Detection of Side-Channel Communication in a Mobile Ad-Hoc Network Environment Using the Hamming Distance Metric

by

Brent Moore

A Thesis Submitted in Partial Fulfillment

of the Requirements for the Degree of

Master of Science

in

Faculty of Engineering and Applied Science

And the program of Computer Science

University of Ontario Institute of Technology

Supervisor(s): Dr. Ramiro Liscano and Dr. Miguel Vargas Martin

June 2015

Abstract

Side-Channel communication is a form of traffic in which malicious parties communicate secretly over a wireless network. This is often established through the modification of Ethernet frame header fields, such as the Frame Check Sequence (FCS). The FCS is responsible for determining whether or not a frame has been corrupted in transmission, and contains a value calculated through the use of a predetermined polynomial. A malicious party may send messages that appear as nothing more than naturally corrupted noise on a network to those who are not the intended recipient. A Hamming Distance (HD) difference between the FCS values of purposely corrupted and naturally corrupted frames is proposed as a metric for the detection of side channel communication. In theory, it should be possible to recognize purposely corrupted frames based on how high this HD value is, as it signifies how many bits are different between the expected and the received FCS values. It is hypothesized that a range of threshold values based on this metric exists, which may allow for the detection of Side-Channel communication across all scenarios. In order to achieve this threshold range, a calculation known as F-Score has been used. Several approaches to verifying the F-Score thresholds have been presented to verify this range, as well as the validity of F-Score itself such as: Receiver Operating Characteristic (ROC) curves, and Support Vector Machines.

I dedicate this thesis to those that never stopped supporting me:

My best friend and partner Ellen Coombs

for all of her love and assistance,

which I could not have done without.

As well as Dave, Gayle, Ben and Ollie for keeping me motivated.

Acknowledgements

This project was supported in part by Defence Research & Development Canada and NSERC.

It is with the utmost appreciation that I thank my supervisors Dr. Ramiro Liscano and Dr. Miguel Vargas Martin. Your support, trust, and guidance were invaluable. Even when I was unable to meet certain deadlines, I knew your faith in me was not entirely lost.

A special thank you goes out to my colleague Visal Chea, for all of your support and guidance in getting me caught up in the project, and for assisting me in preparation of my experiment environments.

I would also like to acknowledge and thank Mazda Salmanian, Ming Li and Peter Mason at Defence Research & Development Canada for lending their expertise, insight and assistance.

Table of Contents

List of Tables

List of Figures

Chapter 1- Introduction

Mobile Ad-Hoc Networks (MANETS) provide an easily configurable mobile platform where nodes can communicate without requiring the use of additional hardware to provide routing. While convenient, these networks are not without their share of drawbacks in terms of security, management, or packet loss. As with all wireless networks, there is some level of information loss or corruption due to signal fading, frame collisions, or environmental interference. As a result, frames can become corrupt and will be disregarded by wireless receivers as noise. Side-Channel communication takes advantage of this as a method for discretely transmitting messages between two or more nodes. This process can be difficult to detect since it is hard to tell these intentionally corrupted messages from naturally corrupted ones; unfortunately for other nodes in a network, both appear as noise. Through the use of frame manipulation, it is possible for a malicious party to modify a frame in order for it to appear corrupted. Upon receiving a corrupted frame, most nodes will simply discard the frame, disregarding its existence. This behaviour is expected for all nodes on a network, unless the recipient has been configured in order to correctly receive and decode these secret frames. This type of communication is described as “Side-Channel Communication”, and while the process is conceptually simple to achieve, finding adequate hardware that supports the functionality required is rather difficult. Detection of frames based on this form of communication provide the foundation for the research presented throughout this thesis.

A network frame has several fields that are used for a variety of functions in transit, such as the source and destination fields. The field responsible for determining whether or not a frame has been corrupted in transmission is known as the Frame Check Sequence (FCS), and is a four-octet length field containing a value that is calculated prior to transmission [1]. Upon arrival, the receiver node utilizes an agreed upon algorithm in an attempt to re-calculate the value of the frame's FCS field via a Cyclic Redundancy Check (CRC). If the calculated value matches the FCS field's value, then the frame has arrived safely; however, if the calculated FCS value does not match the one present in the frame, the frame is considered corrupted and immediately discarded by the node.

1.1 - Problem Statement

A common question that occurs is "*Why go through the effort of establishing a Side-Channel, when the transmission can be encrypted?"* The reason for this is that while encrypted messages may be difficult to decode, they are not an inconspicuous means of communication. This is where network steganography techniques such as Side-Channel communication come in, as they allow multiple users the opportunity to transmit secret messages in plain sight. One of the main reasons why Side-Channel communication is so difficult to recognize is that there is no discernible difference between malicious Side-Channel frames, and those that were naturally corrupted through the transmission process.

In previous works [2], it was suggested that an increase in Side-Channel frames would provide an easily recognizable increase in Frame Error Rate (FER), to the point where a network

with Side-Channel would have a noticeably larger volume of errors than those without. Since these Side-Channel messages appear as nothing more than corrupted frames to average nodes, logically a substantial increase in corrupted messages on the network would provide evidence that Side-Channel must be occurring; however, this is not necessarily true. Since there are a variety of factors that could influence a wireless network, the amount of FER is never truly consistent. The level of noise on a network could change completely erratically, and almost arbitrarily. Many errors that occur in a wireless network may be triggered by seemingly inconsequential events, such as an individual walking through the transmission zone, a microwave or other device operating on the same frequency, nearby construction, or even changes in weather. With no persistent baseline for comparison, this eliminated the possibility of utilizing FER as a detection metric.

When determining a metric for use in detecting anomalous behaviour, such as Side-Channel, one of the requirements is that it must provide consistent results. In dealing with a MANET environment, this becomes increasingly more challenging as there is no guarantee of the surroundings or reliability of the network. There is also very little control over factors such as interference, and as such, the metric must be unaffected by FER. If such a metric is found, the challenge of validation still remains.

1.2 – Contributions

Due to the fact that FER is such an unreliable variable, there is a need for a detection metric uninfluenced by its fluctuations. The Hamming Distance (HD) calculation based on a frame's CRC polynomial appears to fulfill this requirement. There is a notable difference between the CRC value of Side-Channel frames using a modified CRC calculation and those using a standard algorithm, which will be touched upon further in Section 2.4. The difference between these values, or delta-CRC, is calculated by performing an XOR between the expected CRC value and the received one in order to compute an HD [3]. This metric was tested against a static ad-hoc scenario [3], and validated through various methods which will be described more in-depth in the Related Works section. While the HD metric appears to be relatively unaffected by factors such as FER, there are still several aspects of the work left to explore. This work provides four major contributions to the topic:

- Testing of the HD threshold technique against multiple MANET experiment scenarios in order to evaluate its effectiveness

- Determining a threshold range using a mathematical calculation known as "F-Score" to detect Side-Channel communications in MANETs when a Side-Channel is created by modifying the CRC polynomial

- Testing the validity of the F-Score approach by graphing the test against various datasets using an ROC curve

- Provides additional validation of the HD as a metric by performing analysis using the Support Vector Machine learning algorithm

1.3 – Structure of Thesis

The structure of this thesis was designed to provide the reader with the knowledge necessary to understand exactly what a Side-Channel is, along with many of the mechanics necessary for detection. Once the appropriate background concepts have been established, information regarding scenario procedures will be presented. Having a thorough understanding of the concepts, the reader should then be able to examine the results. In Chapter 2, previous works in the field will be examined. While Side-Channel communication is a relatively underrepresented area of research, there are a number of quintessential papers that provide the foundation for the concept. The aim of this chapter is to show just how challenging of an issue it is to not only detect Side-Channel, but also establish an adequate testing environment. In Chapter 3, the reader will be presented with the fundamentals necessary to understand the concepts of Side-Channel, the HD metric, and Mobile Ad-Hoc Networks. These areas provide insight that should help to understand the experimental design choices shown in the following chapter. Chapter 4 explains why simulation wasn't possible for this unique problem; how the OLSR routing protocol works; a very basic explanation of Platooning; and the parameters and methodologies used to conduct and analyze the experiments. Chapter 5 is where the information captured in the experiments comes to life in a series of figures that showcase the results, and alternative methodologies. Finally, Chapter 6 concludes by recognizing how the

contributions of this work impact the field, and possible directions for further examination of the Side-Channel problem.

Chapter 2 – Background & Related Work

2.1 – Steganography

In order to properly assess the detection of Side-Channel communication, it is important to understand where the concept originated, and analyze previously suggested detection methods. *Steganography* is a form of covert communication that dates back several millennia [4]. During this archaic era, messages were sent between generals by hiding them on the reverse of wax writing tables, on the stomachs of rabbits, and by tattooing them on the scalps of slaves [4]. Steganography was also used throughout the World Wars, where spies were able to use more advanced techniques, accredited to the invention of photography and technologies such as Microdots or Microfilm [5]. Steganography is used to hide a covert message, but does not hide the fact that two parties are in communications [4]. In the internet age, steganography is most commonly linked with techniques involving graphical images or audio files as a carrier medium; however, this is not the only digital steganography currently in practice.

The research presented by Szczypiorski [6] may be accredited as one of the pioneering articles in the topic of Network Steganography, inspiring a large amount of interest in the field. While most implementations of steganography systems are typically dedicated to multimedia, the research presented in HICCUPS: Hidden Communication System for Corrupted Networks [6] offers a unique approach, aiming to develop a steganographic system from a network

perspective. Even though research in network steganography is not that uncommon, many of the techniques examined rely on optional packet header fields belonging to very specific network protocols [7] [8] [9]. HICCUPS [6] was developed with the idea that if messages are modified at the Data Link Layer of the OSI model, it is possible to take advantage of naturally occurring imperfections in network transmission, such as noise. It is believed that this system offers an advantage over many of the other implementations in that it does not require any specific protocol. The results of the study by Szczypiorski [6] concluded that while steganographic communication may be possible, there are very specific and challenging criteria that must be met. As Szczypiorski [6] suggests, one of the issues with developing this form of Side-Channel is that there is a distinct lack of network interface cards that allow for the modification of frame header fields, such as the FCS field.

In a standard wireless network, all devices receive a copy of each message sent but often disregard those messages when they are not the intended recipients. This functionality stems from the use of CSMA (Carrier Sense Multiple Access) and CSMA/CD (CSMA with Collision Detection) protocols on a network. These protocols operate at the Data Link Layer, and measure a network for an absence of traffic prior to transmission. The intended usage for these protocols is to ensure that data collisions do not occur, or occur less frequently on a shared medium such as a wireless frequency. Szczypiorski [6] identifies these protocols as one of the three properties required for a working implementation of HICCUPS to exist.

The three requirements that must be met by a network susceptible to HICCUPS [6] are: a shared medium network with some form of CSMA; a publicly known method of cipher initiation (such as initiation vectors); and finally, integrity mechanisms for encrypted frames (such as FCS). The CSMA requirement stems from the fact that this mechanism gives all nodes the ability to "hear" all traffic on the network, meaning that malicious parties can analyze the traffic to find exploitable features. Three possible mechanisms that may be exploited to allow for the creation of a Side-Channel were also outlined in his work [6]: a channel based on a corrupted FCS field; a channel based on MAC network addresses; and a channel based on a cipher's initialization vectors.

The Initialization Vector channel requires all devices involved to be included within a hidden group that establishes a secret key for ciphers embedded in a steganographic system. This method was designed to work in a unicast, multicast or even broadcast mode utilizing the Diffie-Hellman algorithm for key exchange among nodes. A major drawback to a system of this type is that key exchange is difficult to mask from observers, requiring operation on a standard channel [6]. The second channel proposed by Szczypiorski [6] was entitled the "Basic Channel", the establishment of which requires a cipher's initialization vectors and MAC network addresses. It is suggested that the primary purpose of this mode was to allow for a channel characterized by low bandwidth where the exchange of control messages among hidden group stations occurs [6]. The third and final suggested Side-Channel has been referred to as "Corrupted Frame Mode" [6]. The detection and prevention of the "Corrupted Frame Mode" channel form the reasoning for this thesis. Szczypiorski [6] proposed that information could be

exchanged through frames which feature intentionally created corrupt FCS fields. The benefit of a channel of this design is that it provides the ability to utilize nearly 100% of the bandwidth for a certain period, and relying on the functionality of CSMA, nodes that are not the intended recipients will simply discard these frames as noise. Szczypiorski [6] felt that this method was out of the scope of his research, as he was unable to acquire a network interface card allowing for the manual modification of CRC checksums. While initial research on his third proposed channel was left largely untested in his work, it has become the focus of many works involving the Defence Research & Development Canada (DRDC) and the University of Ontario Institute of Technology (UOIT) [3] [10] [11].

2.2 – Anomaly Detection

Anomaly Detection mechanisms are often an alternative to everyday anti-malware or firewall solutions. These systems can offer an advantage over conventional intrusion detection methods in that they are not reliant on the use of a signature database. Some of the detection methods examined in this research include network traffic analysis, behaviour analysis, and smartphone security. While not all of these systems are network oriented, many of their functions and properties were considered as potential monitoring techniques.

In recent years, anomaly detection has received extensive interest from the academic community. While there are many researchers looking to develop the next Intrusion Detection System (IDS) through the use of machine learning or behavioral analysis, very few have had success with acquiring mainstream usage. Many of these systems employ the use of machine

learning algorithms, and several [12] [13] [14] were successful in their implementations; even so, anomaly detection is still not as mainstream as conventional anti-malware solutions. Sommer et al. [13] attribute the scarcity of such systems to the fact that the intrusion detection domain has been established for so long that there is a high barrier to entry for new applications.

In the field of intrusion detection, there is a high cost of failure if a misclassification were to occur. A false positive is often considered an inconvenience, and may require I.T. personnel resources to be spent examining incident reports for an alert that was triggered in error. Even a small rate of false positives can render a network intrusion detection system unusable [13]. On the other side of this scale, false negatives exhibit catastrophic results in which information is compromised, systems are damaged, or a loss of service occurs. While the usage of Machine Learning algorithms and other automated systems may offer several benefits for network monitoring, many of the systems employing anomaly detection often feature a false positive rate that may be considered unacceptably high [15] [16].

Sommer et al. [13] outlined several issues affecting the adoption of machine learning anomaly detection methodologies. Such issues include: the distinct lack of classification, the diversity in the forms network traffic can take, and the difficulties with evaluation. Further research on anomaly detection from Bolzoni et al. [17] explains that when alerts are raised by anomaly-based IDSs, the system is able to detect the anomaly, but has too little information to determine a classification for the attack. This limitation suggests that many anomaly-based

alerts require manual processing by I.T. personnel in order to classify an alert. Not only would this increase the work required by security teams, but also the time required to appropriately respond to the situation. Panacea [17] is a system that uses machine learning techniques to automatically and systematically classify attacks identified by an anomaly-based intrusion detection system, using information gathered about their payload. The idea behind such a system as Panacea [17] is the fact that attacks will often share common patterns in their payloads, such as byte-sequences. By examining this pattern, it is likely that there is an attack occurring belonging to a specific class, and an alert can be triggered.

Network characteristics such as bandwidth, application support, and network policies governing the length of a connection are all features that can prohibit widespread adoption of anomaly detection systems when dealing with network intrusion detection. When taking data for training phases, one of the most difficult considerations when dealing with networks concerns usage patterns. For example, usage can be highly variable over certain time intervals, resulting in many false positives when using a detection method heavily reliant on network patterns as a metric; while fluctuations are less notable over a large sample size, network traffic can see significant increases and decreases on an hourly basis. It is worth noting that a flux in usage is not the only observable challenge. Protocol specifications may operate in such a way that their behavior and the amount of traffic they produce varies, depending on the level of heterogeneity across the network, or based on the status of the current communication session.

Another challenge that the implementation of an anomaly detection system faces is in regards to determining a proper evaluation of the mechanism. Traditional signature-based anti-malware systems or other IDS systems such as firewalls often have access to publically available testing data or communities that produce content for testing purposes. An example of these tools comes in the form of voluntary spam requests [18], which allow for I.T. personnel to test their mail filters. When attempting to evaluate anomaly detection mechanisms, there is a limited amount of available datasets for appraisal, meaning that the evaluation process can be quite difficult. Should anomaly detection see more widespread usage in the future, or if researchers begin to focus their efforts on the development aspect of anomaly detection, evaluation of these systems could become much easier.

Many network operators utilize a system known as deep packet inspection (DPI) [19] [20]. DPI involves examining application layer protocols and content, such as port destinations, in order to monitor and control activity on a network. These systems are most frequently used in organizations, or countries with heavy restrictions on the content available to the general public. While effective for general purpose network monitoring, they are susceptible to exploits such as Protocol Misidentification or Polymorphic Blending Attacks [19] [21].

Protocol Misidentification is the process of labeling a packet designed to operate with a certain application layer protocol as another. Using this system, it is possible to bypass detection systems which employ DPI. For example, in an environment where FTP traffic is prohibited, a malicious party could mask the port number and protocol information attached to

outbound packets, and disguise them as HTTP traffic destined for port 80. This allows for a bypass around many network IDS systems and filters. To demonstrate this capability, Dyer et al. [19] proposed a system known as FTE (format-transforming encryption). FTE was capable of transforming ciphertext into a format of their choosing in order to bypass DPI. In addition to this functionality, the system can also act as a proxy for communication outside of restrictive countries or networks, with little to no bandwidth overhead. A counter to this form of attack would be to place a limitation on the types of protocols allowed to be transmitted across a network, but even this solution faces limitations. Users could determine which protocols are and aren't allowed on a network and simply adapt their approach to compensate for the restrictions. This is yet another area where anomaly detection implementations could provide an observable benefit. Based on typical network traffic, if an anomaly based intrusion detection system were to suddenly see a network inundated with an abnormal amount of FTP traffic, it would be a tip off that some form of protocol misidentification were occurring. An example of a network anomaly detection mechanism capable of providing such functionality exists, and has been presented in the form of a tool known as Spectrogram [14].

Spectrogram [14] is a network oriented anomaly detection system which operates in a passive state. This system, proposed by Song et al. [14], works as a filter that examines multitudes of web requests in an attempt to find a small subset of attack traffic. Spectrogram operates at the packet layer so that it can easily be implemented in conjunction with a port-mirror. By incorporating this functionality, the port-mirror is able to forward a copy of all received packets directly to Spectrogram, allowing for a system that does not add an additional

possible bottleneck to the receiver. The system operates by gathering packets, and analyzing them based on their content distribution and structure. Once packets have been picked out, they are passed through a system utilizing the machine learning algorithm known as Markov Chains, which afterwards requires some minor human interaction [14]. Unlike many of the other systems that were examined in this subsection [12] [17] [13], this system is not completely autonomous. After analysis, a final likelihood score for whether or not the anomaly is an attack is presented, and I.T. personnel can decide on a solution.

While the primary focus of this work is to detect Side-Channel anomalies on a network, it is important to consider other implementations of anomaly detection systems. In doing so, the evaluation of these systems may provide a better understanding of the functionality required to develop a proper detection mechanism. Some of the other areas that were assessed include anomaly detection on mobile devices [22] and masquerade detection [23], both of which feature techniques that are oriented towards behavioural analysis.

Threats against smartphones do not necessarily have to come in the form of malware, as the small size and transportability of these devices increases the risk of theft from simple eavesdropping techniques [22]. The limited functionality of these devices leaves their protection to rudimentary versions of security and authentication methods. Such features include pattern-based lock sequences, or simple 4-digit pin passcodes. Issues arise from this simplicity, as the limited functionality and vulnerable nature of these devices creates a challenge in preventing malicious access to a user's device. A process outlined by Muslukhov et

al. [16] involves the use of a trusted process which operates in the background and monitors a user's usage patterns. Once enough information has been obtained, a behavioural model of the user can be developed. Behaviour analysis allows for a detection mechanism to determine whether a particular action has been made by the user or a malicious party/software based on common patterns that a user may exhibit when interacting with their system or device. Should anomalous activity occur, a defensive action could trigger, such as a prompt for password authentication or a device lockdown.

Another key area for anomaly detection is the prevention of masquerade attacks. Masquerade attacks are one of the most common types of malicious activity on both networks and computer systems [23]. A masquerade is a type of attack in which a malicious party has gained access to a system or network session and intends to impersonate a legitimate user for the purposes of accessing confidential information or to gain access to permissions that they have not been granted. These types of attacks can be incredibly difficult to detect since the malicious party is often using the credentials of a legitimate user, and is for all intents and purposes that user (as far as the system is concerned).

A common method for the detection of these system masquerade attacks is to enlist the help of machine learning algorithms capable of classifying normal behavior and identifying suspicious activity. According to research presented by Salem et al. [23], an excellent way to determine if a masquerade is occurring on a computer system is to examine the search behaviour of the user. An average user will have fairly accurate knowledge of the layout of their

file system, thus when searching for a file, they will be able to do so in a limited fashion. It is the use of the search mechanisms in a Windows environment which plays an essential role in anomaly detection in the study presented by Salem et al. [23]. An extreme example of how search behaviour analysis could be effective is that a user will understand that photos of his family vacation would not be stored in his System32 folder, while a masquerading party or software may not. The result of such a lack of knowledge is a broader, more extensive search across the file system in a manner that is uncharacteristic of the typical user.

In their paper, Salem et al. [23] modelled the usage behaviour of 18 individuals who were working with their own personal computers for a period of 4 days. The results were then compared against simulated data created from 40 additional users performing a mock masquerade on a system unfamiliar to them. The data gathered were run through an SVM based anomaly detection mechanism, and was able to provide a 100% detection rate with a false positive ratio of only 1.1% [23]. These results are within an acceptable range, but with the requirement of such a large sample size, it is difficult to determine if this would be feasible in a real-world scenario without extensive training performed by the user [15] [16].

An additional area of anomaly detection which should be considered is the Support Vector Machine algorithm. Arguably the most successful classification method in machine learning, Support Vector Machines (or SVMs) are algorithms used to analyze data in order to recognize patterns and linear classifiers [24]. SVMs are a form of supervised learning, meaning that the details of the program are dependent on choices of parameters, which can be tuned by

the program given a set of objects of known classification [25]. In a lecture from the University of Caltech [24], Dr. Abu-Mostafa details the principled components of the method, which include finding the optimal margin, arriving at a solution analytically, and transforming the data nonlinearly; i.e., expanding the machinery to applicability with nonlinear data. These steps follow the approach developed by Cortes & Vapnuk [26] for binary classification in 1995, and are the characteristics of a program able to perform such tasks.

The term linearly separable is used to describe data for which there exists a linear decision boundary that separates positive from negative examples [27]. Given such a two-class, separable training dataset, there are many possible separating lines and margins of error [24]. SVMs search for the best linear separator by looking for the decision surface that is maximally distanced from all data points [28]. As Dr. Abu-Mostafa explains [24], the process of generating data may result in noise, and the bigger the margin, the greater the chances that the new point will still be on the correct side of the line. In other words, maximizing the margin gives a classification safety margin, meaning that a slight error in measurement or documentation will not result in a miscalculation [28]. These points lying on the boundaries are called the support vectors, and our optimal separating hyperplane occurs in the middle of this margin [26].

