MFDSS), recursive neighbor validation (ReNVaS), and tight medoid clustering (TMC). MFDSS includes three steps: sensing data fusion, reputation propagation and fusion, and decision fusion. The reputation management in MFDSS is robust and uses the principles from eigen-trust and maturity based reputation models. MFDSS has better than 83% spectrum detection accuracy even when there are 30% malicious devices.

ReNVaS maintains a virtual cluster of past neighbors which are used to validate the sensing inputs provided by the current neighbors. In TMC, the inputs from the neighbors are recursively clustered using partition around medoids algorithm to identify the tightest bound set. The validated inputs from ReNVaS and TMC forms a decision cluster and these are fused to form the decision. For this purpose two data fusion methods 'non-unified fusion' and 'unified fusion' are presented. Our solutions provide accurate spectrum occupancy detection, even under heavy SSDF attack. In the results, performance with up to 50% of the devices being malicious is analyzed and our solutions are seen to perform extremely well. The best performance is seen when using unified fusion utilizing results from ReNVaS and TMC,

where the number of false-negatives is minimized. The unified and non-unified decision making using ReNVaS and TMC function well in a slow mobility network, since it takes ReNVaS multiple recursions to gather the inputs required to validate the network. In case of a fast mobility network, unified and non-unified fusion (using ReNVaS and TMC) falls back on TMC to provide the decision cluster inputs, which is not as robust as using MFDSS which works better in a fast mobility network due to its quicker decision making time. Additionally, as the number of malicious devices increases beyond 40%, the performance of MFDSS degrades gracefully compared to the performance of unified and non-unified fusion.

In future, we would like to extend this to study the impact when the number of malicious users in greater than half the number of devices. We would like to study CR-MANET under very fast changing network topologies so as to apply it to vehicular networks. Finally, we would like to study the conglomeration of sensor network and CR-MANET to see if cognitive principles can be applied to create a hybrid cognitive radio sensor network.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] B. Warneke, M. Last, B. Liebowitz, and K. S. J. Pister, "Smart dust: Communicating with a cubic-millimeter computer," *Computer*, vol. 34, pp. 44–51, January 2001.

[2] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[3] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[4] J.-P. Kaps and B. Sunar, "Energy comparison of aes and sha-1 for ubiquitous computing," in *IFIP International Conference on Embedded and Ubiquitous Computing (EUC 2006), LNCS*, Springer, 2006.

[5] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk, "Enhancing source-location privacy in sensor network routing," in *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems*, (Washington DC), 2005.

[6] K. Mehta, D. Liu, and M. Wright, "Location privacy in sensor networks against a global eavesdropper," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP 2007)*, October 2007.

[7] Y. Ouyang, Z. Le, D. Liu, J. Ford, and F. Makedon, "Source location privacy against laptop-class attacks in sensor networks," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, (New York, USA), pp. 1–10, ACM, 2008.

[8] M. Shao, Y. Yang, S. Zhu, and G. Cao, "Towards statistically strong source anonymity for sensor networks," in *INFOCOM 2008, The 27th Conference on Computer Communications*, pp. 51–55, April 2008.

[9] Y. Yang, M. Shao, S. Zhu, B. Urgaonkar, and G. Cao, "Towards event source unobservability with minimum network traffic in sensor networks," in *WiSec '08: Proceedings of the first ACM conference on Wireless network security*, (New York, NY, USA), pp. 77–88, 2008.

[10] H. Wang, B. Sheng, and Q. Li, "Privacy-aware routing in sensor networks," *Computer Networks*, vol. 53, no. 9, pp. 1512–1529, 2009.

[11] Y.-C. Hu, M. Jakobsson, and A. Perrig, "Efficient constructions for one-way hash chains," in *Applied Cryptography and Network Security*, pp. 423–441, 2005.

[12] B. Xiao, B. Yu, and C. Gao, "Chemas: Identify suspect nodes in selective forwarding attacks," *Journal of Parallel and Distributed Computing*, vol. 67, no. 11, pp. 1218–1230, 2007.

[13] A. A. Nezhad, A. Miri, and D. Makrakis, "Location privacy and anonymity preserving routing for wireless sensor networks," *Computer Networks*, vol. 52, no. 18, pp. 3433–3452, 2008.

[14] C. Karlof, N. Sastry, and D. Wagner, "Tinysec: a link layer security architecture for wireless sensor networks," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*, (New York, USA), pp. 162–175, ACM, 2004.

