COVERT CHANNELS IN SECURE WIRELESS NETWORKS

VOIES CLANDESTINES DANS LES RESÉAUX SANS FIL SÉCURISÉS

A Thesis Submitted

to the Division of Graduate Studies of the Royal Military College of Canada

by

Paul Edwin Charles Martin, CD
Major

In Partial Fulfillment of the Requirements for the Degree of

Masters of Applied Science in Computer Engineering

April, 2007

Your file Votre référence
ISBN: 978-0-494-29822-0
Our file Notre référence
ISBN: 978-0-494-29822-0

For Janique, Gabriel and Nicolas. In addition, I'd like to dedicate this work to all the Canadian trees who gave their lives in the production of this document.

Acknowledgements

I would like to thank my supervisor, Dr. G. S. Knight, for assisting me in finding my muse to complete this thesis. His focused guidance, limit-less patience and kind encouragement allowed me to improve my academic self-confidence and enjoy the Master's experience. I was extremely lucky to have him as my supervisor.

I would also like to thank Dr. G. Phillips, Maj. R. Smith and the rest of the CSL crew for their open and frank comments that greatly assisted in the shaping and presentation of this work.

Abstract

Martin, Paul Edwin Charles. M.A.Sc. Royal Military College of Canada, April, 2007. *Covert Channels in Secure Wireless Networks*. Supervised by Dr. G. Scott Knight.

Covert channels are unexpected and hidden communication paths embedded within a communication system that violate the system security policy. Covert communication occurs when a user or application deliberately manipulates and embeds information into some property of a communication system in such a way that the embedded information is not apparent to the legitimate users of the communication system. Internet based covert channels with low bit rates are enough to convey critical information such as network encryption keys or system access codes. This paper discusses how the threat of covert channels can be applied to secure wireless networks through the design and implementation of a working covert communication system. The resulting communication system reliably communicates private information in the presence of noise. In this case noise is other legitimate network traffic using the same encrypted link. The covert channel implementation presented here has been tested on a live network with background noise traffic and demonstrates the feasibility and limitations of the design.

Keywords: covert, channel, wireless, network.

Resumé

Martin, Paul Edwin Charles. M.A.Sc. Collège militaire royal du Canada, Avril, 2007. *Voies clandestines dans les reséaux sans fil sécurisés.* Thèse dirigée par M. G. Scott Knight, Ph.D.

Les voies clandestines sont des chemins de communications inattendus et cachés, incorporés dans un système de communication, et qui violent la politique de sécurité du système. La communication clandestine se produit quand un utilisateur ou une application manipule et incorpore délibérément de l'information dans une propriété du système de communication d'une manière telle que l'information dissimulée soit inapparente aux utilisateurs légitimes du système de communication. Les voies clandestines à faibles débits binaires, basées sur Internet, suffisent pour véhiculer de l'information critique telle que des clefs de chiffrement de réseau ou des codes d'accès du système. Cet article discute comment la menace des voies clandestines peut être appliquée aux réseaux sans fil sécurisés au moyen de la conception et l'exécution d'un système de communication clandestin fonctionnel. Le système de communication résultant communique l'information privée en présence d'un bruit. Dans ce cas-ci, un bruit constitue un trafic de réseau légitime sur le même lien chiffré. L'implémentation de la voie clandestine présentée ici a été testée sur un réseau opérationnel doté d'un trafic de bruit de fond et démontre la praticabilité ainsi que les limitations de sa conception.

Mots clés : voies, clandestines, sans fil, sécurisés

Table of Contents

List of Tables

List of Figures

List of Acronyms

802.11	Wireless LAN standards developed by the IEEE LAN/MAN Standards Committee
BSS	Basic Service Set
CCMP	Counter Mode with CBC-MAC Protocol
CNI	Computer Network Intelligence
DARPA	Defence Advanced Research Projects Agency
FCS	Frame Check Sequence
FEC	Forward Error Correction
IEEE	Institute of Electrical and Electronics Engineers
ISO	International Standards Organization
IP	Internet Protocol
LAN	Local Area Network
LLC	Logical Link Control
MAC	Medium Access Control
MTU	Maximum Transmission Unit
OSI	Open Source Interconnect
PHY	Physical layer component
PRNG	Pseudorandom Number Generator
SNAP	Sub-Network Access Protocol
SNR	Signal to Noise Ratio
TCP	Transmission Control Protocol
TKIP	Temporal Key Integrity Protocol
UDP	User Datagram Protocol
USB	Universal Serial Bus
WEP	Wired Equivalent Privacy
Wi-Fi	Wireless Local Area Network based on IEEE 802.11 specifications
WPA	Wi-Fi Protected Access