Through examining related works, a few important details regarding anomaly detection have been discovered. Due to the lack of popularity in real-world environments, such mechanisms often face a multitude of challenges throughout the development process. A lack of available training data means that implementation would require a lengthy training process

on the part of the technician. This is especially evident when trying to develop a solution for a very specific or obscure problem, such as Side-Channel detection. Coupled with the inherently large amount of sample data and time required by some of the machine learning algorithms, it is arguable that anomaly detection mechanisms may not exist in a mature enough state to completely supersede conventional signature-based mechanisms. By utilizing the delta-CRC approach proposed by Chea [3], detection is transitioned into packet analysis (delta-CRC) rather than signal analysis (FER), meaning that detection may be possible without the use of machine learning algorithms.

Many advantages to a variety of anomaly detection methods have been outlined above; however, these mechanisms are not perfect. Like all systems, anomaly detection mechanisms have a set of policies, which must be met in order for data to be determined anomalous. Engla et al. [21] suggest that should a malicious user develop an understanding of how an anomaly detection system operates through brute-force attempts to match the criteria, they could in theory trick a system into believing a user or message is legitimate.

The functionalities of anomaly detection methods, and prior research in the field that have been presented herein, should be highly favourable when developing a Side-Channel detection system. In spite of all of the weaknesses that are presented above, it is important not to discount the potential benefits to using machine learning algorithms. For the scope of this thesis, several of the aforementioned algorithms will not be considered; however, SVM will be examined as a potential candidate in Section 5.5. The justification of such an algorithm's

selection is primarily due to the fact that the Side-Channel problem is a binary decision, and SVM provides a distinct binary classification using linear separation.

2.3 – Establishing a Side-Channel

Najafizadeh [11] was able to establish a simulated Side-Channel communication in what was otherwise a rudimentary simulator application known as Sinalgo [29]. Using collected data from his Side-Channel simulations, he examined the ratio of corrupt to non-corrupted traffic during periods where Side-Channel existed, comparing them to those that had no Side-Channel. He was able to show that a system based on the network's historical data would showcase a high degree of variance in the amount of Frame Error Rate (FER) when Side-Channel communication was occurring [11]. Using this, Najafizadeh [11] proposed an agent-based detection system that would trigger an alert depending on whether or not the variance of FER fell outside of an upper bound. One of the limitations to his approach was a lack of exhaustive scenarios.

Another approach to the detection of Side-Channel communication was presented through the use of the RTS/CTS network mechanic by Madtha et al. [2] The Request to Send / Clear to Send (RTS/CTS) mechanic is one employed on many wireless networks as an optional feature used to prevent information loss due to packet collisions. In a network using this feature, a sender node will transmit an RTS frame to check the availability of a channel prior to sending out a data packet. If the channel is available, the destination node will reply with a CTS frame, informing other nodes to refrain from transmitting any data for a period of time. As soon as the sender node receives the CTS message, it will begin transmitting data packets.

In the work of Madtha et al. [2], it was hypothesized that for every Side-Channel frame, there should be a corresponding RTS/CTS message pair. What this means is that while non-malicious parties may be unable to determine whether or not a message has been purposely corrupted, analysis of the traffic should present a substantially higher number of RTS/CTS messages indicative of extra communication occurring on the network. For every RTS frame, there should be a corresponding data frame; this means that the ratio of received application data and RTS frames should be 1:1 in a network with no data loss. In the presence of Side-Channel, this ratio will increase, and the amount of RTS messages will be significantly higher [2].

Several experiments were run using the QualNet [30] simulator, such as an increase in the number of nodes, a varied number of Side-Channel links, and a range of inter-nodal distances. In the results presented by Madtha et al. [2], a distinct increase in RTS messages was shown to be disproportionate to the amount of known data packets when a Side-Channel is present. Unfortunately, while this research provided promising results, several weaknesses were identified. In order for their method to work, a network is required to be running the RTS/CTS mechanic, which may not necessarily hold true for all networks. Additionally, this method is incredibly sensitive to Frame Error Rate, and in networks with a high degree of FER, there will be substantially more RTS messages as packets become naturally corrupted and require re-transmission, effectively skewing the results. The research presented Madtha et al. [2] demonstrated a need for an FER insensitive metric.

2.4 – Hamming Distance Delta-CRC

In a wireless network, corrupted packets are typically detected through the use of a Frame Check Sequence (FCS). Frame Check Sequences work by appending a fixed-length binary sequence to the FCS field in the frame. This sequence is calculated by the source node based on the data within the frame. Figure 2.1 showcases several of the expected fields on a typical frame, including the FCS field. Other key fields which are present in a frame are the Preamble, SFD, Source/Destination Addresses, EtherType, and Payload. The Preamble contains a 56-bit binary pattern, which allows network devices to synchronize their receiver clocks. An SFD is used to signify the end of the preamble, and the beginning of the frame. Source and Destination fields contain the MAC addresses of both nodes, and allow nodes on a network to determine if a frame is meant for them. The EtherType field is two octets long, and often provides information regarding the length of a Payload. The frame's Payload is where the actual data is stored, and can have a size between 42 and 1500 octets in length.

Upon receiving a frame, the destination node recalculates the FCS sequence and compares it with the one included with the frame. If these values do not match, the frame is considered corrupt, and the node may request retransmission or drop the frame. The most common type of Frame Check Sequence is Cyclic Redundancy Check (CRC). This CRC value is calculated by considering the result of the remainder when dividing the polynomial for the data payload by the CRC polynomial. Prior to this calculation, both of the polynomials are converted to their binary form.

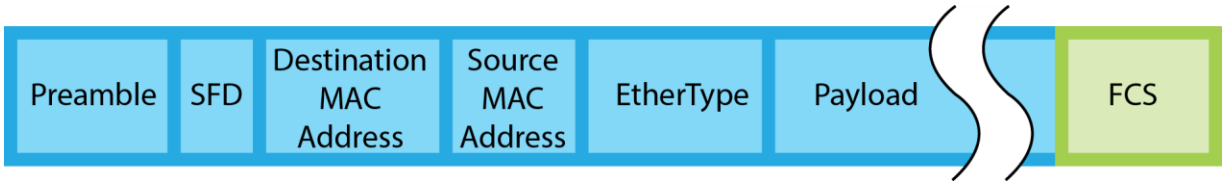

Figure 2.1 – A diagram of a standard frame w/ included FCS field.

After a CRC value has been calculated, it is appended to the FCS field and the frame may then be transmitted. There is not just one standard CRC polynomial, but rather a set of standardized polynomials as defined by the IETF [31], ITU [32], and other similar organizations [33]. In fact, there are over 72 possible standard CRC polynomials [34], which range from CRC-3 bits up to CRC-82 bits. It is also worth noting that while the CRC calculation does assist with the detection of transmission errors, it is entirely possible that certain bits may corrupt in such a way that a receiver may still calculate a CRC identical to the one that was sent [35]. In the work conducted by Koopman and Chakravarty [35], it is stated that a given CRC polynomial may operate more or less effectively on any given application. This means that a range of CRCs are necessary in order to facilitate a variety of operations, and the effectiveness of each is measurable.

Further work by Koopman [34] explores the effectiveness of many well-known CRC polynomials, and classifies each based on their effectiveness and ideal payload size. The work presented demonstrates that a 32-bit CRC polynomial, commonly known as the "Koopman Polynomial", provides the best error detection out of any of the standardised CRC calculations at the time. In his research, Chea [3] utilized this 32-bit CRC polynomial as a stand-in for the Side-Channel communication CRC in his experiments on HD as a metric for detection. Chea's [3]

justifications were in part due to the fact that the 32-bit CRC polynomial is so widely used, and has a high effectiveness in error detection. As such, this is the CRC calculation that has been used for the experiments described herein; however, testing against a less effective CRC polynomial, while outside of the scope of this research, could provide further verification on the effectiveness of HD as a metric for Side-Channel detection.

Throughout this work, the term "Hamming Distance" or "HD" refers to the number of bits that differ between an expected calculated CRC value and the actual value calculated by the recipient node. HD is a mathematical concept that was introduced by Richard Hamming in 1950 [36], and is commonly used today in coding theory when comparing the difference between bit strings of equal length [37]. The HD value refers to the number of characters in given positions for which corresponding items are different, or the number of characters that must be changed in order for two items to match. In summary, it is a numeric representation for how different two same-length strings are. Consider the following examples demonstrating the HD between two similar, but different strings:

"CA**T**" & "CA**R**" have an HD of 1
"Bri**gette**" & "Bri**ttany**" have an HD of 5

Figure 2.2 – A simple demonstration of the HD between simple ASCII strings.

This measurement can also be used to determine a difference in CRC values by examining them in their binary notation. When dealing with binary strings, the HD is equal to the number of ones in an XOR between two strings of length *n*. An example of this has been provided using

the two CRC polynomials that will be utilized in this research in Table 2.1. First, the Default and Koopman 32-bit CRC polynomials must be converted to their binary notation, and then an XOR may be performed to get their HD. In Table 2.1 below, it is shown that there are 13 positions with an XOR binary value of 1, meaning that the expected HD between the Koopman and Default CRCs is 13. This, of course, does not mean that there will always be an HD value of 13, as there is a chance that natural corruption may occur on Normal and Side-Channel frames. This corruption may cause certain bits to flip, creating CRCs with a wide variety of HDs, as will be shown in Section 5.3.

Default (0x04C11DB7)	1	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	1	0	0	0	1	1	1	0	1	1	0	1	1	0	1	1	1
Koopman (0x741B8CD7)	1	0	1	1	1	0	1	0	0	0	0	0	1	1	0	1	1	1	0	0	0	1	1	0	0	1	1	0	1	0	1	1	1
XOR	0	0	1	1	1	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	1	0	0	0	1	0	1	1	0	0	0	0	0

Table 2.1 – An XOR to calculate the HD of the Default 32-bit CRC polynomial, and the Koopman 32-bit CRC polynomial

In order for HD to work as a metric, one must first make the assumption that the number of bits different in a naturally corrupted Normal frame from a Side-Channel frame will be large enough to provide a visible separation between the two. The concept for using HD as a metric for Side-Channel detection was originally proposed by Chea [3]. This metric was the inspiration for much of the experiment planning for this work, and provided a basis to draw from and expand upon. The methodology and results presented by Chea [3] provide a better understanding for the rationale of many of the procedures used below.

Chea [3] suggests that the average HD between two CRC values could be used to detect Side-Channel where the malicious party has used an alternative CRC polynomial to mask his/her

traffic. In his work, Chea [3] also compared the HD of several CRC polynomials, including the Default 32-bit CRC, Koopman 32-bit CRC and Castagnoli 32-bit CRC to find the mean HD between them. His preliminary research showed that there was a clearly defined difference between the mean HD of the expected Default CRC vs. Koopman, and Default vs. Castagnoli [3]. Chea's [3] work also provides merit to the fact that it should be possible to adequately determine if a Side-Channel exists given the assumptions:

1. The CRC polynomial used for Side-Channel communication will be a known polynomial, such as the Koopman 32-bit CRC

2. The CRC chosen by the malicious user for Side-Channel CRC will be different enough from the Default CRC (i.e. not a CRC containing a single flipped bit difference from the Default)

The experiments presented in this thesis utilize a Default 32-bit CRC polynomial for all Normal traffic, and the Koopman 32-bit CRC polynomial for Side-Channel traffic. Due to current hardware limitations, the ability to modify these CRC polynomials for the different channels in the experiments was not possible, and instead a MATLAB [38] script was used to alter the CRC of specific frames in post-processing. The process of modifying the CRC values will be covered more in-depth in Chapter 4.

2.5 – MANETs & OLSR

Mobile Ad-Hoc Networks, or as they're more colloquially known, "MANETs" are self-configuring wireless networks consisting of mobile devices. These networks often lack any defined infrastructure, and instead build a routing table on the fly (thus, Ad-Hoc). The routing table of a MANET often exists in a peer-to-peer (P2P) nature. Every device in a MANET can move independently and in any direction, which leads to a number of challenges, such as maintaining a routing table, and creating persistent connectivity [39]. In order for a MANET to operate successfully, each device must have the ability to continuously maintain a routing table in order to properly route traffic. Some protocols, such as OLSR, rely on using designated devices to keep all other nodes up to date on the available routes. MANETs are also known for their heterogeneity, meaning that these networks may contain multiple transceiver types, resulting in an even more complex topology.

There are several categorizations for MANETs, each with their own unique uses and purposes. Vehicle oriented MANETs (VANETs) [40] are typically used for inter-vehicular communication and communication from vehicles to roadside equipment. A more recent usage is that of Smart Phone Ad-Hoc Networks (SPANs) [41], which rely on hardware existing within current commercial smartphones in order to create P2P networks to communicate while circumventing typical carrier networks. Another core application for MANETs environments are military communication devices. The experiments shown in this thesis are meant to emulate a military-style MANET environment [42] using the OLSR routing protocol. This research could

also be applied to different non-military applications such as: Campus Networks, Smart Phone Networks, or other similar Ad-Hoc applications.

Optimized Link State Routing (OLSR) is a variation of the standard Link State Routing Protocol [43], whereby each node in the network will independently form a routing table by determining the best path to each destination node in the network. OLSR was intended to optimize link state algorithms for use on a wireless ad-hoc network, especially one featuring embedded devices, smartphones, or other similarly resource-limited devices. OLSR is an improvement upon standard Link State protocols in that each node selects a set of neighbor nodes known as “multipoint relays” (MPR). It is only these MPRs that forward control traffic, effectively reducing the number of transmissions required, and therefore the flood of control messages [44].

2.6 – Classification Quality Measures

The F-Score method (also known as F_1 score or F-measure) is commonly used in a variety of works for classification, such as: information retrieval, search measurements, document classification, and query classification [45]. This approach is often used when testing the effectiveness of a feature, such as the HD between two CRC values. Another area that makes use of the F-Score calculation is the evaluation of word segmentation or speech recognition. In their work, Sangwan et al. [46] relied on the use of the F-Score calculation when testing their keyword model for phone-based speech recognition. Using this calculation, they were able to

propose a new threshold estimation technique for the detection of keywords in conversational speech patterns.

The F-Score calculation, while useful in classification, is far from perfect in that it has issues determining an effective threshold when trying to detect multiple classes. Tao et al. [47] have identified some of the limitations of the F-Score approach in their work, and proposed a new method of calculating a weighted F-Score. The F-Score measurement has been shown to produce issues when there is inter-class overlapping or inconsistent features [47]. The suggested issues arise due to the fact that F-Score weighs all features equally, which may not be idea for certain experiment conditions. To combat this, Tao et al. suggest a technique useful for selecting the most effective features for classification by taking the average value of a feature in a dataset and comparing it against another feature on a per-feature basis. This ensures that the most accurate feature is selected for the given problem. Since the described Side-Channel detection method relies on just a single feature, any multi-class limitations of F-Score will not be an issue. The F-Score measurement is calculated based on the compounded harmonic mean of precision and recall. Further information regarding this calculation will be examined in Chapter 3.

Some problems require a more specialized approach, where Precision and Recall may not be considered equally weighted. Two other commonly used F-measures exist for this purpose, and are called the F_2 and $F_{0.5}$ measures. F_2 places more weight into recall than precision, while the $F_{0.5}$ measure is more heavily weighted towards precision. These alternative

"F_β" measures were proposed by Van Rijsbergen [48] as a means for when users place "β times as much emphasis on recall as precision". Other work conducted by Xie et al. [49] demonstrates a multi-feature variation on the F-Score approach, where each feature is further verified through the use of the SVM machine learning algorithm. The use of SVM for verification is harkened back to within this work (Section 5.5), as a means for determining the effectiveness of the HD metric.

Chapter 3 –Side-Channel Identification Based on an F-Score Quality Measure

3.1 – The F-Score as a Hamming Distance Detection Quality Measure

An HD threshold is a whole number at which the maximum number of Side-Channel messages are detected, while avoiding false alarms when naturally corrupted Normal messages have a non-zero HD. The difficulty when selecting a value is that if your threshold is too high you will easily miss large volumes of Side-Channel communication; alternatively, if the threshold is too low, you will end up with a large amount of false positives. There are several quality measures for this type of classification, but the primary measurement chosen for this threshold calculation is a concept known as F-Score. It is important to understand how this measurement works in order to recognize how effective the calculated threshold may be.

When classifying data the most common approach to verification is to assess the data against a trusted set of correctly identified results. In doing so, you are able to determine whether or not data has been flagged as True/False Positive, or True/False Negative. Two common quality measures exist, known as Precision and Recall, which take these classifications into consideration when determining the relevance of the classification. Precision focuses largely on what fraction of the results were relevant to the classifier, by taking the number of True Positives and dividing it by the total number of data points identified as Positive.

$$Precision = \frac{True\ Positives}{(True\ Positives + False\ Positives)}$$

Recall is oriented towards determining how successful the classification was, and does so by dividing the number of True Positives over the number of data points which should have been classified as positive (True Positives & False Negatives). As such, this value is heavily impacted by the number of False Negative classifications.

$$Recall = \frac{True\ Positives}{(True\ Positives + False\ Negatives)}$$

The following Table 3.1 should be used for reference in order to better understand the influence of each of the classifications when calculating a threshold value. These concepts are not just essential for understanding how F-Score calculates a threshold, but also pivotal in gauging a threshold based on an ROC curve, as shown in Section 5.4.

Term	Definition
True Positive (TP)	The classifier has correctly categorized a data point as **Side-Channel** communication.
True Negative (TN)	The classifier has correctly categorized a data point as **Normal** traffic.
False Positive (FP)	The classifier has incorrectly categorized a **Normal** traffic data point as **Side-Channel** communication.
False Negative (FN)	The classifier has incorrectly categorized a **Side-Channel** data point as **Normal** traffic.
Precision	How accurately the number of data points classified as **Side-Channel** was, when compared to the number of False Positives.
Recall	How accurately the number of data points classified as **Side-Channel** was, was when compared to the number of missed Side-Channel frames (flagged as False Negative).

Table 3.1 – Definitions of calculations / classifications utilized when calculating an F-Score value.

Both Precision and Recall are compounded in order to calculate a composite value known as F-Score. F-Score utilizes the Harmonic Mean [50] of Precision and Recall in order to find the best possible combination, or in this case, the *Optimal HD Threshold*. F-Score is displayed as a value which falls between 0 and 1, where 0 is considered highly inaccurate and 1 is considered perfectly accurate. F-Score can be calculated using the following formula:

$$F = 2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}$$

In the work presented by Chea [3], F-Score was used to define a threshold for experiments by selecting the threshold which presented the highest F-Score. This work builds upon this, but instead aims to present a precise *range* of thresholds that could be used to detect Side-Channel in a variety of situations. In order to calculate a threshold using F-Score, a dataset with known Positive and Negative samples must exist, along with some metric to test against a threshold value. Hamming Distance is used as the metric for this purpose.

When using F-Score to calculate a threshold, the first step is to select a range of threshold values to test. Each of these thresholds are checked against the HD metric of a dataset in order to determine the number of TP, FP, TN, & FN. For example, given a possible threshold range of 1-30, the HD value for each individual frame must be checked against the specific threshold in order to classify data. Precision and Recall calculations are then performed using the TP, FP, and FN values in the formulas described above. Once Precision and Recall have been determined for each of the thresholds being examined (in this case 1-30), F-Score

calculations may commence. Finally, the threshold value with the largest F-Score is determined to be the optimal threshold for the given dataset.

The two other variables that were considered for the purposes of selecting a Hamming Distance threshold were Accuracy and Specificity. Accuracy is the measurement of how close the measurement results are to the true value, and how reproducible these results are. While Accuracy appears to provide a solid value for determining a threshold, this value has been shown to operate poorly when dealing with inconsistent conditions, such as the disproportionate levels of Side-Channel with respect to normal traffic. Specificity is the percentage of correctly identified True Negatives. While this is also important to measure, this calculation ignores a decrease in the number of True Positives as the threshold increases in favour of a greater rate of True Negatives.

3.2 – Testing the F-Score Threshold

In Section 2.4 it was explained that the HD is calculated by performing an XOR on two CRC values. This HD provides a metric for anomaly detection, and F-Score provides a threshold. Knowing the total number of each frame type, along with the Hamming Distance values for all of the Normal & Side-Channel frames, allows a user to determine which threshold provides the most optimal detection. Determining whether or not a frame is flagged as Side-Channel is as simple as comparing the HD value to the threshold. For example, if an F-Score Optimal Threshold of 12 is selected, any frames with an HD value of 12 or greater would be considered Side-Channel whether or not they do in fact belong to Side-Channel communication. This makes

the selection of a threshold a delicate balance between the highest possible naturally corrupted Normal frame HD, and the lowest possible Side-Channel HD.

The following Figure 3.1 presents results from a rudimentary proof-of-concept Side-Channel experiment where Side-Channel was emulated through the use of FTP, and HTTP traffic was generated to represent Normal traffic in a 2-node scenario with 30% FER. The Hamming Distance was calculated for these frame types, where the Koopman 32-bit polynomial was used for Side-Channel CRC generation. As you can see, when frames become naturally corrupted their HD increases, and with an increased percentage of FER there is a larger volume of frames that could be incorrectly flagged as Side-Channel. Consider the scenario, and imagine a threshold of 12 is applied to it. The threshold would capture the eight Side-Channel frames with HD values in the 13 – 18 range, but unfortunately would also incorrectly identify eighteen naturally corrupted Normal frames that have an HD of 12 – 17. This demonstrates the challenge of determining a threshold, as selecting a value that is too high or low could result in inaccurate results.

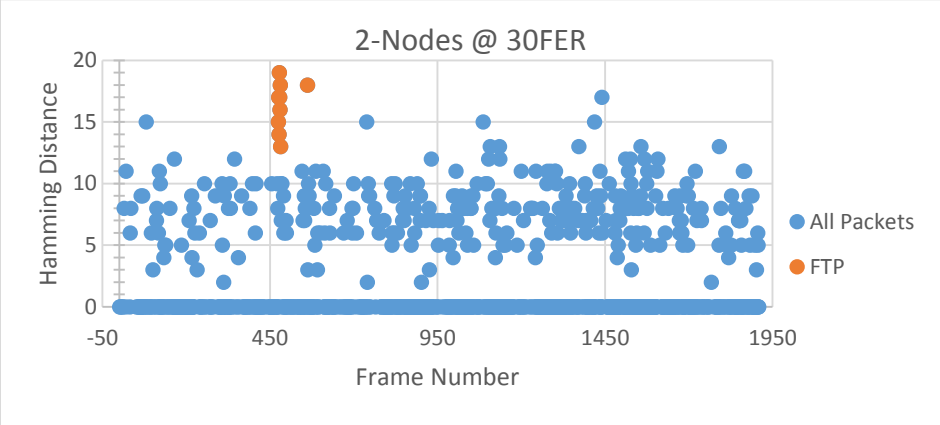

Figure 3.1 – Comparison of Side-Channel (FTP) Hamming Distance to Normal traffic.