[15] R. Watro, D. Kong, S.-f. Cuti, C. Gardiner, C. Lynn, and P. Kruus, "Tinypk: securing sensor networks with public key technology," in *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, (New York, USA), pp. 59–64, ACM, 2004.

[16] A.-S. K. Pathan and C. S. Hong, "Serp: secure energy-efficient routing protocol for densely deployed wireless sensor networks," *Annales des Télécommunications*, vol. 63, no. 9-10, pp. 529–541, 2008.

[17] T. Roosta, S. Shieh, and S. Sastry, "Taxonomy of security attacks in sensor networks and countermeasures," in *The First IEEE International Conference on System Integration and Reliability Improvements*, December 2006.

[18] J. Loughry and D. Umphress, "Information leakage from optical emanations," in *ACM Transactions on Information and System Security (TISSEC)*, pp. 262–289, 2002.

[19] D. Asonov and R. Agrawal, "Keyboard acoustic emanations," in *IEEE Symposium on Security and Privacy*, (Oakland, California), pp. 3–11, 2004.

[20] L. Zhuang, F. Zhou, and J. D. Tygar, "Keyboard acoustic emanations revisited," in *CCS '05: Proceedings of the 12th ACM conference on Computer and communications security*, (New York, NY, USA), pp. 373–382, ACM Press, 2005.

[21] Y. Berger, A. Wool, and A. Yeredor, "Dictionary attacks using keyboard acoustic emanations," in *13th ACM conference on Computer and Communications Security*, (New York, NY, USA), pp. 245–254, ACM Press, 2006.

[22] C. Gebotys, C. C. Tiu, and X. Chen, "A countermeasure for EM attack of a wireless PDA," in *International Conference on Information Technology: Coding and Computing (ITCC'05)*, (Washington, DC), pp. 544–549, IEEE Computer Society, April 2005.

[23] K. Tiri, D. Hwang, A. Hodjat, B. Lai, S. Yang, P. Schaumont, and I. Verbauwhede, "A side-channel leakage free coprocessor IC in 0.18μm CMOS for embedded AES-based cryptographic and biometric processing," in *DAC '05: Proceedings of the 42nd annual conference on Design automation*, (New York, NY, USA), pp. 222–227, ACM Press, 2005.

[24] S. Moore, R. Anderson, and M. Kuhn, "Improving smartcard security using self-timed circuit technology," in *Fourth ACiD-WG Workshop, Grenoble*, pp. 211– 218, 2002.

[25] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister, "System architecture directions for networked sensors," in *Architectural Support for Programming Languages and Operating Systems*, pp. 93–104, 2000.

[26] B. Atwood, B. Warneke, and K. Pister, "Preliminary circuits for smart dust," in *Southwest Symposium on Mixed-Signal Design*, (San Diego, California), pp. 87–92, February 2000.

[27] S. Chari, J. Rao, and P. Rohatgi, "Template attacks," in *Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, (London, UK), pp. 51–62, 2003.

[28] C. Collberg, C. Thomborson, and D. Low, "A taxonomy of obfuscating transformations," Tech. Rep. 148, University of Arizona, July 1997.

[29] D. Chaum, "Blind signatures for untraceable payments," in *CRYPTO 82: Advances in Cryptology*, pp. 199–203, Plenum Press, 1982.

[30] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, RSA, DSS, and other systems," in *CRYPTO '96: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology*, (London, UK), pp. 104–113, Springer-Verlag, 1996.

[31] P. C. Kocher, J. Jaffe, and B. Jun, "Differential power analysis.," in *CRYPTO*, pp. 388–397, 1999.

[32] J. Daemen and V. Rijmen, "Resistance against implementation attacks: A comparative study of the ales proposals," in *The Second AES Candidate Conference*, (Gaithersburg, MD), pp. 122–132, National Institute of Standards and Technology, 1999.

[33] Y. Ishai, M. Prabhakaran, A. Sahai, and D. Wagner, "Private circuits 2: Keeping secrets in tamperable circuits," in *Proceedings of Eurocrypt*, pp. 308–327, May 2006.

[34] D. Agrawal, B. Archambeault, J. R. Rao, and P. Rohatgi, "The em side-channel(s)," in *CHES '02: 4th International Workshop on Cryptographic Hardware and Embedded Systems*, (London, UK), pp. 29–45, Springer-Verlag, 2003.