In order to exhibit how F-Score is used in this decision, observe a snippet of the results from the F-Score calculation performed on this dataset in Table 3.2. The Threshold value of 16 is calculated as the best choice for this dataset due to the fact that it presents the highest F-Score. While a threshold of 14 shows a larger amount of True Positives, the number of False Positives raises. This demonstrates the concept of finding a harmonic mean.

Threshold	True(+)	False(+)	True(-)	False(-)	Accuracy	Sensitivity	Specificity	Precision	F-Score
14	7	13	1038	2	0.9858	0.7778	0.9876	0.3500	0.4828
15	6	4	1038	3	0.9933	0.6667	0.9962	0.6000	0.6316
16	6	2	1038	3	0.9952	0.6667	0.9981	0.7500	0.7059
17	4	1	1038	5	0.9943	0.4444	0.9990	0.8000	0.5714

Table 3.2 – F-Score Calculations on a dataset with 2-Nodes & 30% FER.

Chapter 4 – Experimental Design

The goal of the experiments is to simulate a military foot soldier platoon's ad-hoc communication; as such, the selected MANET protocol, the positioning of the nodes [51], and the communication methods have all been kept in line with potential real-world scenarios [52]. Further details will be outlined in the sections below.

4.1 The Hybrid Experimental Environment

There are two possible routes that could have been pursued when developing experiments for this problem. Option 1 involves conducting experiments in a real network using real data, with two challenges in doing so: Should it exist, hardware capable of establishing a Side-Channel must be used; and if not, then a method for emulating the behavior of Side-Channel must be clearly defined. Option 2 is to use a network simulator, which outputs files for processing or allows for modification of source code in order to implement desired functionalities. Due to the obscure nature of the Side-Channel problem, it was unlikely that network simulators would offer this functionality off the shelf.

In his work, Najafizadeh [11] aimed to develop a system in which a Side-Channel link could be established. Previous works in the topic [6] [10] had shown that a form of Side-Channel communication may be possible through modification of the FCS field. Taking the results of those works into account, Najafizadeh [11] tested these hypotheses. This work also outlined the issues related to many of the simulator options, and described an exhaustive

search for compatible hardware. The Sinalgo [29] network simulator was chosen as the medium for simulation. Sinalgo [29] is an open source Java-based network simulator which provided core functionality necessary for simulation, and allowed for easier modification than other simulators such as NS-2 [53] or QualNet [11]. The elegant simplicity of Sinalgo left many features to be desired that would need to be added by Najafizadeh [11] before he could begin testing his hypothesis. One of the limitations of Sinalgo that was recognized in his work [11] was the fact that corrupted frames were simply discarded by the simulator, rather than being transmitted to the recipient. In addition to this, several other issues which he would later resolve were lack of a proper channel fading model, and a need for promiscuous agent nodes who could monitor and capture traffic.

With little to no available hardware for Side-Channel testing, the most likely option for experimentation was to use a simulation environment. Simulators provide an inexpensive, scalable solution for testing, and typically allow for easily modifiable parameters. Network simulators in particular also often provide channel models, routing, and full TCP stacks. With support for the OLSR protocol, the ability to instantiate controlled mobility, and featuring a full TCP/IP protocol stack, QualNet [30] appears to be the ideal Simulator for Side-Channel experiments. Unfortunately, there are several issues when attempting to post-process or analyze data from QualNet scenarios, as frames generated within experiments are actually devoid of any useable information, such as FCS values.

One of the challenges facing the development of Side-Channel experiments is a distinct lack of hardware with the capability of establishing a genuine Side-Channel. Due to this limitation, performing experiments through simulation seems to be the most logical step; however, these simulators often lack the necessary functionality when it comes to transferring an actual data frame, as well as calculation and transmission of corrupt frames. Instead, a way to emulate this behavior is needed. In order to establish Side-Channel via the methods proposed in the previous works of Szczypiorski [6] or Najafizadeh [11], it must be possible to allow an application to generate its own FCS, which, as with all MAC layer operations, is a functionality that is locked into the firmware of most current 802.11 wireless network cards.

A chipset known as the Atheros AR5212 developed by Qualcomm [55] supports a flexible MAC layer allowing for modification of the device's CRC algorithm. Unfortunately, devices with this chipset are no longer in production, and have become scarcely available. Without the ability to generate an alternative CRC using hardware, a substitute method for emulating Side-Channel communication is necessary. The emulated Side-Channel must also be easily recognizable during the analysis phase in order to allow for modification in post-processing. The simplest way to execute this is to establish a Constant Bitrate (CBR) communication on a port that differs from the rest of the network traffic.

Chea [3] attempted to modify and recompile the source code for QualNet in order to implement FCS capabilities and frame information, but was unsuccessful in his efforts. His hypothesis was attempted in the QualNet simulator, tested using MATLAB with Simulink, and

finally accomplished with hardware using an emulated Side-Channel [3]. The QualNet [30] simulator was considered, but presented a lot of challenges regarding the implementation of a payload and CRC calculation. When unable to correctly analyze scenario information from QualNet, Chea [3] experimented with hardware environments, using FTP as an emulated Side-Channel. This allowed for the ability to conduct detection on a network without needing the proper hardware required to modify the FCS field. The network described for this work was similar to that of a hub-and-spoke, with five nodes communicating to a central "Server" node and an agent node collecting data. Chea was able to show that there was a distinction between Hamming Distances of Side-Channel and non-SC nodes, with very simple experimental procedures. He developed a system known as the "Hybrid Testing Approach" [3] where results from a Wireshark [54] capture file were manipulated using MATLAB [38] to generate CRCs and FER. This "Hybrid" approach was borrowed for the processing and preparation of data for this thesis. Building from that, the work within this thesis aims to expand upon and improve much of what was shown in [3], while bringing it to a MANET environment.

The Hybrid Experimental Environment involves parsing files captured from Wireshark, and executing a number of functions in MATLAB to organize the data into a format that can be correctly analysed. After six files created through the TShark scripts and Java program have been generated, the format of which are described in Section 4.5, a MATLAB script (see Appendix A) takes them as input. MATLAB/Simulink are the tools responsible for the actual CRC modification that constitute the emulated Side-Channel. With the frames parsed through TShark and Java, several output files have been created, but these files are simply the first step

towards collecting the data into a format that can be analyzed. As mentioned above, a different CRC polynomial is used for the purpose of Side-Channel than that of Normal traffic. Figure 4.1 illustrates the process interactions between MATLAB and the parse files described above. The "olsrFCSCheck.txt" file is used by MATLAB in conjunction with the "hexDataDump.txt" file in order to identify which of the frames are good and which are naturally corrupted. Any naturally corrupted frames are automatically discarded to prevent skewing of the FER generation process. Next, the "olsrPorts.txt" file is used to determine if the received frame was destined for Port 1337, or another port. This is done to determine if the frames should be treated as Side-Channel or Normal traffic. Control frames and port 80 destined data frames are considered "Normal" traffic. Once a frame's port, and corresponding type have been identified, they are forwarded to a decision gate and on to the CRC generation algorithm. If the frame was destined for Port 1337, a CRC is generated using the Koopman CRC polynomial (simulated Side-Channel) while all other frames compute a CRC based on a Default 32-bit CRC polynomial. At this point in the simulation process, each of these frames are treated as uncorrupted, and the Normal frames would be shown to have a HD value of 0 if compared to the expected polynomial.

Once the appropriate CRC is generated for each frame, it is appended onto the end and simulation of channel properties can begin. The probability of each frame being erroneous is considered, and if it is selected to be so, a number of bits is flipped. The frame is given a chance to become corrupted based on the probability value inputted into the "Frame Error Probability Decider" function, and if it is not corrupted then it is sent on to have its HD calculated. If the frame is selected to become corrupted, it is passed through an "AWGN" channel for corruption.

The SNR value for this channel comes from the "*SINRFile.txt*" that was generated earlier. After corruption is complete, the corrupted frames are also sent on to have their HD calculated.

In the final portion of the MATLAB script, the original Default polynomial is compared against the CRC of the current frame. From this, the HD value is calculated through an XOR of the two polynomials. If the HD value is 0, the frame is a non-corrupted Normal frame; otherwise, if there is an HD value that is greater than 0, the frame is either part of the Side-Channel communication or a corrupted Normal frame. The HD value will always be a positive integer relative to the number of bits different it is from the expected CRC. The HD value for each frame is then reported and output to a file. In addition to this, several values are calculated and output to files at this point, such as: frame count statistics and F-Score calculations (True Positives, False Positives, True Negatives, False Negatives, accuracy, sensitivity) for a range of thresholds (1 – 30).

While this system allows for the emulation of a physical Side-Channel network, it does have a few limitations that must be addressed. Firstly, in order to increase the number of nodes in the experiment, one must re-run the physical experiments with an additional node. Secondly, the errors generated for each of the frames chosen for corruption are based on the AWGN channel model, and may only be as good as the channel model. This means that there may be some bias in regards to the accuracy of the Hamming Distances generated for corrupted frames.

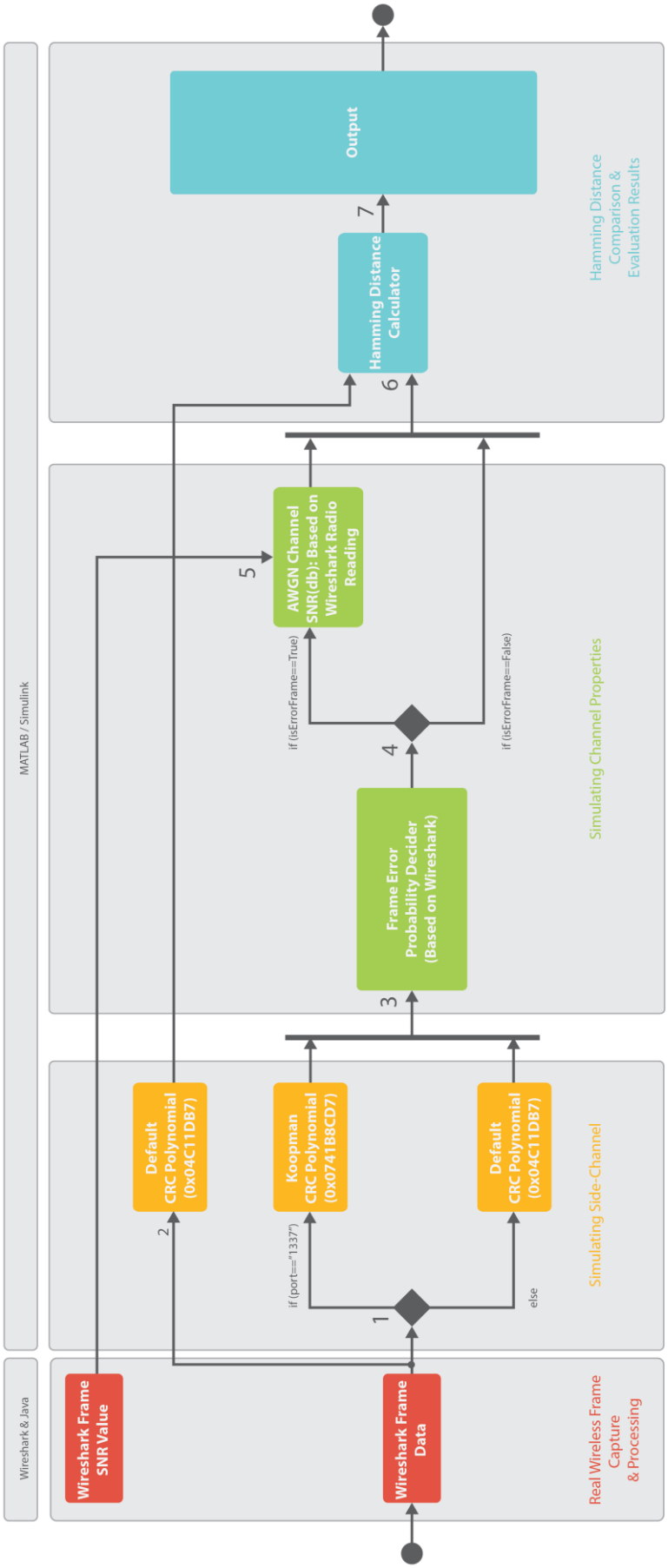

Figure 4.1 – Hybrid Test Environment Model Flow Diagram [3]

4.2 – Platooning: Forging a Formation

Platoon formations define the expected arrangements of soldiers in relation to each other. The goal of these formations is to provide as much flexibility to adapt to situations as needed while maintaining control of a unit. In almost all formations, Team and Squad Leaders are in the front, allowing for these individuals to lead by example, and as such all soldiers within a platoon are required to have line of sight on their leader at all times. The platoon formation is typically selected by a leader after he or she has considered factors such as Mission objectives, Enemy, Terrain, Troops, and Time available (METT-T) [52]. The selection of this formation should ideally provide maximum protection, and allow for the maintenance of unit cohesion, stable momentum, and a smooth transition between offensive and defensive actions.

In an attempt to design experiments that represent military platoon scenarios as accurately as possible, several sources were examined when considering node placement. The "Fire Team Wedge" formation [52] is the most basic formation a fire team can select. This formation provides the unit with visibility of the Team Leaders, while covering a large patrol area. The interval between soldiers in this formation is suggested to be 10 meters, however this inter-operative distance is variable depending on visibility, terrain conditions, availability of space, or other factors affecting the functionality of the wedge. The inter-operative distance may shrink or expand in order to ensure visibility of the squad leader. According to the FM 7-8 Infantry and Platoon Field Manual [52], it is not uncommon for a wedge to contract to the point where units may move in single file, if for example the platoon has entered into an indoor

environment. Figure 4.2 has been provided for reference of a standard Fire Team Wedge. The flexibility of the inter-operative spacing allowed for some modification to the platoon positioning in the experiments. Due to spatial limitations, and given that volunteers did not have radios for communication, inter-operative positioning was reduced to a standard distance of 5 feet.

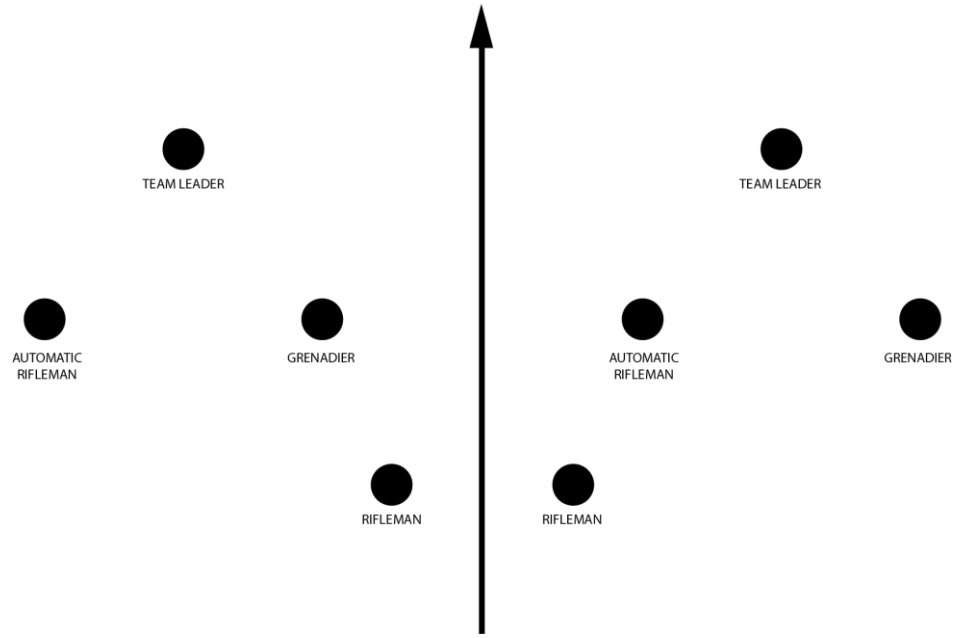

Figure 4.2 – The Fire Team Wedge, a standard platoon formation [52].

Both the number of soldiers and layout of the Fire Team Wedge were selected as parameters for the experiment. Not only did this formation provide a more realistic approach to node positioning, it also allowed for maximum control while directing volunteers. Based on the requirements of the experiments, and because the volunteers were not military trained, the Fire Team Wedge was selected both for simplicity, and to represent possible positioning during a standard patrol mission.

4.3 – The OLSR Routing Protocol

For the experiments, a stable version of the OLSR protocol (0.6.8) was installed and run on the wlan0 interface using Ubuntu 14.04. Figure 4.3 shows a screen capture of OLSR's output from a laptop with the IP Address 10.10.10.15. A similar output was printed to the terminal by OLSR on a per second basis as the routing table information was requested; however, the frequency of this update is configurable and can be increased or decreased as needed. In this capture, the links of each of the nodes on the network are present, along with their Link Quality (LQ) and Expected Transmission Count (ETX). Following this, the nodes which are considered direct one-hop neighbours to each node, and whether or not they are MPRs, are also observable. Finally, the list of neighbours that are accessible through two-hops are listed, along with their total cost and prospective routes.

Each of the nodes in the scenarios were provided with a static IP Address belonging to the 10.10.10.0/24 network. By providing static IP Addresses, analyzing captured data was easier, and allowed for a clear overview of the network activities in the live feed displayed in Wireshark [54].

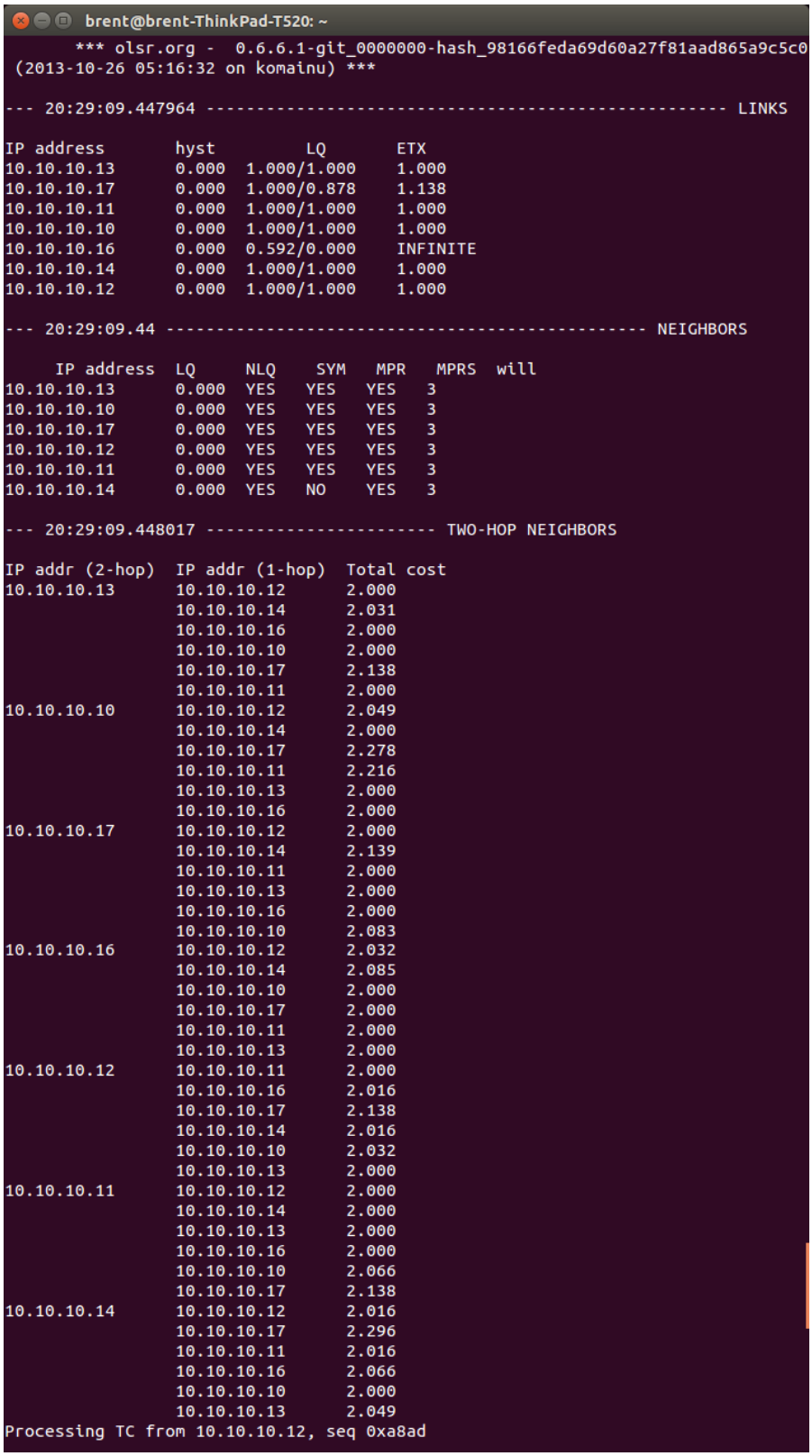

Figure 4.3 – A screen capture of the Ubuntu terminal showcasing the OLSR routing table output.

4.4 – Physical Experimental Scenarios

In each of the experiment scenarios, the nodes formed a typical platoon of 8 members as defined by the Michigan Tech AROTC [51]. Squad members consisted of UOIT and Durham College volunteers, all of whom were informed of their right to withdraw, and their required tasks for the experiment, all of which was approved by UOIT's Review Ethics Board (REB). A marker was used to signify an Observation Post that the squad needed to reach within the allotted five minute experiment scenario. During the experiment, volunteers were instructed to maintain a relative distance of 5 feet while walking. This distance was not arbitrarily chosen, but rather a scaled down variation of the standard 10m inter-operative patrol distance, as per the guidelines outlined in the sourced platoon field manuals [51] [52]. The key observation of this experiment was to monitor the relative effectiveness of the CRC HD metric in a MANET environment while subjected to normal interference. As such, participants in the experiments were merely a means of transporting devices in order to provide mobility, and as such no bias was made in regards to volunteers.

Seven of the volunteers were responsible for carrying seven of the platoon nodes, and one volunteer was responsible for both the eighth node and the agent node. The arrangement of this was to illustrate that the agent could exist as a module within one of the nodes. Figure 4.4 showcases the positioning and location of nodes for various times throughout the scenario. The hardware for each of the nodes was a Lenovo T520 laptop, with a second generation i5 CPU and 8GB of RAM. Each of the devices were configured using the Ubuntu 14.04 [56] operating

system, with the OLSR routing protocol installed. Ubuntu was chosen due to the fact that at the time of writing, there is currently no implementation of OLSR for non-Linux operating systems. The agent node was also a Lenovo T520 with the same hardware listed above; however, this device was running Windows 7, the Wireshark traffic analysis tool, and a USB network capture card. Volunteers maintained a constant walking speed and relative distance while following the predetermined route. It is also worth noting that due to the proximity of these devices, there was a full overlap, and all nodes could reach one another. The experiments were performed outdoors during overcast weather conditions with some light rain. Proof of concept experiments that were performed on similar conditions were also performed in an indoor setting and upon comparison, similar results were achieved within an FER percentage of 0.2%.