[35] L. Goubin and J. Patarin, "DES and sifferential power analysis (the duplication method)," in *Cryptographic Hardware and Embedded Systems*, pp. 158–172, 1999.

[36] J.-S. Coron and L. Goubin, "On boolean and arithmetic masking against differential power analysis," in *Second International Workshop on Cryptographic Hardware and Embedded Systems*, (London, UK), pp. 231–237, Springer-Verlag, 2000.

[37] S. P. Skorobogatov and R. J. Anderson, "Optical fault induction attacks," in *4th International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 2–12, 2002.

[38] E. Biham and A. Shamir, "Differential fault analysis of secret key cryptosystems," *Lecture Notes in Computer Science*, vol. 1294, pp. 513–525, 1997.

[39] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," *Lecture Notes in Computer Science*, vol. 1233, pp. 37–51, 1997.

[40] A. Shamir and E. Tromer, "Acoustic cryptanalysis: on nosy people and noisy machines," in *Proceedings of EUROCRYPT*, 2004.

[41] R. E. Priestley, "The signal service in the european war of 1914-1918," *Institution of Royal Engineers*, 1921.

[42] K. Okeya and T. Iwata, "Side channel attacks on message authentication codes," vol. 3813, pp. 205–217, 2005.

[43] V. Gratzer and D. Naccache, "Blind attacks on engineering samples." Cryptology ePrint Archive, Report 2005/468, 2005.

[44] S. Micali and L. Reyzin, "Physically observable cryptography," in *Theory of Cryptography*, vol. 2951, pp. 278–296, Springer Berlin Heidelberg, 2004.

[45] B. Chevallier-Mames, M. Ciet, and M. Joye, "Low-cost solutions for preventing simple side-channel analysis: Side-channel atomicity," *IEEE Transactions on Computers*, vol. 53, pp. 760–768, June 2004.

[46] L. Batina, N. Mentens, and I. Verbauwhede, "Side-channel issues for designing secure hardware implementations," in *11th IEEE International On-Line Testing Symposium*, (Washington, DC, USA), pp. 118–121, IEEE Computer Society, 2005.

[47] P. C. Kocher, "Design and validation strategies for obtaining assurance in countermeasures to power analysis and related attacks," in *NIST Physical Security workshop*, sept 2005.

[48] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler, "Spins: security protocols for sensor networks," *Wireless Networks*, vol. 8, no. 5, pp. 521–534, 2002.

[49] A. Pirzada and C. Mcdonald, "Circumventing sinkholes and wormholes in ad-hoc wireless networks," in *International Workshop on Wireless Ad-hoc Networks (IWWAN 2005)*, (London, England), 2005.

[50] R. Roman, M. C. Fernandez-Gago, and J. Lopez, "Featuring trust and reputation management systems for constrained hardware devices," in *Proceedings of the 1st international conference on Autonomic computing and communication systems*, (ICST, Brussels, Belgium), 2007.

[51] H. Chen, H. Wu, X. Zhou, and C. Gao, "Reputation-based trust in wireless sensor networks," in *MUE '07: Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering*, (Washington, DC, USA), pp. 603–607, IEEE Computer Society, 2007.

[52] S. Ganeriwal and M. B. Srivastava, "Reputation-based framework for high integrity sensor networks," in *2nd ACM Workshop on Security of Ad Hoc and Sensor Networks*, (NY, USA), 2004.

[53] I. Krontiris, T. Dimitriou, T. Giannetsos, and M. Mpasoukos, "Intrusion detection of sinkhole attacks in wireless sensor networks," in *Proceedings of the 3rd international conference on Algorithmic aspects of wireless sensor networks*, (Berlin, Heidelberg), pp. 150–161, Springer-Verlag, 2008.

[54] E. C. H. Ngai, J. Liu, and M. R. Lyu, "An efficient intruder detection algorithm against sinkhole attacks in wireless sensor networks," *Computer Communication*, vol. 30, no. 11-12, pp. 2353–2364, 2007.

[55] C.-C. Su, K.-M. Chang, Y.-H. Kuo, and M.-F. Horng, "The new intrusion prevention and detection approaches for clustering-based sensor networks [wireless sensor networks]," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 4, pp. 1927–1932 Vol. 4, March 2005.

[56] Y. Xiao, *Security in Sensor Networks*. Boca Raton, Florida: Auerbach Publications, 2006.

[57] I. F. Akyildiz, W. yeol Lee, and K. R. Chowdhury, "Crahns: Cognitive radio ad hoc networks," *Ad Hoc Networks*, vol. 7, pp. 810–836, 2009.