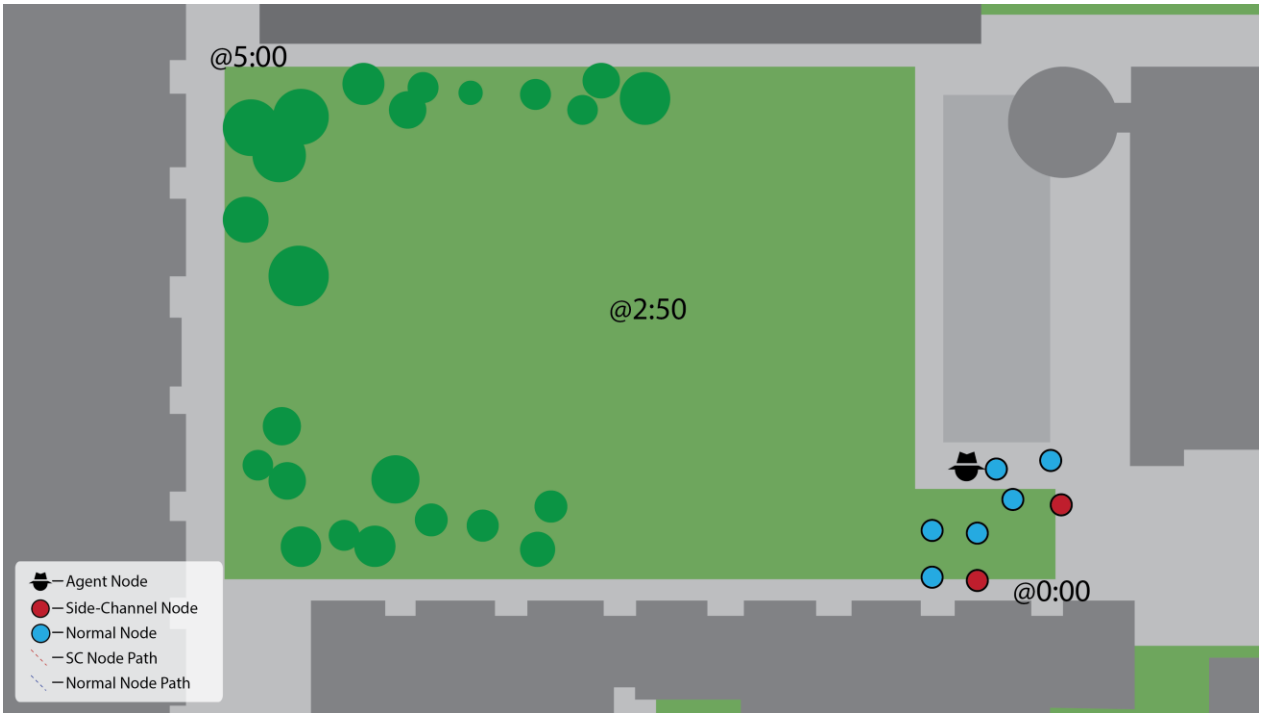

Figure 4.4 – A topology diagram detailing the communication and positioning for the OLSR Experiment.

In the experiments conducted by Madtha et al. [2] using the QualNet simulator, a Constant Bitrate (CBR) application was utilized for traffic generation. Drawing from this, it was discovered that an application known as “Nping” [57] provided desirable functionality for the experiments. Nping is an open source network packet generation tool that is commonly used in networks for measuring response times, detecting active hosts, and can even be used to generate raw packets for stress testing, ARP poisoning, or Denial of Service attacks. The Nping tool is versatile, and provides the ability to control the rate of transmission, number of packets sent, and destination port. This tool was chosen for the experiment since it would allow for raw packets to be transmitted across distinct ports. For the experiments, Side-Channel traffic was transmitted across port 1337, while “Normal” traffic was sent through port 80. In the experiment scenarios that were run, the rate of Side-Channel transmission was varied in order to test the HD detection technique against different ratios of Side-Channel to normal traffic communication.

Each of the packets generated by Nping were 86 bytes, a relatively small size chosen in order to illustrate that the proposed technique could be used in even the most minimal traffic. Communication existed between multiple source and destination nodes, in order to generate as much traffic as possible while mitigating possible hardware bottlenecks when nodes were transmitting and receiving at the same time, and was sent using UDP. Figure 4.5 shows the communication links and the direction of transmission between each of the nodes in the platoon, where the blue links represent the traffic sent via port 80, and the red link represents the Side-Channel communication transmitted via port 1337. The agent node was not included

in the OLSR network, but instead simply captured nearby traffic in a promiscuous state using Wireshark [54] and the AirPCap TX [58] network capture card.

The direction of communication for these experiments consisted of nodes 10.10.10.10, 10.10.10.11, and 10.10.10.16 transmitting Normal traffic to the nodes ending in 14, 12 & 15 respectively, while the malicious node 13 was responsible for transmitting Side-Channel traffic to 17. For reference, the Team Leader position is represented by nodes 10 & 14, the Grenadiers are 11 & 15, the Automatic Riflemen are 12 & 16, and the Riflemen are 13 & 17.

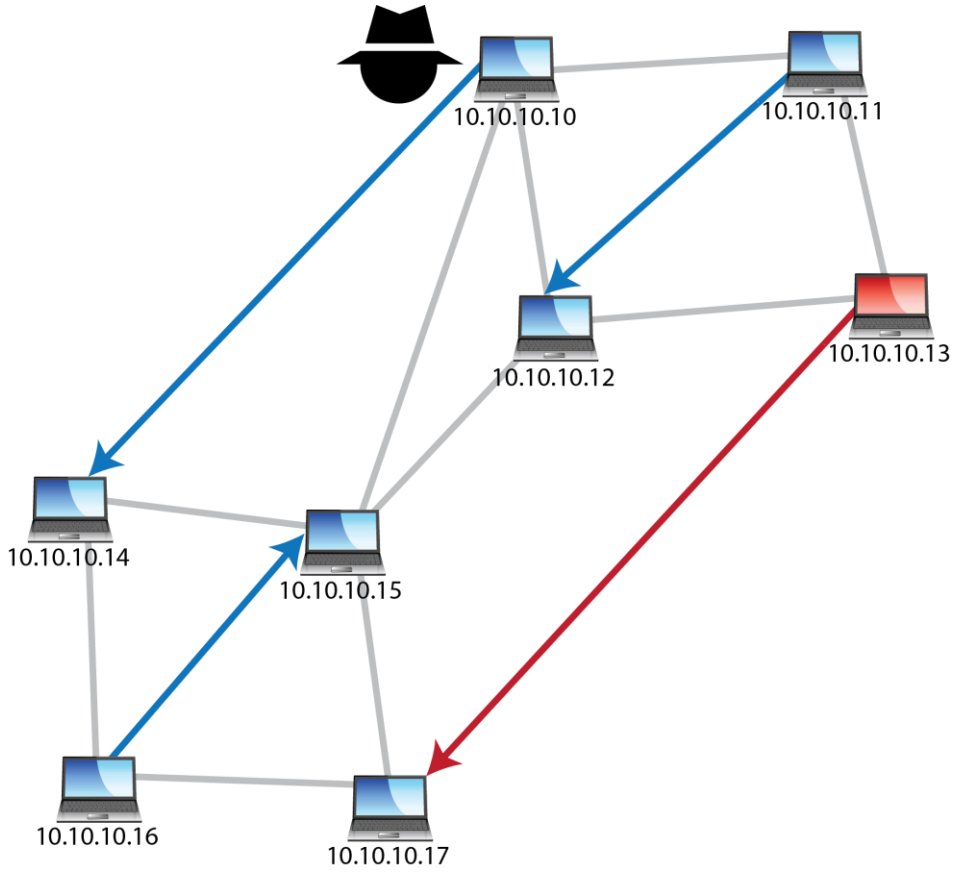

Figure 4.5 – Demonstration of the communication links between the Side-Channel and normal traffic.

Two parameters were varied for the experiments: the ratio of Side-Channel to normal communication, and the percentage of Frame Error Rate. While the ratio of Side-Channel to Normal traffic required running an additional experiment with each increase in volume, the modification to the percentage of FER was achievable offline in post-processing. During each of the five-minute experiments, Side-Channel communication would begin at the one minute and thirty second mark, and continue for ninety seconds. The experiments were run a total of 12 times, with the number of Side-Channel messages per second increased by an additional message each time in order to provide varying ratios of Side-Channel to normal traffic. The range of Side-Channel to normal traffic began with 9% of the total traffic as side channel, corresponding to one Side-Channel frame per second being transmitted over a 90 second window. The percentage was increased by gradually adding an additional Side-Channel frame per second, until 40% of the traffic consisted of Side-Channel frames. The conclusion of experiments at 40% Side-Channel was not an arbitrary choice, but as will be shown in Chapter 5, an observable threshold plateau began to appear. The scenarios consisted of the following percentages of Side-Channel traffic: 9%, 12%, 14%, 19%, 22%, 25%, 28%, 30%, 35%, 37%, and 40%. These percentages correlated directly to an additional 1 Side-Channel frame per second. Adding or subtracting nodes from the network would of course directly increase/decrease the percentage of Side-Channel traffic relative to the change in normal traffic, and is a variable which could be examined in future experiments.

As previously mentioned, there was a second controlled variable: the modification of the Frame Error Rate. Modification of this value was performed during post-processing through

the use of MATLAB and Simulink, where a select percentage of communication was artificially corrupted in order to increase the amount of noise. FER was modified in order to examine the effectiveness of the Hamming Distance metric for various levels of noise on the network. With increased error rate, the number of non-Side-Channel frames featuring higher than expected HD should be exponentially higher than instances with little to no noise. Through post-processing, 22 different levels of FER were introduced to each of the 12 experiments, creating 264 unique datasets for analysis. More information regarding the process of FER modification is shown in Section 4.5.

4.5 – Pre-processing the Data

With the data from the experiments captured and stored within a Wireshark file, it was important to parse it properly. All of the relevant data needed to be kept, while traffic not belonging to the OLSR network had to be stripped out and ignored. Due to the nature of monitoring wireless traffic, there are often a lot of packets captured which may belong to other networks within range. Fortunately, as part of its standard installation Wireshark includes a command line interface known as TShark. TShark allows a user to capture information similarly to Wireshark, but without a GUI. This tool also allows for the output of select frames and information from a capture file into a text file based on user-defined filter criteria. Several of the commands used for the filtering of traffic can be seen below in Figure 4.6.

```
1) tshark -r "captureFile.pcapng" -T fields -e wlan.fcs_good >>
   allFCSCheck.txt

2) tshark -Y "udp.port==1337 || udp.port==80 || olsr" -r "captureFile.pcapng"
   -T fields -e wlan.fcs_good >> olsrFCSCheck.txt

3) tshark -Y "udp.port==1337 || udp.port==80 || olsr" -r "captureFile.pcapng"
   -T fields -e udp.dstport >> olsrPorts.txt

4) tshark -Y "udp.port==1337 || udp.port==80 || olsr" -r "captureFile.pcapng"
   -T fields -e radiotap.db_antsignal >> SINRFile.txt

5) tshark -Y "udp.port==1337 || udp.port==80 || olsr" -r "captureFile.pcapng"
   -x >> hexDataDump.txt
```

Figure 4.6 – TShark Filter commands used in the generation of files.

The five commands above, while all appearing nearly identical, actually provide a variety of information. The first command outputs a file called “allFCSCheck.txt”, which contains a binary list defining whether or not each of the frames captured suffered from natural error. This initial list is for all frames received, whether or not they were related to OLSR or the experiment. If a frame was corrupted, a value of 0 will be displayed for that particular frame. It is important to note that each of the output files generated will feature the frames in the order that they were received by the AirPCap device. The second command generates a file similar to that of command 1, with the exception that it filters out and displays only frames which have been transmitted using UDP with a destination of port 1337 or port 80, or if they were simply using the OLSR routing protocol. These filters allowed for the capture of all OLSR control packets, Side-Channel packets, and generated “Normal” traffic in the experiment. The third command also filters frames using the same criteria, but this time outputs a file titled “olsrPorts.txt”, containing the port values of all of the frames relevant to the experiment for the

OLSR network. The fourth command generates the “SINRFile.txt” file, containing the Wireshark calculated Signal-to-Noise ratio for each captured frame. Finally, command 5 generates a file containing the data bytes captured by Wireshark in their hexadecimal format.

With the data of the captured frames output to a file, it is then necessary to convert them from Hexadecimal into a format that can be analyzed. Using a java program originally created by Chea [3], the dumped hexadecimal data was serialized into a binary format and output to a file for use by MATLAB. Figure 4.7 shows an example of the type of conversion performed on a frame by the program. On the left are the Wireshark bytes in their hexadecimal format. These bytes represent the frame’s fields and data. To better evaluate what is provided in this data, it is important to recognize what is present in this text. The first column containing clusters of four digits is used to signify the position of each line within the hex dump. The start of each new frame is clearly defined by Wireshark through the use of the value “0000”. The middle values, shown in clusters of two hexadecimal digits, contain the actual data portion of the frames. Lastly, the rightmost column is an ASCII translation of the hexadecimal numbers within the frame. For the purposes of the Java program, only the first two columns are considered. The program captures each line of hexadecimal and then converts it into a binary sequence, appending the lines together in order to form a single binary sequence for each frame, removing the bits correlating to the FCS field. These bits are removed so that a new CRC may be incorporated into them using MATLAB. After all of the frames have been assessed by the program, a text output file is generated, and MATLAB post-processing may begin. Figure

4.8, presents a process diagram of the entire parsing process from start to finish. The significance of each of these files has been detailed in-depth in Section 4.1.

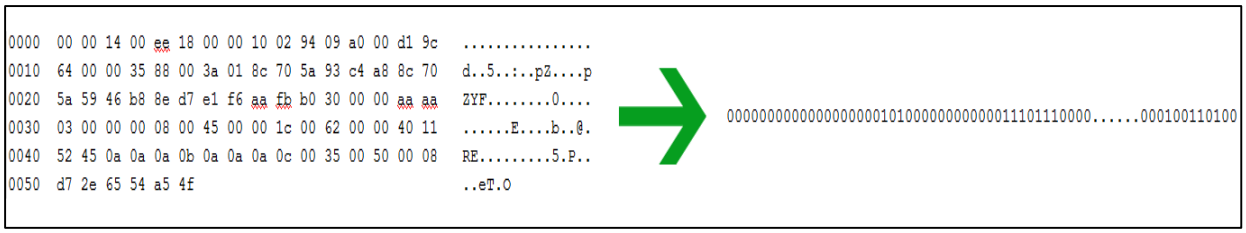

Figure 4.7 – An example output of Chea's [3] TShark hex dump Java program.

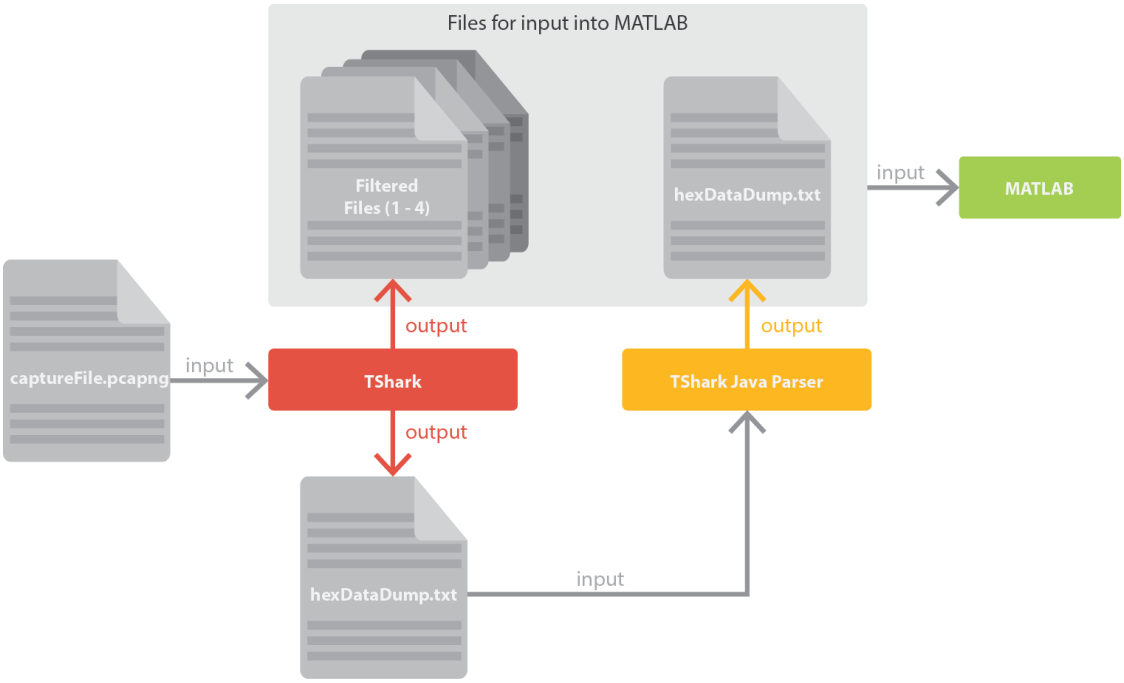

Figure 4.8 – Process diagram illustrating the steps involved in parsing Wireshark capture files for MATLAB.

Chapter 5 – Analysis

The goal of this analysis chapter is to determine whether or not it is possible to define a threshold, or range of thresholds that will detect Side-Channel communication with as few false negatives or false positives as possible in a MANET. Another important task is to compare the results presented by Chea [3] against the MANET results, to determine whether his stationary network could utilize the same threshold range presented below.

5.1 – Performing the F-Score Calculation

Using F-Score for calculation, threshold values from 1 – 30 were tested for each of the experiment scenarios, with the value presenting the highest F-Score value ultimately being chosen as the threshold as presented in Section 3.2. The following information presented in Table 5.1 shows the statistical information for a scenario with 14% of the total traffic consisting of Side-Channel communication, and using the actual (unmodified by MATLAB) calculated FER.

Percentage of Frame Error Rate:	*0.83%*
Total Number of Frames:	*2,680*
Total Number of Uncorrupted Frames:	*2663*
Total Number of Naturally Corrupted Frames:	*17*
Total Number of Non-SC (Normal) Frames:	*2316*
Total Number of Side-Channel Frames:	*364 (14%)*

Table 5.1 – Statistical information regarding the 14% SC scenario using its unmodified 0.83% FER

The process of calculating F-Score was described in-depth in the above Chapter 3, and as such will not be re-examined here. Instead, observe the results of the F-Score calculations for proposed threshold values from 1 – 30 in Table 5.2. With an F-Score value of 0.9973 (on a scale from 0 – 1), the threshold of 9 was selected in this scenario. Upon further examination of the F-Score results, it is evident that while a threshold value of 9 did not have the highest number of True Positives, it did in fact have a lower number of False Negatives than some of the higher thresholds. A threshold of 10 or higher had even fewer False Positives, but began to present a larger number of False Negatives, continuing this trend as the threshold grew higher. This suggests that the F-Score Calculation places a higher level of significance to the best harmonic combination of False Positives and False Negatives when considering a threshold value.

Threshold	True(+)	False(+)	True(-)	False(-)	Accuracy	Sensitivity	Specificity	Precision	F-Score
1	364	14	2302	0	0.9948	1	0.994	0.963	0.9811
2	364	14	2302	0	0.9948	1	0.994	0.963	0.9811
3	364	14	2302	0	0.9948	1	0.994	0.963	0.9811
4	364	12	2304	0	0.9955	1	0.9948	0.9681	0.9838
5	364	11	2305	0	0.9959	1	0.9953	0.9707	0.9851
6	364	8	2308	0	0.997	1	0.9965	0.9785	0.9891
7	364	6	2310	0	0.9978	1	0.9974	0.9838	0.9918
8	364	5	2311	0	0.9981	1	0.9978	0.9864	0.9932
9	363	1	2315	1	0.9993	0.9973	0.9996	0.9973	0.9973
10	361	0	2316	3	0.9989	0.9918	1	1	0.9959
11	352	0	2316	12	0.9955	0.967	1	1	0.9832
12	331	0	2316	33	0.9877	0.9093	1	1	0.9525
13	310	0	2316	54	0.9799	0.8516	1	1	0.9199
14	279	0	2316	85	0.9683	0.7665	1	1	0.8678
15	223	0	2316	141	0.9474	0.6126	1	1	0.7598
16	159	0	2316	205	0.9235	0.4368	1	1	0.608
17	107	0	2316	257	0.9041	0.294	1	1	0.4544
18	63	0	2316	301	0.8877	0.1731	1	1	0.2951
19	35	0	2316	329	0.8772	0.0962	1	1	0.1754
20	15	0	2316	349	0.8698	0.0412	1	1	0.0792
21	5	0	2316	359	0.866	0.0137	1	1	0.0271
22	2	0	2316	362	0.8649	0.0055	1	1	0.0109
23	0	0	2316	364	0.8642	0.00	1.00	NaN	NaN
24	0	0	2316	364	0.8642	0.00	1.00	NaN	NaN
25	0	0	2316	364	0.8642	0.00	1.00	NaN	NaN
26	0	0	2316	364	0.8642	0.00	1.00	NaN	NaN
27	0	0	2316	364	0.8642	0.00	1.00	NaN	NaN
28	0	0	2316	364	0.8642	0.00	1.00	NaN	NaN
29	0	0	2316	364	0.8642	0.00	1.00	NaN	NaN
30	0	0	2316	364	0.8642	0.00	1.00	NaN	NaN

Table 5.2 – Demonstration of F-Score results used to select a threshold of 9 for the given scenario.

5.2 – Assessing a Threshold Range

The F-Score calculation was tested against the data from 252 out of the total 264 experiments in order to determine the possibility of a potentially universal threshold, or range of thresholds that would allow for the detection of a Side-Channel in any network. The scenarios for an FER of 0% were omitted for these calculations, as even though the experiments featured an incredibly low actual FER (0.4% - 0.8%), the likelihood of establishing communication with an absolutely 0% FER can be considered virtually impossible. Table 5.3 contains the calculated Optimal HD Thresholds for these experiments, where each column represents a different percentage of Side-Channel communication, and each row is the percentage of simulated FER. By graphing these calculated thresholds, such as in Figure 5.1 or Figure 5.2, it initially appears as though there is very little consistency in thresholds. As the percentage of FER increases, the threshold fluctuates seemingly at random. While there is some observable patterning in the HD, analyzing the data in this way provides what appears to be a range of thresholds from an HD value of 5 up to an HD value of 15. This spread of threshold ranges is far too large to be considered effective, and establishing a cohesive range of thresholds would be unreliable. Fortunately, there are ways to narrow down these threshold values into an acceptable range.