[58] R. Chen, J. Park, Y. T. Hou, and J. H. Reed, "Toward secure distributed spectrum sensing in cognitive radio networks," *IEEE Communications Magazine Special Issue on Cognitive Radio Communications*, Apr 2008.

[59] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks Journal (Elsevier)*, vol. 50, pp. 2127–2159, 2006.

[60] V. Barnett and T. Lewis, *Outliers in Statistical Data*. Wiley Publishers, 1994.

[61] R. B. D'Agostino and G. L. TIetjen, "Simulation probability points of b_2 for small samples," *Biometrika*, vol. 58, pp. 669–672, 1971.

[62] B. Iglewicz and D. Hoaglin, "How to detect and handle outliers," *The ASQC Basic References in Quality Control: Statistical Techniques*, vol. 16, 1993.

[63] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The eigentrust algorithm for reputation management in p2p networks," in *Proceedings of the 12th international conference on World Wide Web*, (New York, NY, USA), pp. 640–651, ACM, 2003.

[64] P. B. Velloso, R. P. Laufer, D. de O. Cunha, O. C. M. Duarte, and G. Pujolle, "Trust management in mobile ad hoc networks using a scalable maturity based model," *IEEE transactions on networks and Service Management*, vol. 7, no. 3, pp. 172–185, 2010.

[65] C. Ghosh, S. Roy, M. B. Rao, and D. P. Agrawal, "Spectrum occupancy validation and modeling using real-time measurements," in *Proceedings of the 2010 ACM workshop on Cognitive radio networks*, CoRoNet '10, (New York, NY, USA), pp. 25–30, ACM, 2010.

[66] R. Chen, J. Park, and K. Bian, "Robust distributed spectrum sensing in cognitive radio networks," in *27th Conference on Computer Communication, INFOCOM*, pp. 1876–1884, 2008.

[67] R. Chen, J. Park, and K. Bian, "Robustness against byzantine failures in distributed spectrum sensing," *Elsevier Computer Communications*, vol. 35, pp. 2115–2124, 2012.

[68] K. S. Jayaweera and M. Bkassiny, "Learning to thrive in a leasing market: an auctioning framework for distributed dynamic spectrum leasing (d-dsl)," in *IEEE Wireless Communications and Networking Conference (WCNC'2011)*, (Cancun, Mexico), 2011.

[69] M. Bkassiny, K. S. Jayaweera, Y. Li, and A. A. K., "Wideband spectrum sensing and non-parametric signal classification for autonomous self- learning cognitive radios," *IEEE Transactions Wireless Communications*, vol. 11, pp. 2596–2605, 2012.

[70] G. El-Howayek and K. S. Jayaweera, "Distributed dynamic spectrum leasing (d-dsl) for spectrum sharing over multiple primary channels," *IEEE Transactions Wireless Communications*, vol. 10, pp. 55–60, 2011.

[71] G. El-Howayek and K. S. Jayaweera, "Efficient spectrum sharing with autonomous primary users: Distributed dynamic spectrum leasing (d-dsl)," in *IEEE Globecom 2010 Workshop on Broadband Wireless Access (BWA 2010)*, (Miami, Florida), 2010.

[72] W. Wang, H. Li, Y. Sun, and Z. Han, "Attack-proof collaborative spectrum sensing in cognitive radio networks," in *Proc. 43rd Annual Conference on Information Sciences and Systems (CISS'09)*, pp. 130–134, Mar. 2009.

[73] K. Zeng, P. Pawetczak, and D. Cabric', "Reputation-based cooperative spectrum sensing with trusted node assistance," *IEEE Communications Letters*, Dec 2009.

[74] P. Kaligineedi, M. Khabbazian, and V. K. Bhargava, "Secure cooperative sensing techniques for cognitive radio systems," in *IEEE International Conference on Communication*, pp. 3406–3410, May 2008.

[75] A. W. Min, K. G. Shin, and X. Hu, "Secure cooperative sensing in ieee 802.22 wrans using shadow fading correlation," *IEEE Transactions on Mobile Computing*, vol. 10, pp. 1434–1447, Oct. 2011.

[76] C. S. Hyder, B. Grebur, and L. Xiao, "Defense against spectrum sensing data falsification attacks in cognitive radio networks," in *7th International ICST Conference on Security and Privacy in Communication Networks (SecureComm)*, (London, UK), 2011.