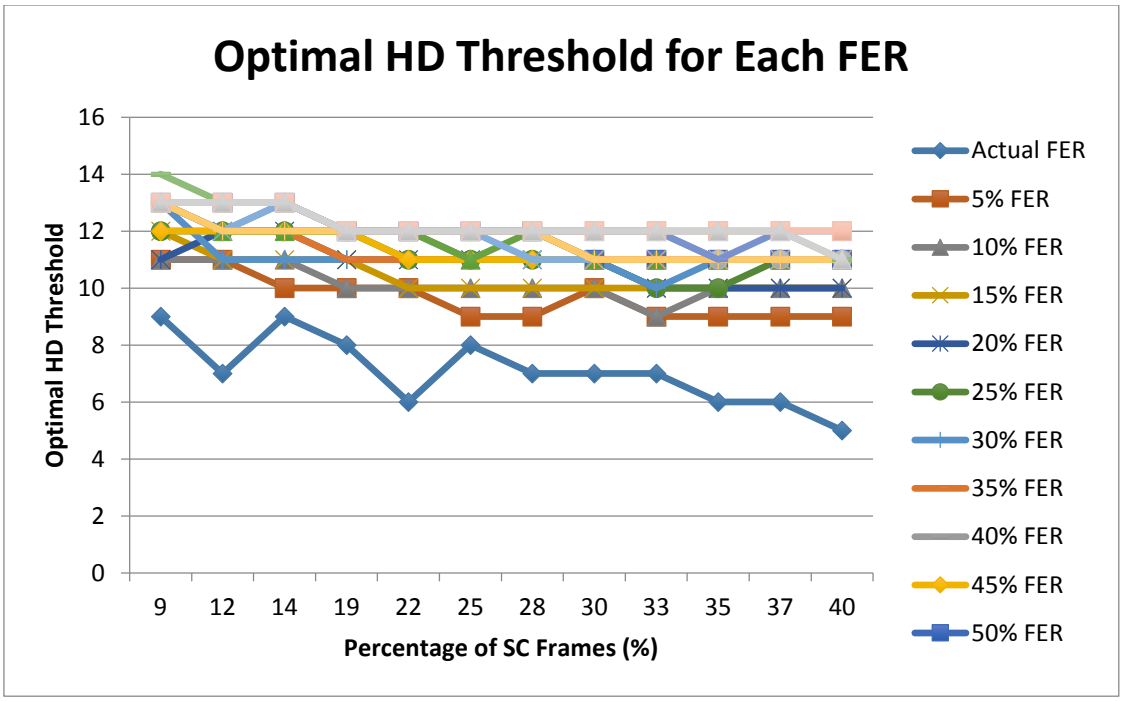

Figure 5.1 – A graph showing the calculated Optimal HD Threshold for each SC percentage where each line represents the percentage of FER.

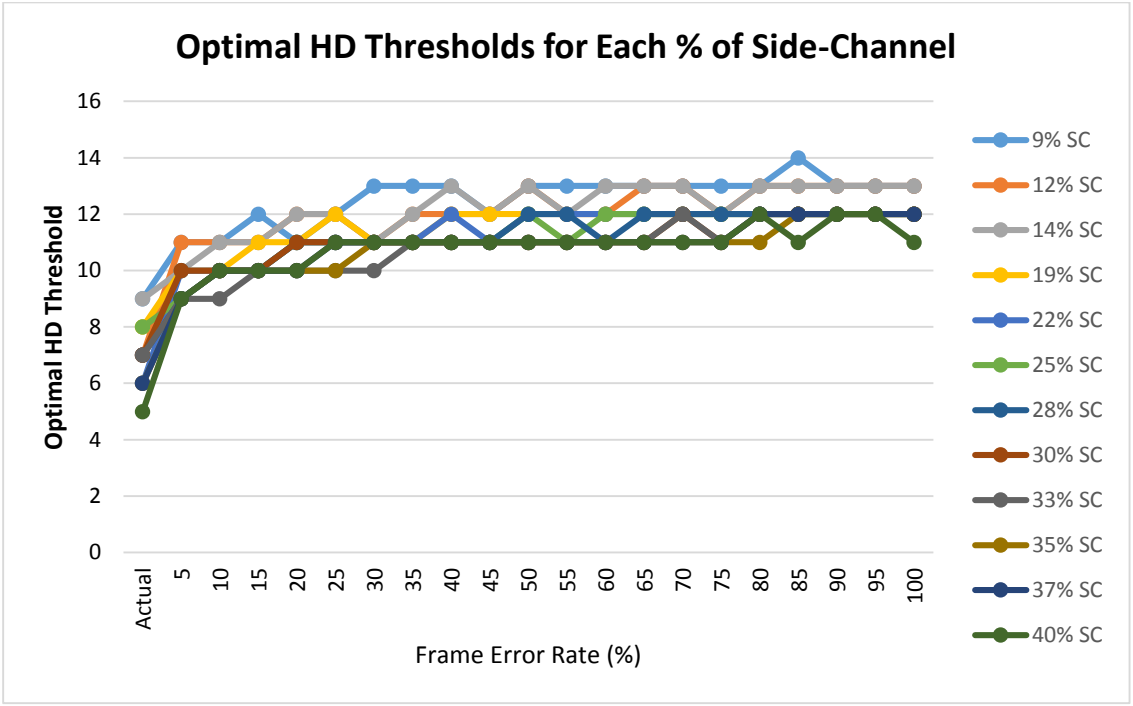

Figure 5.2 – An FER based examination of the calculated Optimal HD Thresholds.

While the huge variance in calculated thresholds makes determining a narrow range of thresholds appear to be quite difficult, further inspection of the data suggests that this is not actually the case. Observe Table 5.3, and notice that the mean, median and mode of these thresholds fall between 10 and 13, and begins to normalize after the percentage of Side-Channel increases beyond 25%. Taking into consideration how HD operates, you cannot suggest that an HD value is not a whole number; you must round up or down (i.e. you cannot have an HD that is 11 ½ bits different from the expected CRC). By rounding these HD threshold means, a consistent range of thresholds with an average mean of 11 – 12 appears, all calculated with a substantially high typical F-Score hovering around 0.99.

By calculating the Standard Deviation of the thresholds for each of the varying amounts of Side-Channel, it is possible to say with 95% and 99% confidence that the range of Optimal Thresholds still falls between the 11 – 12 range (with rounding), as shown in Figures 5.3 and 5.4. In addition to this, as the percentage of Side-Channel communication increases past 19%, the mean threshold remains relatively consistent, with a maximum variance of 0.6.

	9%	12%	14%	19%	22%	25%	28%	30%	33%	35%	37%	40%
0% FER	1	1	1	1	1	1	1	1	1	1	1	1
Actual FER	9	7	9	8	6	8	7	7	7	6	6	5
5% FER	11	11	10	10	10	9	9	10	9	9	9	9
10% FER	11	11	11	10	10	10	10	10	9	10	10	10
15% FER	12	11	11	11	10	10	10	10	10	10	10	10
20% FER	11	12	12	11	11	11	11	11	10	10	10	10
25% FER	12	12	12	12	11	11	11	11	10	10	11	11
30% FER	13	11	11	11	11	11	11	11	10	11	11	11
35% FER	13	12	12	11	11	11	11	11	11	11	11	11
40% FER	13	12	13	12	12	11	11	11	11	11	11	11
45% FER	12	12	12	12	11	11	11	11	11	11	11	11
50% FER	13	13	13	12	12	12	12	11	11	11	11	11
55% FER	13	12	12	12	12	11	12	11	11	11	11	11
60% FER	13	12	13	12	12	12	11	11	11	11	11	11
65% FER	13	13	13	12	12	12	12	11	11	11	11	11
70% FER	13	13	13	12	12	12	12	12	12	11	11	11
75% FER	13	12	12	12	12	12	12	11	11	11	11	11
80% FER	13	13	13	12	12	12	12	12	12	11	12	12
85% FER	14	13	13	12	12	12	12	12	12	12	12	11
90% FER	13	13	13	12	12	12	12	12	12	12	12	12
95% FER	13	13	13	12	12	12	12	12	12	12	12	12
100% FER	13	13	13	12	12	12	12	12	12	12	12	11

Mode	13	12	13	12	12	12	12	11	11	11	11	11
Mean	12.4	12.0	12.1	11.4	11.2	11.1	11.1	11.0	10.7	10.7	10.8	10.6
Median	13	12	12	12	12	11	11	11	11	11	11	11
Variance	1.26	1.85	1.29	1.06	1.96	1.23	1.59	1.25	1.61	1.73	1.79	2.15
SD	1.12	1.36	1.14	1.03	1.40	1.11	1.26	1.12	1.27	1.32	1.34	1.47
Kurtosis	3.26	8.65	1.44	5.44	9.42	2.20	4.77	7.50	2.33	7.63	7.80	11.36
Skewness	-1.69	-2.55	-1.33	-2.23	-2.79	-1.53	-2.01	-2.28	-1.34	-2.36	-2.43	-3.05
95% C.I.	0.47	0.57	0.47	0.43	0.59	0.46	0.53	0.47	0.53	0.55	0.56	0.61
99% C.I.	0.62	0.75	0.62	0.56	0.77	0.61	0.69	0.61	0.70	0.72	0.73	0.80
Low. Limit	12.53	11.43	11.53	11.57	11.41	10.54	10.47	10.53	10.47	10.45	10.44	10.39
Up. Limit	12.90	12.52	12.57	11.86	11.78	11.61	11.62	11.42	11.25	11.22	11.32	11.23

Table 5.3 – Optimal HD Thresholds for all 264 unique experiment combinations.

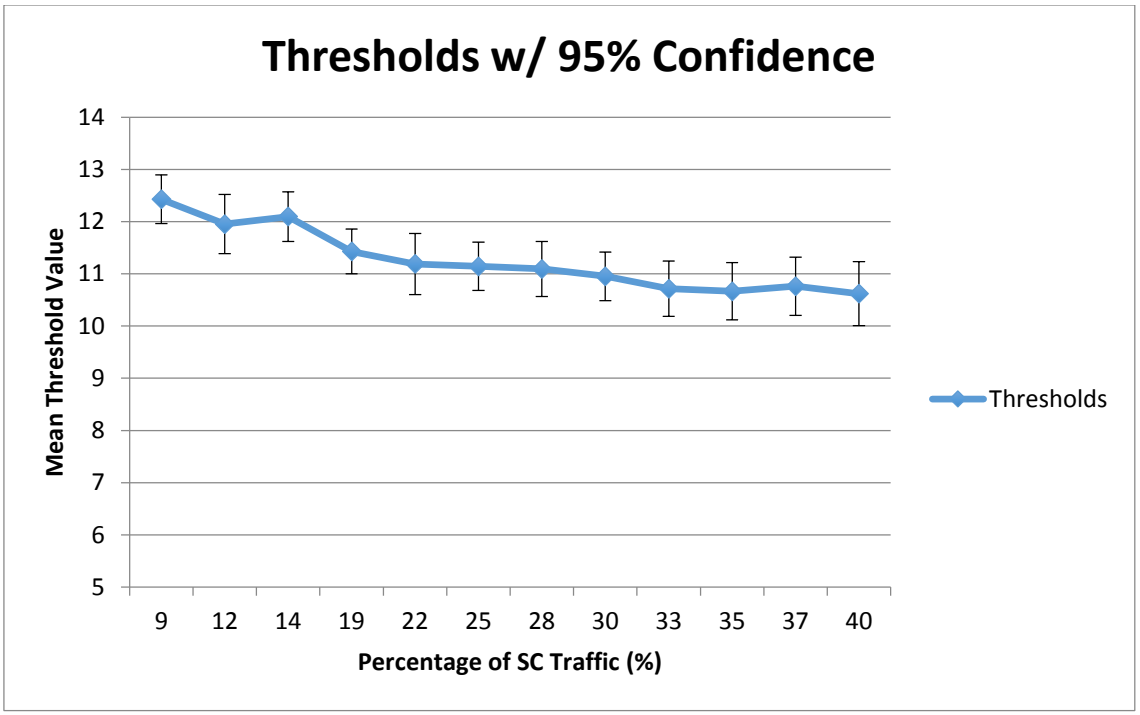

Figure 5.3 – Mean HD thresholds shown to fall within the suggested 11 – 12 range, with 95% confidence.

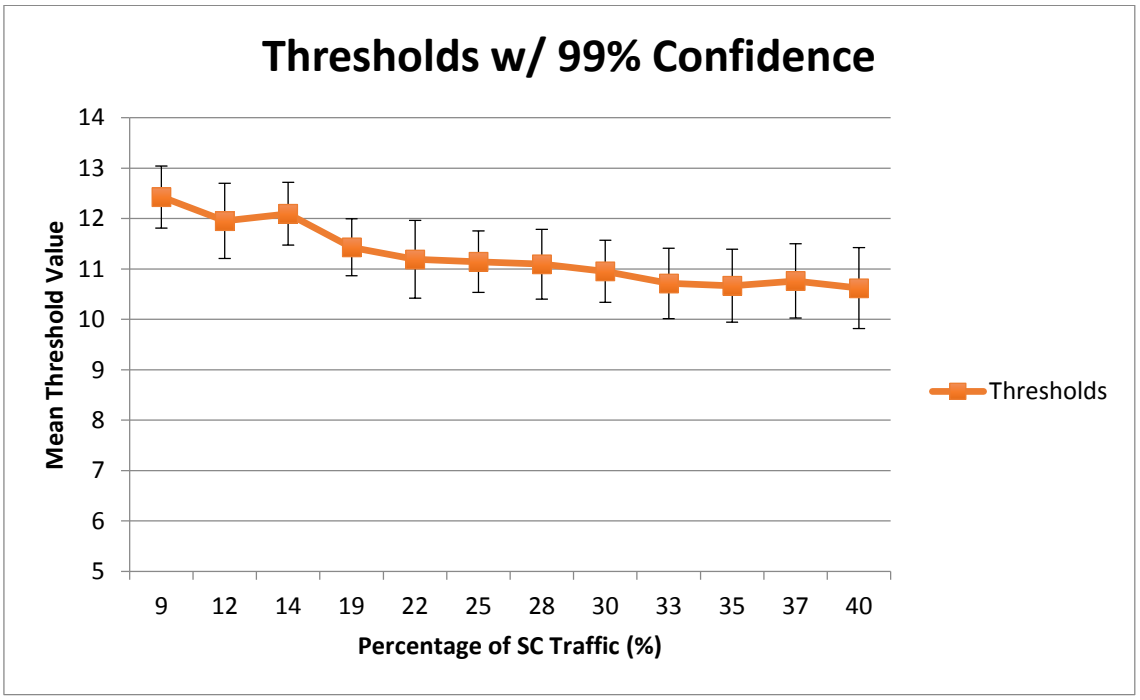

Figure 5.4 – Mean HD thresholds shown to fall within the suggested 11 – 12 range, with 99% confidence.

5.3 – Using the Threshold to Find a Side-Channel

Now that F-Score has been used to determine a threshold value, the results of the experiments must be tested against these thresholds. As was demonstrated in Section 5.1, the threshold calculation with the highest F-Score value should represent the most accurate value that avoids a large volume of False Positives and Negatives. By graphing the HD of each of the frames from one of the individual scenarios, it is possible to visually demonstrate how well these thresholds may actually perform when differentiating Side-Channel frames from normal traffic.

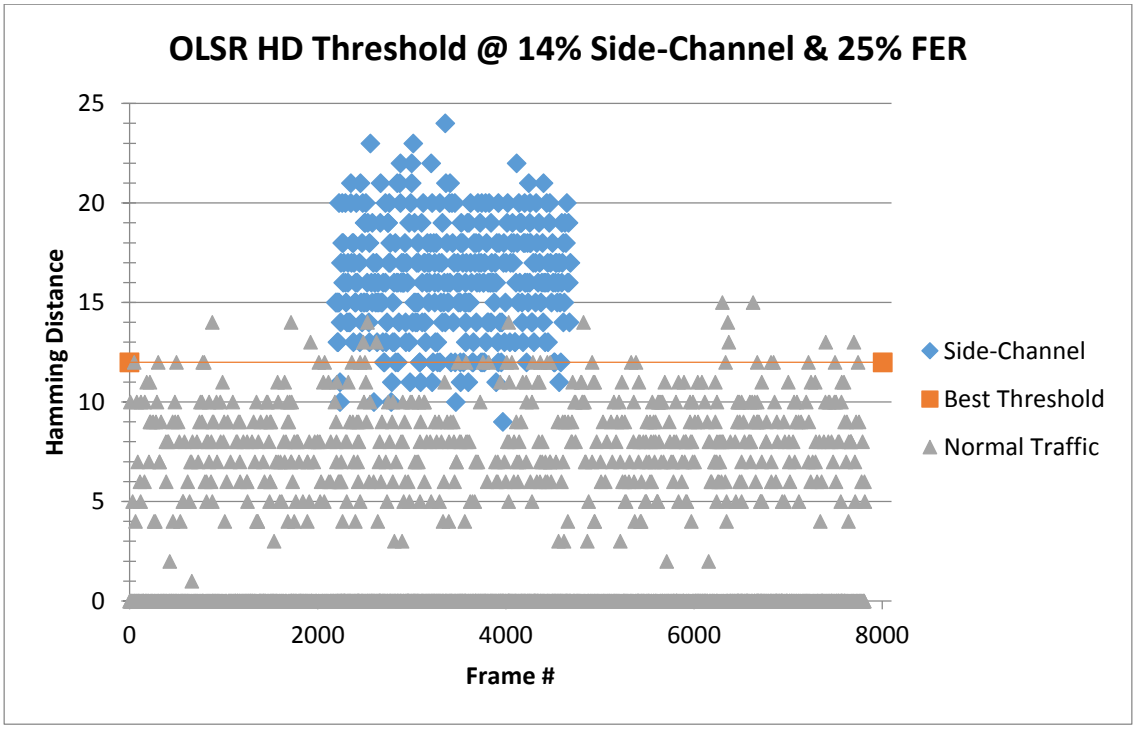

Figure 5.5 – A visual representation of the HD values for all frames in a scenario with 14% Side-Channel traffic & 25% Frame Error Rate.

As depicted in Figure 5.5 above, even with a higher volume of naturally corrupted frames, there remains a distinct difference between the HD of purposely corrupted Side-

Channel frames and any naturally corrupted Normal Traffic. The question is, how accurate is this trend across multiple scenarios? If you take the mean of the HD for Side-Channel and Normal traffic from a population sample of all experiments featuring their actual FER percentages (Table 5.4), a distinct difference in these values is evident between traffic types. Using the suggested threshold range described above (11 – 12), one can see that this range fits nearly centered between the two HD mean trend lines in Figure 5.6.

	9% SC		12% SC		14% SC		19% SC		22% SC		25% SC		28% SC		30% SC		33% SC		35% SC		37% SC		40% SC	
	Norm	SC	Norm	SC	Norm	SC	Norm	SC	Norm	SC	Norm	SC	Norm	SC	Norm	SC	Norm	SC	Norm	SC	Norm	SC	Norm	SC
Mean	8.0	15.7	8.8	16.2	7.1	16.2	8.2	16.0	7.8	16.0	7.4	16.1	7.7	16.0	8.0	16.0	8.8	15.9	8.8	15.9	6.7	16.1	8.9	16.0
Median	8.5	16	10	16	7	16	8	16	7.5	16	8	16	8	16	7.5	16	8	16	9	16	6	16	9	16
Mode	9	15	10	16	9	16	10	17	6	16	8	16	7	15	7	15	7	17	8	16	6	17	12	16
Variance	3.82	7.47	5.69	7.80	3.92	6.33	7.69	7.98	8.97	7.75	4.55	8.28	6.42	7.96	2.00	8.36	12.44	8.00	4.44	7.94	5.47	8.01	12.29	8.26
SD	0.63	4.45	0.56	5.49	0.53	5.61	0.52	6.44	0.40	6.80	0.42	7.15	0.51	7.31	0.30	7.47	0.53	7.64	0.49	7.75	0.31	7.97	0.57	8.00
Kurtosis	0.18	-0.62	-1.16	0.12	-1.21	-0.15	-0.41	-0.16	-1.34	-0.27	-1.15	-0.06	1.48	-0.11	1.50	-0.10	-0.62	-0.31	2.55	-0.29	-1.29	-0.13	0.03	0.11
Skewness	-0.97	-0.04	-0.37	0.07	-0.18	-0.03	0.20	-0.13	0.17	0.05	-0.53	-0.05	-1.02	-0.06	1.41	0.11	0.02	0.00	0.37	-0.08	0.60	0.02	-0.61	-0.02
C.I. (95%)	0.36	0.66	0.36	0.62	0.28	0.58	0.34	0.58	0.32	0.55	0.29	0.54	0.30	0.52	0.29	0.50	0.34	0.49	0.32	0.47	0.25	0.47	0.34	0.45
C.I. (99%)	0.47	0.87	0.48	0.81	0.36	0.76	0.45	0.76	0.42	0.73	0.38	0.70	0.40	0.68	0.39	0.66	0.45	0.64	0.42	0.62	0.33	0.61	0.44	0.59
95% L.L.	8.14	15.34	9.64	15.38	6.72	15.42	7.66	15.42	7.18	15.45	7.71	15.46	7.70	15.48	7.21	15.50	7.66	15.51	8.68	15.53	5.75	15.53	8.66	15.55
99% L.L.	8.03	15.13	9.52	15.19	6.64	15.24	7.55	15.24	7.08	15.27	7.62	15.30	7.60	15.32	7.11	15.34	7.55	15.36	8.58	15.38	5.67	15.39	8.56	15.41
95% U.L.	8.86	16.66	10.36	16.62	7.28	16.58	8.34	16.58	7.82	16.55	8.29	16.54	8.30	16.52	7.79	16.50	8.34	16.49	9.32	16.47	6.25	16.47	9.34	16.45
99% U.L.	8.97	16.87	10.48	16.81	7.36	16.76	8.45	16.76	7.92	16.73	8.38	16.70	8.40	16.68	7.89	16.66	8.45	16.64	9.42	16.62	6.33	16.61	9.44	16.59

Table 5.4 – Population means of each of the scenarios at their "actual" FER value, with Standard Deviation, Variance, Kurtosis, Skewness, 95/98% Confidence Intervals, and Upper & Lower Limits.

By calculating the confidence level results at 95% and 99% confidence for each of the Normal and Side-Channel HDs, it can be stated that each of the HD means of Normal frames fall within the range of 8.0 +/-0.32 with 95% confidence, and 8.0 +/-0.42 with 99% confidence. Additionally, the population mean for Side-Channel frames falls within the range of 16.0 +/-0.54 with a confidence level of 95%, and 16.0 +/- 0.70 with a confidence level of 99%. These results suggest that the appropriate threshold should exist somewhere between the calculated population means of the two values. To better interpret these results, please refer to Figures 5.6 & 5.7, which showcase the calculated HD population means at 95% and 99% confidence, respectively.

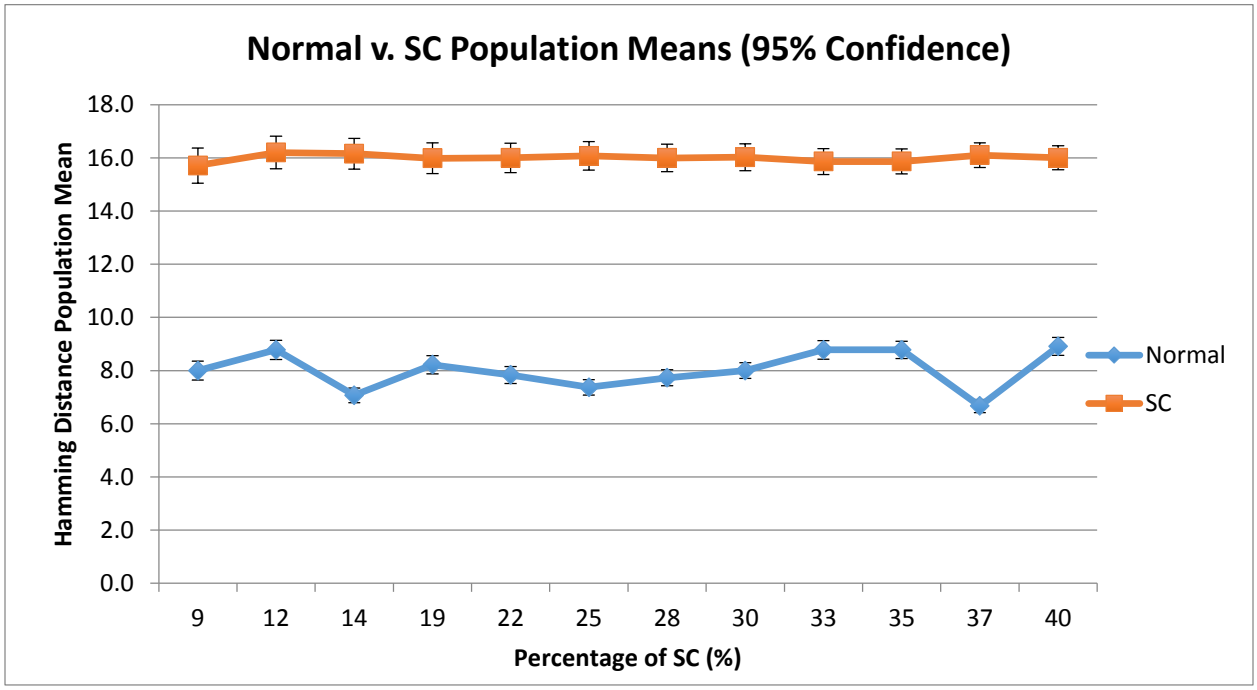

Figure 5.6 – The HD population means for Normal and Side-Channel traffic at 95% confidence.

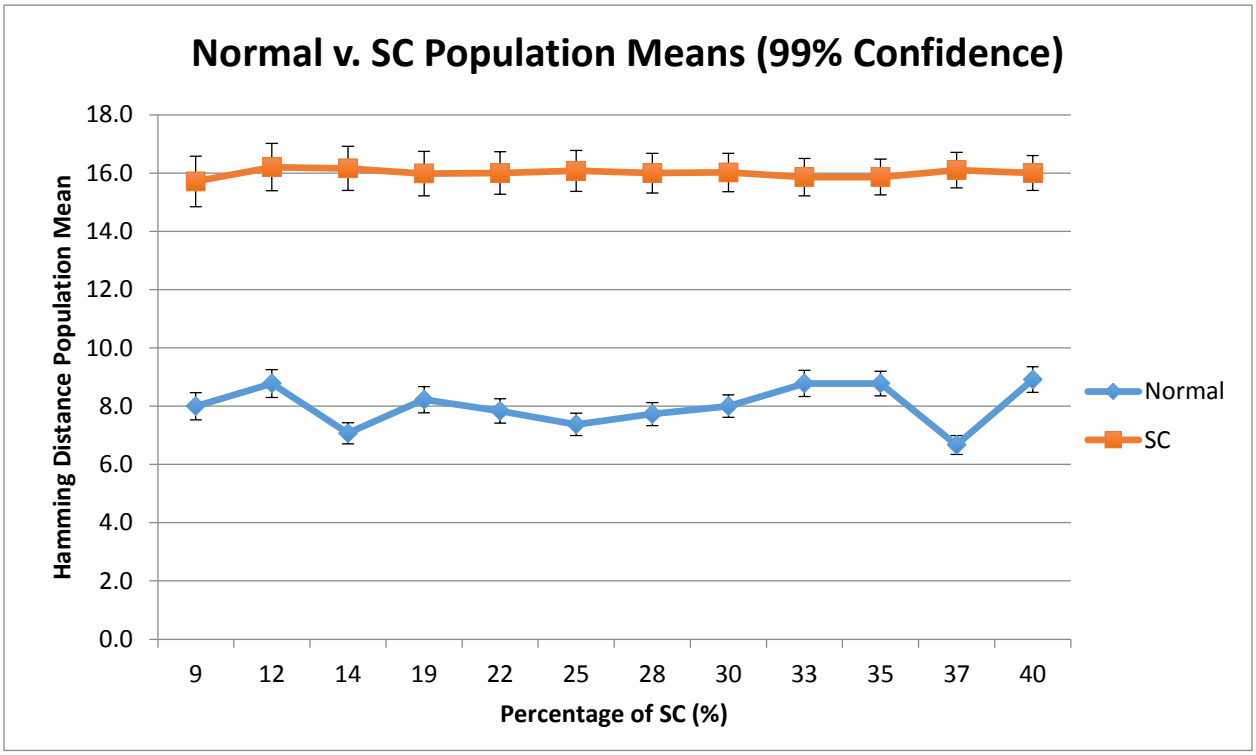

Figure 5.7 – The HD population means for Normal and Side-Channel traffic at 99% confidence.

Looking back at the proposed threshold range for the sum of the experiments (Figures 5.3 & 5.4), it was stated with 95% and 99% confidence that the threshold would exist within a range of 11 – 12. This is well within the isolation of the population means shown above for all of the tested percentages of Side-Channel traffic.

In his work, Chea [3] suggested that a threshold would exist in the range of 15 – 16, but the results shown above advocate that the threshold should in fact exist within the realm of 11 – 12. This difference in threshold is expected; as shown in Table 5.3, variation in Side-Channel frames will effectively skew the position of the mean Optimal Threshold based on the percentage of Side-Channel, until a plateau is reached at 19% Side-Channel traffic. In the results

presented by Chea [3], the total percentage of Side-Channel traffic was roughly 1% for his experiments. This implies that the threshold directly correlates to the ratio of Side-Channel traffic for a given period, and provides further proof that a Windowing technique could allow for more precise threshold detection in real-time. With F-Score shown to be capable in its ability to define a threshold based on HD, it is important to ensure the validity of the HD metric itself. Section 5.4 & 5.5 respectively explore the use of the Receiver Operating Characteristic curve and the Support Vector Machine learning algorithm to determine if the HD metric is useful for classification, or if the F-Score calculation presents a biased level of effectiveness.

5.4 – ROC Curves

The Receiver Operating Characteristic (ROC) curve is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) (1 – Specificity) for all possible thresholds of a diagnostic experiment. The technique provides information on how effective a given feature is for a particular set of data. The calculation for the TPR is the same as Sensitivity (also referred to as "Recall"), which has been shown in Section 3.1 as part of the F-Score calculation and assesses how positive a given threshold is for the technique. The TPR and FPR calculations rely on the count of True/False Positives and True/False Negatives for a given threshold tested against a dataset. The general purposes of an ROC curve are to provide the following [59]:

1. Demonstrate a tradeoff between Sensitivity and Specificity, where increasing one decreases the other

2. The closer that the area under the ROC curve is to 1, the better the classifier is at detection

3. The closer the curve droops to the central 45-degree diagonal of the graph (less area under the curve), the less accurate the threshold can be considered

Assessing a threshold technique using an ROC curve is relatively simple: the more a curve represents a 90-degree angle, the more optimal the test is for that particular dataset [59] [60]. A prediction method with the best possible outcome would yield a point at the upper leftmost corner of the ROC space, which represents 100% Sensitivity (no False Negatives) and 100% Specificity (no False Positives). If an arbitrary guess at classification is taken, then the ROC curve should appear as a point along a 45-degree diagonal line, also referred to as the "Line of No Discrimination." Figure 5.8 illustrates this concept, where Point A represents a perfect classification, Point B represents a moderate classification, Point C represents a poor classification. Point D may also represent an exceptional classification, as a curve along this point would describe the presence of Normal traffic. In the case of F-Score thresholds, an ideal threshold should have a high TPR, whilst maintaining a low FPR. Not all data is equal, so a technique may be more or less effective for a particular dataset. Performing an analysis using ROC curves will demonstrate how effective the calculated HD thresholds are for a given Side-Channel experiment.

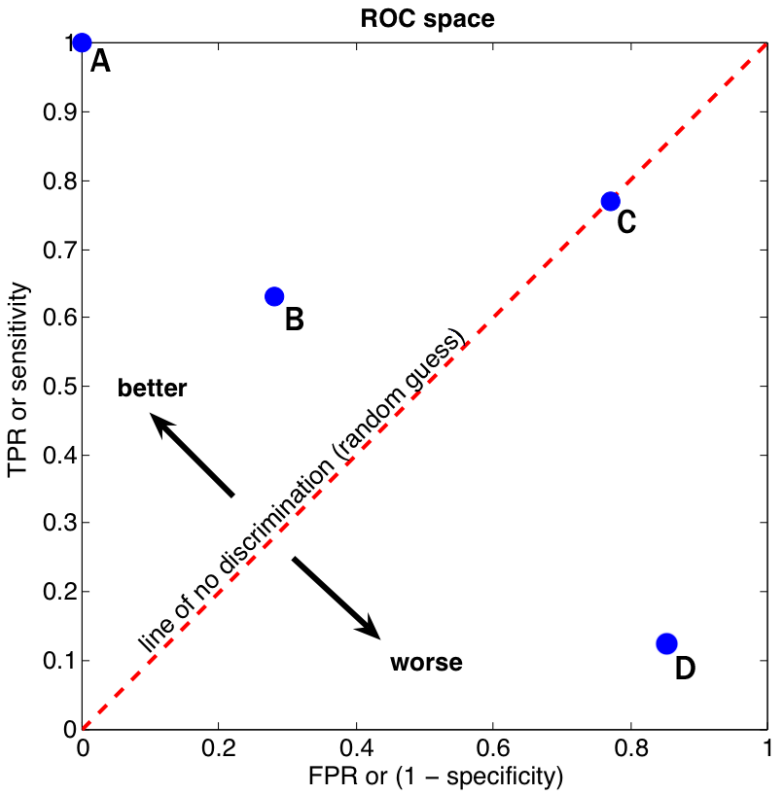

Figure 5.8 – Determining the effectiveness of a point in the ROC space [61].

Figure 5.9 presents the ROC curves calculated for the HD thresholds on a variety of scenarios. These scenarios range from 9% to 25% Side-Channel, and from 0.8% (Actual FER) to 95% FER. This wide range of scenarios was arbitrarily chosen to demonstrate how effectively the HD metric can be used across a large variation in scenarios. As mentioned above, the closer the area under the curve approaches 1, the more effective the test is against a particular dataset. Figure 5.9 demonstrates just how effective thresholds based on HD are against the Side-Channel problem.

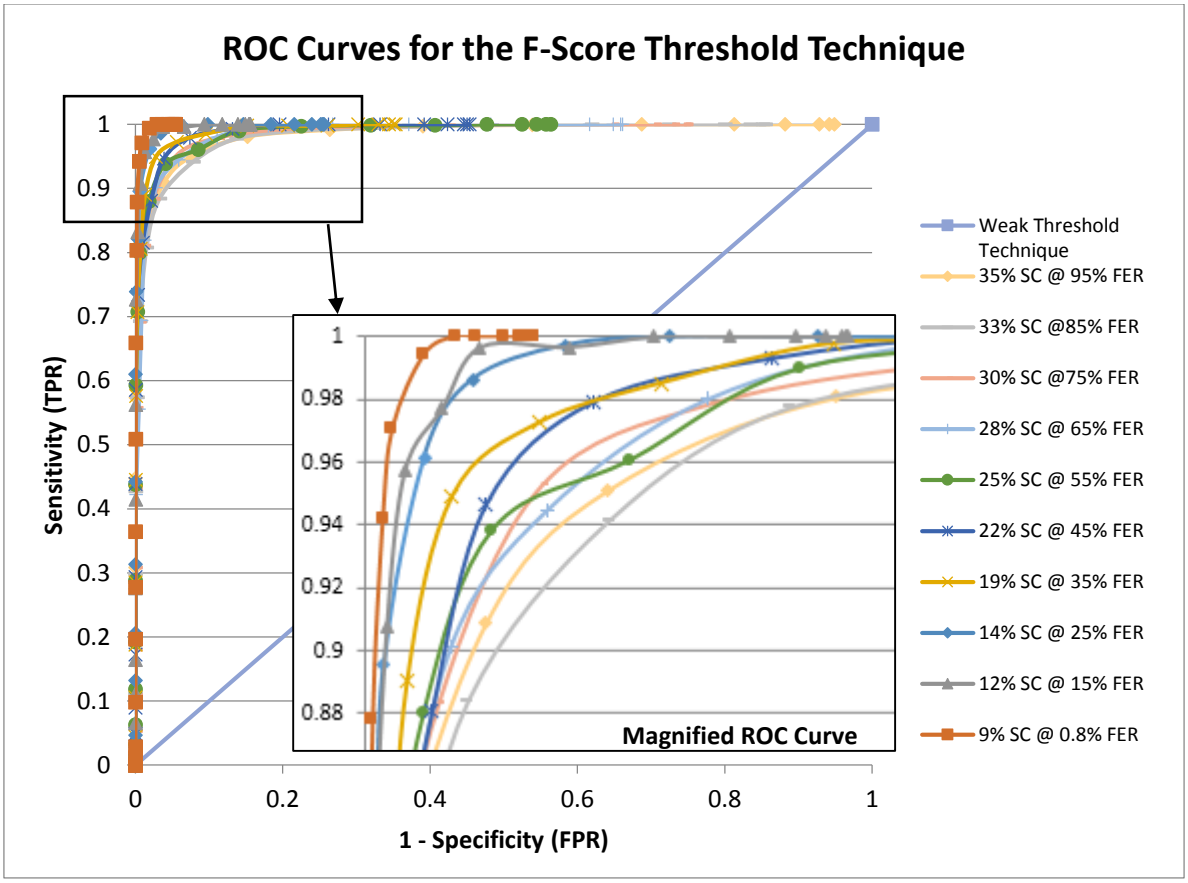

Figure 5.9 – The ROC curve demonstrating the effectiveness of the F-Score thresholds, with a magnified view of the upper left corner.

As shown above, this technique suggests that the F-Score approach to defining thresholds is very effective when tested against the HD metric. It should be noted that the ROC curve is strongly influenced by False Negatives, and as such the scenarios with lower FER (less naturally corrupted Normal frames) depict a much higher area under the curve. There are some similarities between the ROC & F-Score approaches in that they both rely on the use of Sensitivity & Specificity; while these techniques are assessed very differently, it is important to disprove any bias that may be caused by similar variables. In order to do so, Section 5.5 explores the use of the SVM machine learning algorithm as an alternative means of utilizing the HD metric for detection of Side-Channel.

5.5 – Support Vector Machines

For the purposes of this thesis, SVM classification was performed using a tool known as SVM^{light} [62]. SVM^{light} is an implementation of the Support Vector Machine algorithm programmed in C. Created by Thorsten Joachims of Cornell University in 2008, this software was based on Vapnuk's Support Vector Machine to solve the problem of pattern recognition, regression, and learning a ranking function. The tool itself requires compilation from the source code, and consists of two compiled modules: "*svm_learn"* for learning from a training set, and "*svm_classify."* Svm_classify uses a model generated from the *svm_learn* module in order to classify a test dataset. These modules are used through the Terminal window (Linux/OSX) or the Command Prompt (Windows). Figure 5.8 shows a typical training/classification performed using the data from one of the scenarios.

```
Brents-MacBook-Pro:svm_light brent./svm_classify olsr4/training3.txt
olsr4/model2.txt
Scanning examples...done
Reading examples into memory...100..200..300..400..500..600..700..800..900..1000
..1100..1200..1300..1400..1500..1600..1700..1800..1900..2000..2100..2200..2300..
2400..2500..2600..OK. (2680 examples read)
Setting default regularization parameter C=0.2008
Optimizing.........done. (10 iterations)
Optimization finished (3 misclassified, maxdiff=0.00091).
Runtime in cpu-seconds: 0.01
Number of SV: 10 (including 8 at upper bound)
L1 loss: loss=5.00045
Norm of weight vector: |w|=0.99955
Norm of longest example vector: |x|=23.00000
Estimated VCdim of classifier: VCdim<=121.89034
Computing XiAlpha-estimates...done
Runtime for XiAlpha-estimates in cpu-seconds: 0.00
XiAlpha-estimate of the error: error<=0.37% (rho=1.00,depth=0)
XiAlpha-estimate of the recall: recall=>98.63% (rho=1.00,depth=0)
XiAlpha-estimate of the precision: precision=>98.63% (rho=1.00,depth=0)
Number of kernel evaluations: 26909
Writing model file...done


Brents-MacBook-Pro:svm_light brent$ ./svm_classify olsr4/14SC_FE25_Classify_14SC
.txt olsr4/model2.txt olsr4/sc_predict_14.txt
Reading model...OK. (10 support vectors read)
Classifying test examples..100..200..300..400..500..600..700..800..900..1000..11
00..1200..1300..1400..1500..1600..1700..1800..1900..2000..2100..2200..2300..2400
..2500..2600..done
Runtime (without IO) in cpu-seconds: 0.00
Accuracy on test set: 84.14% (2623 correct,  57 incorrect, 2680 total)
Precision/recall on test set: 46.13%/100.00%
```

Figure 5.10 – Performing SVM learning and Classification in the SVM^{light} tool.

The *svm_learn* module takes in a space delimited input file containing training examples. The format of each of the lines in the training example should look similar to:

<target> <feature>:<value> <feature>:<value> ... <feature>:<value> # <info>

Where <target> is a value of +1 for a positive example, -1 for a negative example, or 0 where the target will be treated as a negative example through transduction by default. A data entry can also have multiple features; for example, if characterizing a person, features could refer to traits such as: eye colour, hair colour, etc. For the purposes of this work, each data point (frame) has only one feature, since the only data that must be tested is the HD. Finally,

the value parameter is the actual digit to be checked against the SVM defined threshold, and in this example the HD was used as the value. It is also possible to force the tool to ignore a line, or provide additional information by commenting it out through the use of the # symbol. In order to correctly utilize SVM^{light}, the arrangement of the captured HD data had to be placed into the correct format (commonly known as "scaling"). An example of the formatting can be seen below:

-1 1:0 #Normal

+1 1:17 #Side-Channel

The above example shows two lines taken directly from the training file used for analysis. Selecting feature numbers is a fairly straightforward process, where each item (in this case, a frame) is granted a feature based on some criteria such as weighted importance. Given that the Hamming Distance value is the only metric being used for classification in this analysis, each of the frames had a single feature, which was given a value of 1.

For the SVM classification process a scenario with a moderate amount of Side-Channel was chosen, which for consistency, was the same experiment explored in Section 5.3. This experiment features 14% of the traffic as Side-Channel. In order to achieve the most optimal classification threshold, it is important to begin with a training set that demonstrates the cleanest separation between classes. As such, the training set was formed using a scenario with 14% Side-Channel, but this time a version with an FER of just 0.83% was used. Selecting a

scenario with such a low FER percentage ensures a significantly lower count of naturally corrupted frames, and offers clear separation between the HDs of the Side-Channel and Normal traffic. Training was accomplished using a feature set containing all 2680 frames from the scenario. Upon completion of the training process using the training data created based on the HD values for this experiment, SVM^{light} generated a model file for classification that contained a threshold value. For this particular experiment, SVM^{light} defined the threshold as 11.4559, which further supports the proposed threshold range of 11 – 12.

With training complete, classification using this model was performed on the 14%SC @ 25%FER scenario. The additional FER means more naturally corrupted frames appear, and a larger volume of Normal frames with high HD values offers a greater chance of misclassification. As shown in Figure 5.5, there are a large number of corrupted Normal frames with HDs in the 5 – 12 range.

SVM classifies items as positive or negative based on their value in relation to the defined threshold. Items that are classified as positive are assigned an SVM rank above zero, while negative items are assigned a value below zero. Figure 5.11 illustrates the predictions assigned by SVM for each of the frames, where the Y-axis represents the SVM prediction and the X-axis represents the Frame Number. Frames above 0 on the Y-axis have been classified as Side-Channel, while frames below 0 are classified as Normal traffic. For reference, the data points have been colour-coded based on their actual type. Frames with an SVM ranking above 0 that were accurately classified as Side-Channel frames have been categorized as “True

Positive," while Normal frames have been assigned a category of True Negative/False Positive depending on their SVM ranking. As the SVM rank increases or decreases, so does the likelihood that the value belongs to a certain class. For example, it is 100% likely that a frame with a ranking of -11 is a Normal frame, while a frame with a rank of +31 is most definitely a Side-Channel frame.

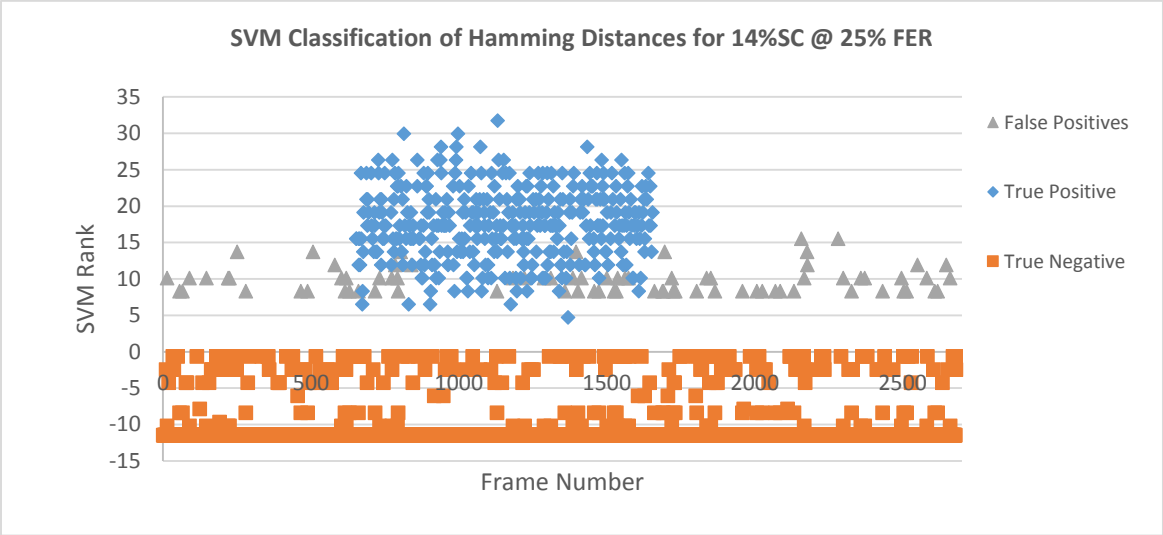

Figure 5.11 – SVM Classification of Side-Channel based on HDs of a scenario with 14% SC at 25% FER.

In Figure 5.12, the SVM classification has been graphed relative to the Hamming Distance of each frame. The calculated SVM threshold of 11 has also been included for reference. While SVM does incorrectly classify 57 data points as Side-Channel, it still manages to correctly classify 2623 of the total possible 2680 frames. The threshold shown through this SVM training (11.4559) is also conducive to the range of thresholds (11 – 12) defined using F-Score in Section 5.2. This signifies that SVM could potentially provide a system in which a suitable threshold for Side-Channel detection could be identified in a real-time approach.

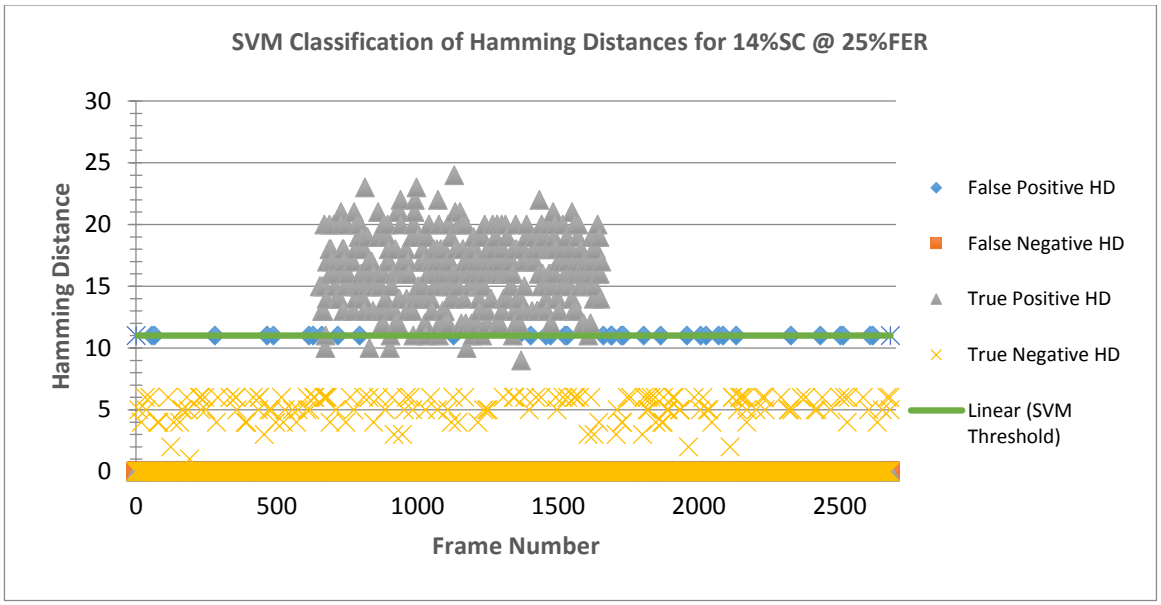

Figure 5.12 – SVM Classification in relation to HD.

Chapter 6 – Conclusion

6.1 – Summary

This thesis has presented the concept of Side-Channel communication through the modification of the CRC polynomials in a MANET environment, and attempted to provide the reader with a thorough understanding of its functionality and the issues involved in detection. In addition to this, several works related to this seemingly obscure problem have been explored, and their contributions and limitations have been assessed. Elements from these works were drawn from or considered when attempting to develop experiment scenarios that could accurately represent a real-world environment. The concepts of Cyclic Redundancy Checks and the HD metric were presented, along with the role that they play when trying to recognize the presence of Side-Channel.

The goal of this thesis was to take a military oriented MANETs environment that had fallen victim to Side-Channel communication and not only attempt to detect it, but also provide a range of threshold values that could work across a variety of situations. Information on Platooning was considered and 12 experiment scenarios came to life through the use of an 8-man platoon of nodes operating over the OLSR routing protocol in a MANET. Communication took place through the use of the Nping software, where Side-Channel traffic was directed through a different port than normal traffic. An agent node running a USB AirPCap packet capture card, in conjunction with Wireshark, was able to intercept messages on the network for

analysis. Through artificially introducing varying levels of FER into the captured data with the help of MATLAB, 264 unique experiment combinations were built and ready for analysis. With no simulators capable of offering the ability to output frame contents, including header information, all of the experiments had to be completed in a hardware environment with an emulated Side-Channel.

The F-Score mechanic was used to calculate a threshold based on HD for each of these scenarios, and it was identified that a mean threshold range of 11 – 12 provided accurate detection capabilities consistently across each of the experiments. Calculating the Standard Deviation and Confidence Interval of the thresholds for each of the varying amounts of Side-Channel made it clear that with 95% and 99% confidence, the range of Optimal HD Thresholds fell well within the defined range. In an attempt to illustrate the difference between Side-Channel and Normal traffic, a population sample was taken from the experiments and it was shown once again with Standard Deviation that Normal traffic typically had with 95% confidence a mean HD of 8.0 +/-0.32, and with 99% confidence a mean of 8.0+/-0.42. Comparatively, the HD of Side-Channel traffic was shown to hover in the area of 16.0 +/-0.54 (95%) & 16.0 +/- 0.70 (99%). Further validation of the F-Score thresholds was performed using ROC curves, which showed that the HD-based F-Score thresholds were incredibly accurate in this purpose.

To verify whether or not HD was a valid metric, and to ensure that F-Score was not biased, the Support Vector Machine algorithm was also considered, and was trained to

successfully classify Side-Channel based on HD as a feature. This evaluation was performed by taking an optimal dataset with little to no noise that featured 14% Side-Channel and training the SVM classifier. Once a model had been established, another scenario with 14% Side-Channel was selected for classification, with the key difference being the 25% FER present in the scenario. Not only did the results show that the HD metric could be used for classification using alternative approaches, it also presented a threshold that fits perfectly into the range proposed using F-Score.

While gathering the appropriate training information on a given network will help to accurately determine a threshold, the proposed range of 11 – 12 should provide an excellent starting point towards solving the issue of Side-Channel communication. Many of the techniques explored within this work have resulted in thresholds of similar values belonging to this range.

6.2 – Future Work

There are several areas that have been identified within this paper which could certainly benefit from additional research. Firstly, the experiments presented all share one similar characteristic in that the same CRC polynomials were used for Normal and Side-Channel frames. This makes the assumption that the Koopman 32-bit CRC would be chosen as the CRC polynomial for Side-Channel; however, there is no guarantee that the malicious party may utilize one of the standard CRC polynomials. Future work should be conducted through thorough experimentation on different standard and non-standard CRC polynomials. An

example of this would be to use a CRC with merely a single bit difference from the CRC used for Normal traffic and determine if our estimated threshold range would remain effective. In the work of Chea [3], it is suggested that there are a total of 2^{2342} CRC combinations, and an exhaustive comparison of the HD metric against even a small subset of these could prove beneficial.

Another key area that was not explored in this thesis, but would be an excellent topic to expand upon, would be the identification of a windowing approach for real-time detection. For the most part, the analysis within this thesis as well as in the previous works of Chea [3] or Madtha et al. [2] have been performed on a dataset of all frames captured during the entirety of an experiment's duration. When attempting to detect anomalies in real-time on an active network, this is of course not possible since as long as the network is up, there will never be a natural point where frame generation halts. As such, many detection methods employ the use of a mechanism known as *Windowing*. Windowing is a form of processing commonly used when analyzing digital signals, where a small subset is taken out of a larger dataset for processing and analysis. The challenge is, of course, knowing how large of a window size to take. A windowing approach with a scenario broken into five 60-second windows was conducted, but not expanded upon in-depth. The results of the F-Score calculations for this rudimentary windowing approach have been included in Appendix B.

Finally, further work into the validation of the F-Score mechanic could be performed by comparing the calculated threshold range against those calculated through the use of several

machine learning algorithms. Other possible solutions which were considered and could be explored include: Markov Chains, Bayesian modelling, and K-means. Given that there is only one feature (Hamming Distance) used for the detection method described in this work, approaching other Machine Learning methods would be superfluous. This means that additional features may be explored to further strengthen threshold predictions.

References

[1] "IEEE SA - 802.3-2012 - IEEE Standard for Ethernet," IEEE, 2015. [Online]. Available: http://standards.ieee.org/findstds/standard/802.3-2012.html. [Accessed 10 May 2015].

[2] N. Madtha, M. Vargas Martin, R. Liscano, B. Moore, M. Salmanian, M. Li and P. Mason, Detection of side-channel communication in ad hoc networks using request to send (RTS) messages, Toronto: IEEE 27th Canadian Conference on, 2014, pp. 1,6,4-7.

[3] V. Chea, "Hamming Distance as a Metric for the Detection of Side Channel in 802.11 Wireless Communications," Oshawa, 2015.

[4] G. C. Kessler, "An Overview of Steganography for," Gary Kessler Associates, June 2014. [Online]. Available: http://www.garykessler.net/library/fsc_stego.html. [Accessed 6 June 2015].

[5] M. Arnold, M. Schmucker and S. Wolthusen, Techniques and Applications of Digital Watermarking and Content Protection, Norwood, Massachusetts: Artech House, 2003.

[6] K. Szczypiorski, "HICCUPS: Hidden communication system for corrupted networks," *The Tenth International Multi-Conference on Advanced Computer Systems ACS'2003,* pp. 31-40, 2003.

[7] M. Fisk, C. Papadopoulos, J. Neil and G. Fisk, "Eliminating Steganography in Internet Traffic with Active Wardens," in *Lecture Notes in Computer Science: 2578*, 2002.

[8] B. Xu, J. Wang and D. Peng, "Practical Protocol Steganography: Hiding Data in IP Header,"

in *AMS '07. First Asia International Conference on Modelling & Simulation*, 2007.

[9] B. Jankowski, W. Mazurczyk and K. Szczypiorski, "Information Hiding Using Improper frame padding," *Telecommunications Network Strategy and Planning Symposium (NETWORKS),,* pp. 1,6,27-30, 2010.

[10] M. Odor, B. Nasri, M. Salmanian, P. Mason, M. Martin and R. Liscano, "A frame handler module for a side-channel in mobile ad hoc networks," *IEEE 34th Conference on Local Computer Networks (LCN),* pp. 20-23,930,936, October 2009.

[11] A. Najafizadeh, "Detection of Covert Communications based on Intentionally," University of Ontario Institute of Technology, Oshawa, 2011.

[12] J. Bacaj and L. Reznik, "POSTER: Signal anomaly based attack detection in wireless sensor networks," in *ACM SIGSAC Conference on Computer & Communications Security*, Berlin, 2013.

[13] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," *2010 IEEE Symposium on Security and Privacy (SP),* pp. 305,316,16-19, May 2010.

[14] Y. Song, A. Keromytis and S. Stolfo, "Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic," *16th Annual Network & Distributed System Security Symposium,* February 2009.

[15] W. Li, M. Duan and Y. Chen, "Network anomaly detection based on MRMHC-SVM algorithm," *Multitopic Conference, 2008. INMIC 2008. IEEE International,,* pp. 307,312,23-24, December 2008.

[16] X. Yong and Z. Yilai, "An inteligent anomaly analysis for intrusion detection based on SVM," *Computer Science and Information Processing (CSIP), 2012 International Conference on,* pp. 739,742,24-26, August 2012.

[17] D. Bolzoni, S. Etalle and P. Hartel, "Panacea: Automating attack classification for anomaly-based network intrusion detection systems," *Twelfth International Symposium on Recent Advances in Intrusion Detection, RAID 2009,* 2009.

[18] T. A. S. Foundation, "The GTUBE," 2015. [Online]. Available: http://spamassassin.apache.org/gtube/. [Accessed 18 June 2015].

[19] K. Dyer, S. Coull, T. Ristenpart and T. Shrimpton, "Protocol misidentification made easy with format-transforming encryption," *In Proceedings of the 2013 ACM SIGSAC conference on Computer & Communications Security (CCS '13),* pp. 61-72, 2013.

[20] SolarWinds, "Deep Packet Inspection & Analysis - Is It The Application Or The Network?," 2014. [Online]. Available: http://go.solarwinds.com/en/npm/deep-packet-inspection. [Accessed 18 June 2015].

[21] P. Engla and W. Lee, "Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques," *Proceedings of the 13th ACM Conference on Computer and Communications Security, ACM 2006,* 2006.

[22] Y. Muslukhov and H. B. K. Boshmaf, "An Approach to Address Physical Threats to Smartphones," *Symposium on Usable Privacy and Security (SOUPS),* 2012.

[23] M. Salem and S. Stolfo, "Modelling User Search Behavior for Masquerade Detection," *Proceedings of the Fourteenth Symposium on Recent Advances in Intrusion Detection (RAID),* 21 September 2011.

[24] Y. Aby-Mostafa, "Lecture 14 - Support Vector Machines," Caltech, 18 May 2012. [Online]. Available: https://www.youtube.com/watch?v=eHsErlPJWUU. [Accessed 10 December 2014].

[25] A. Lesk, "Introduction to Bioinformatics," [Online]. Available: https://books.google.ca/books?id=xYmcAQAAQBAJ&printsec=frontcover#v=onepage&q&f=false . [Accessed 13 December 2014].

[26] D. Meyer, "Support Vector Machines," [Online]. Available: http://cran.r-project.org/web/packages/e1071/vignettes/svmdoc.pdf. [Accessed 13 December 2014].

[27] A. Ben-Hur and J. Weston, "A user's guide to support vector machines," [Online]. Available: http://pyml.sourceforge.net/doc/howto.pdf. [Accessed 10 December 2014].

[28] C. U. Press, "Support Vector Machines: The Linearly Seperable Case," April 2009. [Online]. Available: http://nlp.stanford.edu/IR-book/html/htmledition/support-vector-machines-the-linearly-separable-case-1.html. [Accessed 9 December 2014].

[29] D. C. Group, "Sinalgo," [Online]. Available: http://disco.ethz.ch/projects/sinalgo/. [Accessed 18 June 2015].

[30] "QualNet," SCALABLE Network Technologies, Inc., [Online]. Available: http://web.scalable-networks.com/content/qualnet. [Accessed 7 June 2015].

[31] "The Internet Engineering Task Force," The Internet Engineering Task Force (IETF), [Online]. Available: https://www.ietf.org/. [Accessed 21 June 2015].

[32] ITU, "ITU: Committed to connecting the world," [Online]. Available: http://www.itu.int/en/Pages/default.aspx. [Accessed 21 June 2015].

[33] G. Cook, "Catalogue of parametrised CRC algorithms," 27 March 2015. [Online]. Available: http://reveng.sourceforge.net/crc-catalogue/all.htm. [Accessed 7 June 2015].

[34] P. Koopman, "32-bit cyclic redundancy codes for Internet applications," *International Conference on Dependable Systems and Networks,* pp. 459-468, 2002.

[35] P. Koopman and T. Chakravarty, "Cyclic redundancy checks via table look-up," *Communications of the ACM,* vol. 8, no. 31, pp. 1008-1013, 1988.

[36] R. Hamming, "Error detecting and error correcting codes," *Bell System Technical Journal,* vol. 29, pp. 147-160, 1950.

[37] K. Rosen, "Coding Theory," in *Applications of Discrete Mathematics*, New York, McGraw-Hill Higher Education, 2007, pp. 73-95.

[38] I. The MathWorks, "MATLAB," The MathWorks, Inc., 1994-2015. [Online]. Available: http://www.mathworks.com/products/matlab/. [Accessed 21 June 2015].

[39] M. Zanjireh, A. Shahrabi and H. Larijani, "ANCH: A New Clustering Algorithm for Wireless Sensor Networks," *Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on,* pp. 450,455, 25-28, March 2013.

[40] S. Rehman, M. Khan, T. Zia and L. Zheng, "Vehicular Ad-Hoc Networks (VANETs) - An Overview and Challenges," *Wireless Networking and Communications,* pp. 29-38, 2013.

[41] T. M. Corporation, "SmartPhone Ad-hoc Networking (SPAN)," [Online]. Available: http://www.mitre.org/research/technology-transfer/open-source-software/smartphone-ad-hoc-networking-span. [Accessed 18 June 2015].

[42] R. Sivakami and G. Nawaz, "Secured communication for MANETS in military," *Computer, Communication and Electrical Technology (ICCCET),* pp. 146,151,18-19, 18-19 March 2011.

[43] "OLSR.org Wiki," 2015. [Online]. Available: http://www.olsr.org/mediawiki/index.php/Main_Page. [Accessed 29 April 2015].

[44] N. W. Group, "RFC 3626 - Optimized Link State Routing Protocol (OLSR)," *Project Hipercom, INRIA,* 2003.

[45] S. Beitzel, *On Understanding and Classifying Web Queries,* 2006.

[46] A. Sangwan and J. Hansen, "Keyword recognition with phone confusion networks and phonological features based keyword threshold detection," *Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on,* pp. 711-715, 7-10 November 2010.

[47] P. Tao, Y. Huang, C. Wei, L. Ge and L. Xu, "A Method Based on Weighted F-Score and SVM for Feature Selection," *Control and Decision Conference (CCDC), 2013 25th Chinese,* pp. 4287-4290, 25-27 May 2013.

[48] C. VanRijsbergen, Information Retrieval, Butterworth, 1979.

[49] J. Xie, C. Wang, S. Jiang and Y. Zhang, "Feature selection method combing improved F-

score and support vector machine," *Journal of Computer Applications,* vol. 30, no. 4, 2010.

[50] "Harmonic Mean -- from Wolfram MathWorld," Wolfram Alpha, 2015. [Online]. Available: http://mathworld.wolfram.com/HarmonicMean.html. [Accessed 05 06 2015].

[51] " Cadet Portal AROTC at Michigan Tech," Michigan Tech, 2015. [Online]. Available: http://www.mtu.edu/arotc/cadet-portal/. [Accessed 24 February 2015].

[52] H. D. O. T. A. Army, "FM 7-8 Infantry Rifle and Platoon Squad Field Manual," Washington DC, 1992.

[53] "The Network Simulator - ns-2," [Online]. Available: http://nsnam.isi.edu/nsnam/index.php/User_Information. [Accessed 21 June 2015].

[54] A. Communications, "AR5002 Series Spec Sheet," [Online]. Available: http://mgvs.org/public/midge/datasheet/AR5002+spec+sheet.pdf. [Accessed 15 May 2015].

[55] "Wireshark," Wireshark Foundation, 2015. [Online]. Available: https://www.wireshark.org/. [Accessed 7 June 2015].

[56] C. Ltd., "Ubuntu 14.04.2 LTS (Trusty Tahr)," 2015. [Online]. Available: http://releases.ubuntu.com/14.04/. [Accessed 15 May 2015].

[57] G. Lyon, "Nmap.org," 2015. [Online]. Available: http://nmap.org/nping/. [Accessed 1 May 2015].

[58] R. Technology, "Riverbed AirPCap Adapter for Microsoft Windows," Riverbed Technology, 2015. [Online]. Available: http://www.riverbed.com/products/performance-management-control/network-performance-management/wireless-packet-capture.html. [Accessed 22 June 2015].

[59] "Plotting and Intrepretating an ROC Curve," University of Nebraska Medical Centre, [Online]. Available: http://gim.unmc.edu/dxtests/roc2.htm. [Accessed 23 June 2015].

[60] S. o. Surgery, "Medical Statistics VIII - Receiver operating characteristic (ROC) curves," 12 February 2015. [Online]. Available: https://www.youtube.com/watch?v=caqlX4cs4yQ. [Accessed 23 June 2015].

[61] Indon, "Wikimedia," 2007. [Online]. Available: https://commons.wikimedia.org/wiki/File:ROC_space.png. [Accessed 24 June 2015].

[62] T. Joachims, "SVM Light - Support Vector Machine," Cornell University, 14 August 2008. [Online]. Available: http://svmlight.joachims.org. [Accessed 6 June 2015].

[63] Z. Hong, Y. Shi and N. Ansari, "Hiding Data in Multimedia Streaming over Networks," *Communication Networks and Services Research Conference (CNSR), Eighth Annual,* pp. 50-55, May 2010.

[64] R. Cideciyan, "Frame Error Rate," IBM, 24-26 September 2012. [Online]. Available: http://www.ieee802.org/3/bj/public/sep12/cideciyan_3bj_01a_0912.pdf. [Accessed 11 June 2015].

Table of Appendices

Appendix A

Chea's [3] MATLAB Script for Post-Processing & HD Calculation

```
t=fix(clock);
year=num2str(t(1));
month=num2str(t(2));
day=num2str(t(3));
hour=num2str(t(4));
min=num2str(t(5));
sec=num2str(t(6));

execTime=strcat(year,month,day,hour,min,sec);
scenarioName='AdHoc5N16kB50MB';

subScenarioName=scenarioName(6:15);
subScenarioName2=scenarioName(1:7);


strH1='Frame#';
strH2='CRCPoly:0x04C11DB7(Default)';
strH3='CRCPoly:Recieved(FCS)';
%strH4='CRCPoly:0x04C11DB7(In_Error)';
%strH5='CRCPoly:0x741B8CD7(On_SC)';
strH6='Calculated_HD';
strH7='Protocol';
strH8='isGood';
crc32V1=[1 0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1];
crc32V2=[1 0 1 1 1 0 1 0 0 0 0 0 1 1 0 1 1 1 0 0 0 1 1 0 0 1 1 0 1 0 1 1 1];
crc32V3=[1];



%crcCalV1=comm.CRCGenerator(crc32V1,'InitialConditions',[1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1],'ReflectInputBytes',true,'ReflectChecksums',true,'FinalXOR',[1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]);
%crcCalV2=comm.CRCGenerator(crc32V2,'InitialConditions',[1 1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
1],'ReflectInputBytes',true,'ReflectChecksums',true,'FinalXOR',[1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]);
crcCalV1=comm.CRCGenerator(crc32V1);
crcCalV2=comm.CRCGenerator(crc32V2);
deModulation=comm.CRCGenerator(crc32V3);

%run through the fGood file to get the frame error
fFER=fopen('isFCSGood.dat','r');
fCount=1;
isGood = fgetl(fFER);
```

```
tally=0;
while ischar(isGood)
    if(isGood=='0')
        tally = tally + 1;
    end

    isGood = fgetl(fFER);
    fCount=fCount+1;

end
fclose(fFER);


actualCorr= (tally/fCount)*100
%FE
percentageCorr=[0 actualCorr 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80
85 90 95 100];


strFNames = strcat('HDresult_',scenarioName,'_wiAWGN','_',execTime);
strFMName=strcat('Metrics_',scenarioName,'_wiAWGN','_',execTime);
strFMAName=strcat('Stats_',scenarioName,'_wiAWGN','_',execTime);
strFSName=strcat('Summary_',scenarioName,'_wiAWGN','_',execTime);
strFNSample=strcat('TestTrainSample_',subScenarioName2); %stores frames as
frame# HD class(0-nonSC, 1-SC-ftp) starting at first ftp transmissoin

strFSNameOut=strcat(strFSName,'.txt');
strFNSampleOut=strcat(strFNSample,'.txt');
fSOut = fopen(strFSNameOut,'wt');
%fprintf(fSOut,'           0.5MB            2.5MB            5.0MB     \n');
%fprintf(fSOut,'FER% OptimalThreshold  OptimalThreshold OptimalThreshold\n');
fprintf(fSOut,'%10s\n',subScenarioName);

for corrVal = 1:22
    ferNum = num2str(percentageCorr(corrVal));
    strFNameOut = strcat(strFNames,'_FE',ferNum,'.txt');
    strFMNameOut=strcat(strFMName,'_FE',ferNum,'.txt');
    strFMANameOut=strcat(strFMAName,'_FE',ferNum,'.txt');


    falsePos = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
    trueNeg = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
    falseNeg = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
    truePos = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0];
    strFName=('adHocMatlabInput.dat');
    fInput = fopen(strFName,'r');
    fGood = fopen('isFCSGood.dat','r');
    fProto = fopen('protoFile.dat','r');
    fSINR = fopen('SINRFile.dat','r');
    fOut = fopen(strFNameOut,'wt');
    fprintf(fOut,'%-10s %-32s %-32s %-19s %-16s %-
6s\n',strH1,strH2,strH3,strH6,strH7,strH8);
    tline = fgetl(fInput);
```

```
    isFCSGood = fgetl(fGood);
    protoType = fgetl(fProto);
    SNRstr = fgetl(fSINR);
    cc=1;
    TotFrames=0;
    TotGFrames=0;
    TotCorFrames=0;
    TotFTP=0;

    isFTPLast=0; %0 not started yet, >0 save last seen frame number, start
saving to array
    sampleHDArray=[]; %after end of frames save to file from index 1 to
(sizeofArray-(cc-ftp last))
    sampleClassArray=[]; %0-nonSC-NORMAL, 1-SC-FTP
    sampleIndex=1;
    %otherCC=1;

    while ischar(tline)
        protoName = 'NORMAL';
        matchFTP=strcmp('1337',protoType);
        matchTCP=strcmp('80',protoType);
        if(isFCSGood=='1' && (matchTCP==1 || matchFTP==1)) %when isGoodFile 1
then good(True), 0 is bad(False)
            TotFrames= TotFrames+1; %track total frames

            ascVer=uint8(tline);
            asTemp=ascVer-48;
            X=double(asTemp);
            m=X';

            G1=step(crcCalV1,m); %calculate the default CRC on all frames
            fcsPartV1=(G1(end-31:end))'; %get the good FCS part

            if(matchTCP==1)
                protoName='NORMAL';
            elseif(matchFTP==1)
                isFTPLast=sampleIndex; %we have seen a ftp save the frame
number and start saving to array
                protoName ='FTP';
                G1=step(crcCalV2,m); %calculate the koopman CRC as SC
            end
            SNRValue = str2num(SNRstr);
            r=rand;
            if(r>=0 && r<(percentageCorr(corrVal)/100))
                TotCorFrames=TotCorFrames+1; %track all corrupted frames

                %B1=bsc(G1,0.05); %send the G1 through the BSC
                crcMod=awgn(G1,SNRValue);
                B1=step(deModulation,crcMod); %send the G1 through the AWGN
                %disp('B1')
            elseif (r>(percentageCorr(corrVal)/100) && r<=1)
                TotGFrames=TotGFrames+1; %track all good frames

                B1=G1;
```

```
            end

            fcsPartB1=(B1(end-31:end))';

            %do some Hamming calculation for each FCS value
            hamV1vsB1 = binArray2HammingD(fcsPartV1,fcsPartB1);
            %hamV1vsV2 = binArray2HammingD(fcsPartV1,fcsPartV2);
            %hamV1vsV3 = binArray2HammingD(fcsPartV1,fcsPartV3);

            fprintf(fOut,'%-10d',cc); %'Frame#'
            fprintf(fOut,' ');
            fprintf(fOut,'%-d',fcsPartV1); %'CRCPoly:0x04C11DB7(d)'
            fprintf(fOut,' ');
            fprintf(fOut,'%-d',fcsPartB1); %'CRCPoly:Recieved(FCS)'
            fprintf(fOut,' ');
            fprintf(fOut,'%-19d',hamV1vsB1); %'Calculated HD'
            fprintf(fOut,' ');
            fprintf(fOut,'%-16s',protoName); %'Protocol'
            fprintf(fOut,' ');

            if(hamV1vsB1==0)
                fprintf(fOut,'Good'); %'isGood'
                fprintf(fOut,'\n');
                %otherCC=otherCC-1; %must take one from the total because its
not saving in out array
            else
                if(isFTPLast>0) %start saving frames HD to array we only want
bad
                    sampleHDArray(sampleIndex)=hamV1vsB1;
                    if(matchTCP==1)
                        sampleClassArray(sampleIndex)=0;
                    elseif(matchFTP==1)
                        sampleClassArray(sampleIndex)=1;
                    end
                    sampleIndex=sampleIndex+1;
                end
                fprintf(fOut,'Bad'); %'isGood'
                fprintf(fOut,'\n');
            end

            if(matchTCP==1)
                %protoName='NORMAL';
                for threshold = 1:30

                    if(hamV1vsB1<=threshold)%True-: Correctly identified Non
SC frame as Non SC NORMALHD<=Threshold
                        tempVal = trueNeg(threshold);
                        trueNeg(threshold) = tempVal + 1;
                    elseif(hamV1vsB1>threshold)%False+:Incorrectly Identified
non SC frame as SC NORMALHD>Threshold
                        tempVal = falsePos(threshold);
                        falsePos(threshold) = tempVal + 1;
                    end
                end
```

```
            elseif(matchFTP==1)
                TotFTP=TotFTP+1; %track FTP frames
                %protoName ='FTP';
                for threshold = 1:30
                    if(hamV1vsB1>threshold)%True+:Correctly identified SC
frame as SC FTPHD>Threshold
                        tempVal = truePos(threshold);
                        truePos(threshold) = tempVal + 1;
                    elseif(hamV1vsB1<=threshold)%False-:Incorrectly
identified Non SC frame as SC FTPHD<=Threshold
                        tempVal = falseNeg(threshold);
                        falseNeg(threshold) = tempVal + 1;
                    end
                end
            end
        end
        tline = fgetl(fInput);
        isFCSGood = fgetl(fGood);
        protoType = fgetl(fProto);
        SNRstr = fgetl(fSINR);
        matchFTP=0;
        matchTCP=0;
        cc=cc+1;
        %otherCC=otherCC+1;
    end
    fclose(fProto);
    fclose(fGood);
    fclose(fOut);
    fclose(fInput);
    fclose(fSINR);

    strMH1='Threshold';
    strMH2='True(+)';
    strMH3='False(+)';
    strMH4='True(-)';
    strMH5='False(-)';
    strMH6='Accuracy';
    strMH7='Sensitivity';
    strMH8='Specificity';
    strMH9='Precision';
    strMH10='F-Score';

    fMOut = fopen(strFMNameOut,'wt');
    fprintf(fMOut,'%-9s %-7s %-8s %-7s %-8s %-8s %-11s %-11s %-9s %-
7s(FER=%3s%%)\n',strMH1,strMH2,strMH3,strMH4,strMH5,strMH6,strMH7,strMH8,strM
H9,strMH10,ferNum);

    bThresh = 0;
    bFScore = 0;

    for x = 1:30

        %TP=truePos(x)/TotFTP;
        %TN=trueNeg(x)/(TotGFrames+TotCorFrames);
```

```
        %FP=falsePos(x)/(TotGFrames+TotCorFrames);
        %FN=falseNeg(x)/TotFTP;

        TP=truePos(x);
        TN=trueNeg(x);
        FP=falsePos(x);
        FN=falseNeg(x);


        %Precision = Positive Predictive Value (PPV) = TP/TP+FP [missing from
your equations]
        %Recall = Sensitivity = True Positive Rate (TPR) = TP/TP+FN
        %Specificity = True Negative Rate (TNR) = TN/TN+FP


        %accuracy = ((TP+TN)/(TP+TN+FP+FN));
        %sensitivity = (TP/(TP+FN));
        %specificity = (TN/(TN+FP));
        %Precision = (TP/(TP+FP));

        accuracy = (TP+TN)/(TP+TN+FP+FN);
        sensitivity = TP/(TP+FN);
        specificity = TN/(TN+FP);
        precision = TP/(TP+FP);
        fScore = (2*precision*sensitivity)/(precision+sensitivity);


        if(fScore > bFScore)
           bThresh=x;
           bFScore=fScore;
        end
        fprintf(fMOut,'%-9d',x);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-7d',TP);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-8d',FP);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-7d',TN);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-8d',FN);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-8.4f',accuracy);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-11.4f',sensitivity);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-11.4f',specificity);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-9.4f',precision);
        fprintf(fMOut,' ');
        fprintf(fMOut,'%-15.4f',fScore);
        fprintf(fMOut,'\n');
    end
    fclose(fMOut);
```

```
    fMAOut = fopen(strFMANameOut,'wt');
    if (corrVal==1)
        fprintf(fMAOut,'Percentage of Actual Corrupted packets in Capture: %-
6.4f \n',actualCorr);
    end
    fprintf(fMAOut,'Percentage of Simulated Corrupted packets: %-6.4f
\n',percentageCorr(corrVal));
    fprintf(fMAOut,'Total Frames: %d \n',TotFrames);
    fprintf(fMAOut,'Total Good Frames: %d \n',TotGFrames);
    fprintf(fMAOut,'Total Corrupted Frames: %d \n',TotCorFrames);
    fprintf(fMAOut,'Total FTP Frames (Side-Channel): %d \n',TotFTP);
    if(bThresh>0)
        fprintf(fMAOut,'Best Threshold: %d',bThresh);
        fprintf(fMAOut,'Best Threshold: %1.4f',bFScore);
        fprintf(fSOut, '%10d\n',bThresh);
    else
       fprintf(fMAOut,'No Threshold found \n');
       fprintf(fSOut, '%10d\n',0);
    end

    fclose(fMAOut);
    fSampleOut = fopen(strFNSampleOut,'at');
    %isFTPStop=sampleIndex-(otherCC-isFTPStart);
    ind=1;
    while ind<(isFTPLast+1)
        if((isFTPLast-ind)>0)
            fprintf(fSampleOut,'%d %d\n',
sampleHDArray(ind),sampleClassArray(ind));
        else
            fprintf(fSampleOut,'%d %d\n',
sampleHDArray(ind),sampleClassArray(ind));
        end
        ind=ind+1;
    end

    fclose(fSampleOut);
    strFMANameOut

end
fclose(fSOut);
```

Appendix B

F-Score Calculations for Window Scenario 1

Threshold	True(+)	False(+)	True (-)	False (-)	Accuracy	Sensitivity	Specificity	Precision	F-Score (FER=25%)
1	0	136	399	0	0.7458	0.0000	0.7458	0.0000	0.0000
2	0	135	400	0	0.7477	0.0000	0.7477	0.0000	0.0000
3	0	134	401	0	0.7495	0.0000	0.7495	0.0000	0.0000
4	0	123	412	0	0.7701	0.0000	0.7701	0.0000	0.0000
5	0	111	424	0	0.7925	0.0000	0.7925	0.0000	0.0000
6	0	96	439	0	0.8206	0.0000	0.8206	0.0000	0.0000
7	0	77	458	0	0.8561	0.0000	0.8561	0.0000	0.0000
8	0	54	481	0	0.8991	0.0000	0.8991	0.0000	0.0000
9	0	31	504	0	0.9421	0.0000	0.9421	0.0000	0.0000
10	0	12	523	0	0.9776	0.0000	0.9776	0.0000	0.0000
11	0	7	528	0	0.9869	0.0000	0.9869	0.0000	0.0000
12	0	2	533	0	0.996262	0.0000	0.9963	0.0000	0.0000
13	0	2	533	0	0.9963	0.0000	0.9963	0.0000	0.0000
14	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
15	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
16	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
17	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
18	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
19	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
20	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
21	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
22	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
23	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
24	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
25	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
26	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
27	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
28	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
29	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000
30	0	0	535	0	1.0000	0.0000	1.0000	0.0000	0.0000

F-Score Calculations for Window Scenario 2

Threshold	True(+)	False(+)	True (-)	False (-)	Accuracy	Sensitivity	Specificity	Precision	F-Score (FER=25%)
1	150	101	285	0	0.8116	1.0000	0.7383	0.5976	0.7481
2	150	101	285	0	0.8116	1.0000	0.7383	0.5976	0.7481
3	150	99	287	0	0.8153	1.0000	0.7435	0.6024	0.7519
4	150	95	291	0	0.8228	1.0000	0.7539	0.6122	0.7595
5	150	84	302	0	0.8433	1.0000	0.7824	0.6410	0.7813
6	150	69	317	0	0.8713	1.0000	0.8212	0.6849	0.8130
7	150	55	331	0	0.8974	1.0000	0.8575	0.7317	0.8451
8	150	35	351	0	0.9347	1.0000	0.9093	0.8108	0.8955
9	150	25	361	0	0.9534	1.0000	0.9352	0.8571	0.9231
10	147	14	372	3	0.9683	0.9800	0.9637	0.9130	0.9453
11	143	9	377	7	0.9701	0.9533	0.9767	0.9408	0.9470
12	137	4	382	13	0.9683	0.9133	0.9896	0.9716	0.9416
13	124	1	385	26	0.9496	0.8267	0.9974	0.9920	0.9018
14	110	0	386	40	0.9254	0.7333	1.0000	1.0000	0.8462
15	87	0	386	63	0.8825	0.5800	1.0000	1.0000	0.7342
16	63	0	386	87	0.8377	0.4200	1.0000	1.0000	0.5915
17	44	0	386	106	0.8022	0.2933	1.0000	1.0000	0.4536
18	32	0	386	118	0.7799	0.2133	1.0000	1.0000	0.3516
19	22	0	386	128	0.7612	0.1467	1.0000	1.0000	0.2558
20	10	0	386	140	0.7388	0.0667	1.0000	1.0000	0.1250
21	4	0	386	146	0.7276	0.0267	1.0000	1.0000	0.0519
22	2	0	386	148	0.7239	0.0133	1.0000	1.0000	0.0263
23	0	0	386	150	0.7201	0.0000	1.0000	0.0000	0.0000
24	0	0	386	150	0.7201	0.0000	1.0000	0.0000	0.0000
25	0	0	386	150	0.7201	0.0000	1.0000	0.0000	0.0000
26	0	0	386	150	0.7201	0.0000	1.0000	0.0000	0.0000
27	0	0	386	150	0.7201	0.0000	1.0000	0.0000	0.0000
28	0	0	386	150	0.7201	0.0000	1.0000	0.0000	0.0000
29	0	0	386	150	0.7201	0.0000	1.0000	0.0000	0.0000
30	0	0	386	150	0.7201	0.0000	1.0000	0.0000	0.0000

F-Score Calculations for Window Scenario 3

Threshold	True(+)	False(+)	True (-)	False (-)	Accuracy	Sensitivity	Specificity	Precision	F-Score (FER=25%)
1	196	85	255	0	0.8414	1.0000	0.7500	0.6975	0.8218
2	196	85	255	0	0.8414	1.0000	0.7500	0.6975	0.8218
3	196	84	256	0	0.8433	1.0000	0.7529	0.7000	0.8235
4	196	81	259	0	0.8489	1.0000	0.7618	0.7076	0.8288
5	196	75	265	0	0.8601	1.0000	0.7794	0.7232	0.8394
6	196	61	279	0	0.8862	1.0000	0.8206	0.7626	0.8653
7	196	44	296	0	0.9179	1.0000	0.8706	0.8167	0.8991
8	196	29	311	0	0.9459	1.0000	0.9147	0.8711	0.9311
9	195	21	319	1	0.9590	0.9949	0.9382	0.9028	0.9466
10	194	18	322	2	0.9627	0.9898	0.9471	0.9151	0.9510
11	190	11	329	6	0.9683	0.9694	0.9676	0.9453	0.9572
12	173	1	339	23	0.9552	0.8827	0.9971	0.9943	0.9351
13	159	1	339	37	0.9291	0.8112	0.9971	0.9938	0.8933
14	145	0	340	51	0.9049	0.7398	1.0000	1.0000	0.8504
15	123	0	340	73	0.8638	0.6276	1.0000	1.0000	0.7712
16	89	0	340	107	0.8004	0.4541	1.0000	1.0000	0.6246
17	64	0	340	132	0.7537	0.3265	1.0000	1.0000	0.4923
18	39	0	340	157	0.7071	0.1990	1.0000	1.0000	0.3319
19	25	0	340	171	0.6810	0.1276	1.0000	1.0000	0.2262
20	7	0	340	189	0.6474	0.0357	1.0000	1.0000	0.0690
21	3	0	340	193	0.6399	0.0153	1.0000	1.0000	0.0302
22	1	0	340	195	0.6362	0.0051	1.0000	1.0000	0.0102
23	1	0	340	195	0.6362	0.0051	1.0000	1.0000	0.0102
24	0	0	340	196	0.6343	0.0000	1.0000	0.0000	0.0000
25	0	0	340	196	0.6343	0.0000	1.0000	0.0000	0.0000
26	0	0	340	196	0.6343	0.0000	1.0000	0.0000	0.0000
27	0	0	340	196	0.6343	0.0000	1.0000	0.0000	0.0000
28	0	0	340	196	0.6343	0.0000	1.0000	0.0000	0.0000
29	0	0	340	196	0.6343	0.0000	1.0000	0.0000	0.0000
30	0	0	340	196	0.6343	0.0000	1.0000	0.0000	0.0000

F-Score Calculations for Window Scenario 4

Threshold	True(+)	False(+)	True(-)	False(-)	Accuracy	Sensitivity	Specificity	Precision	F-Score (FER=25%)
1	18	142	377	0	0.7356	1.0000	0.7264	0.1125	0.2022
2	18	140	379	0	0.7393	1.0000	0.7303	0.1139	0.2045
3	18	137	382	0	0.7449	1.0000	0.7360	0.1161	0.2081
4	18	130	389	0	0.7579	1.0000	0.7495	0.1216	0.2169
5	18	117	402	0	0.7821	1.0000	0.7746	0.1333	0.2353
6	18	103	416	0	0.8082	1.0000	0.8015	0.1488	0.2590
7	18	80	439	0	0.8510	1.0000	0.8459	0.1837	0.3103
8	18	54	465	0	0.8994	1.0000	0.8960	0.2500	0.4000
9	18	35	484	0	0.9348	1.0000	0.9326	0.3396	0.5070
10	18	17	502	0	0.9683	1.0000	0.9672	0.5143	0.6792
11	17	4	515	1	0.9907	0.9444	0.9923	0.8095	0.8718
12	16	1	518	2	0.9944	0.8889	0.9981	0.9412	0.9143
13	16	1	518	2	0.9944	0.8889	0.9981	0.9412	0.9143
14	14	0	519	4	0.9926	0.7778	1.0000	1.0000	0.8750
15	12	0	519	6	0.9888	0.6667	1.0000	1.0000	0.8000
16	9	0	519	9	0.9832	0.5000	1.0000	1.0000	0.6667
17	6	0	519	12	0.9777	0.3333	1.0000	1.0000	0.5000
18	4	0	519	14	0.9739	0.2222	1.0000	1.0000	0.3636
19	1	0	519	17	0.9683	0.0556	1.0000	1.0000	0.1053
20	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
21	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
22	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
23	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
24	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
25	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
26	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
27	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
28	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
29	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000
30	0	0	519	18	0.9665	0.0000	1.0000	0.0000	0.0000

F-Score Calculations for Window Scenario 5

Threshold	True(+)	False(+)	True(-)	False(-)	Accuracy	Sensitivity	Specificity	Precision	F-Score(FER=25%)
1	0	128	410	0	0.7621	0.0000	0.7621	0.0000	0.0000
2	0	128	410	0	0.7621	0.0000	0.7621	0.0000	0.0000
3	0	128	410	0	0.7621	0.0000	0.7621	0.0000	0.0000
4	0	125	413	0	0.7677	0.0000	0.7677	0.0000	0.0000
5	0	113	425	0	0.7900	0.0000	0.7900	0.0000	0.0000
6	0	97	441	0	0.8197	0.0000	0.8197	0.0000	0.0000
7	0	84	454	0	0.8439	0.0000	0.8439	0.0000	0.0000
8	0	56	482	0	0.8959	0.0000	0.8959	0.0000	0.0000
9	0	38	500	0	0.9294	0.0000	0.9294	0.0000	0.0000
10	0	19	519	0	0.9647	0.0000	0.9647	0.0000	0.0000
11	0	13	525	0	0.9758	0.0000	0.9758	0.0000	0.0000
12	0	6	532	0	0.9888	0.0000	0.9888	0.0000	0.0000
13	0	3	535	0	0.9944	0.0000	0.9944	0.0000	0.0000
14	0	2	536	0	0.9963	0.0000	0.9963	0.0000	0.0000
15	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
16	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
17	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
18	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
19	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
20	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
21	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
22	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
23	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
24	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
25	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
26	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
27	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
28	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
29	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000
30	0	0	538	0	1.0000	0.0000	1.0000	0.0000	0.0000

Appendix C

SVM Model Output File

SVM-light Version V6.02
0 # kernel type
3 # kernel parameter -d
1 # kernel parameter -g
1 # kernel parameter -s
1 # kernel parameter -r
empty# kernel parameter -u
1 # highest feature index
2680 # number of training documents
11 # number of support vectors plus 1
11.4559208 # threshold b, each following line is a SV (starting with alpha*y)
-0.200847865236406714073424950584O1 1:9 #Normal
0.200847865236406714073424950584O1 1:11 #Side-Channel
-0.200847865236406714073424950584O1 1:10 #Normal
0.200847865236406714073424950584O1 1:9 #Side-Channel
-0.200847865236406714073424950584O1 1:9 #Normal
0.200847865236406714073424950584O1 1:11 #Side-Channel
-0.200847865236406714073424950584O1 1:9 #Normal
0.200847865236406714073424950584O1 1:12 #Side-Channel
-0.198501578879586304537951946258S4 1:9 #Normal
0.198501578879358930862508714199O7 1:12 #Side-Channel