Chapter 1

Introduction

1.1 Thesis Summary

This thesis discusses a series of research activities that investigated how the threat of covert channels can be applied to secure wireless networks through the design and implementation of a working covert communication system. The resulting covert communication system was comprised of two distinct and separate components namely a software Trojan and a covert receiver. The software trojan employed aspects of coding theory [20, 31, 35] to modulate 802.11 wireless traffic with encoded private information from a compromised workstation. The modulated wireless traffic was then captured by a covert receiver passively monitoring the wireless network. The captured information was demodulated and corrected using Forward Error Correction (FEC) techniques [20, 35] to ensure communication reliability within an inherently noisy environment such as wireless networking. The resulting communication system reliably communicated private information in the presence of other legitimate network traffic over an encrypted link.

1.2 Introduction

The increased demand for easily configurable, low cost mobile computing solutions has fuelled the recent popularity of wireless networking products based on the IEEE 802.11 standard. 802.11 based technologies offer network solutions that are highly mobile and rapidly deployable [45, 12, 2]. This type of capability is of significant interest to organizations with the desire to extend corporate resources to employees operating in field conditions, especially in situations where conventional wired network infrastructures do not exist such as developing communities, disaster areas, or war damaged regions.

Information protection in closed private wired networks has been traditionally accomplished in part through the physical separation of the attacker and the targeted media. With the introduction of 802.11 based wireless networks, private networks can now be exposed to potential attackers anywhere within reception range of the wireless transmission. To combat eavesdropping, the 802.11 protocol utilizes secure wireless standards such as Wired Equivalent Privacy (WEP) and Wi-Fi Protected Access (WPA). Although much attention has been focused on the hardening of the secure wireless standards to prevent information interception and exploitation through eavesdropping, the fact still remains that the communication traffic on a wireless network link can be passively monitored by a third party.

1.3 Motivation

Covert channels (section 2.5) are unexpected and hidden communication paths embedded within a communication system that violate the system security policy. By manipulating the property of a communication system, a covert channel can convey information to designated recipients while remaining very difficult to detect. Even covert channels with low bit rates are enough to convey critical information such as access codes or security passphrases. Research into viable covert channels typically centres around the exploitation and detection of communication systems employed for nefarious purposes [23, 32, 47] by intentionally modifying data within legitimate network protocol communication (ex. changes to TCP or IP header fields [32]) to convey information in "plain sight". Despite the potential for these techniques to effectively work over open networks, many of these approaches can be controlled in closed private networks with the application of firewall and intrusion detection technology.

The introduction of wireless network links into a closed private network environment expands the potential use of covert channels to computing assets no longer physically located in the same location as the wired infrastructure. Since network traffic on a wireless network link can be passively monitored by a third party, wireless network infrastructures open new and extremely attractive attack vectors from which hostile agencies can covertly glean sensitive network information through covert channels. Many network-based covert channel techniques deal with information hidden in the upper layers of the TCP/IP protocol suite which are more easily concealed from eavesdropping through the application of protocol encryption such as WPA.

The majority of previous covert channel research has been limited to wired infrastructure environments [32, 44] with some work in the area of wireless networks [23, 47]. Despite the previous work conducted in this field, covert channel research rarely addresses issues of error detection and correction resulting from the coexistence of covert communication within regular network traffic. In an attempt to remain concealed from detection, wireless network based covert channels must operate amongst the noise of regularly expected network traffic. In addition, if a wireless covert communication system is to be a reliable conduit of information, it must also be able to accurately and consistently detect and correct the covert signal from within the concealment of other similar network traffic. To accomplish this challenge, an error detection and correction system must be employed in which the covert communication signal is extracted from the background environmental noise and corrected if errors are present.

Furthermore, the majority of attention with respect to secure wireless networking has been focused on developing stronger ciphers, thus hardening the secure wireless standards in an attempt to prevent information interception and exploitation through eavesdropping. Alternatively, little attention has been given to possible threats to secure wireless networks that originate from within the infrastructure. The security posture of an entire wireless network could be compromised if private information such as security passwords or access codes were divulged to an attacker from within the secure wireless network infrastructure.

1.4 Research Hypothesis

Combining the threat of covert channels with the popularity of 802.11 secure wireless networks, the hypothesis for this research is as follows.

The threat of low bit rate covert channels can be applied to secure wireless networks in order to compromise network security by passing private information such as security passphrases or system access codes.

In order to validate this hypothesis and demonstrate the vulnerability of wireless networks to covert channels, the research will focus on the production of a functioning covert communication system. The resulting communication system will be required to pass private network information from a compromised wireless computer to a covert receiver despite the implementation of a secure wireless standard such as WPA.

1.5 The Working Scenario

The *working scenario* for this thesis considers an organization such as a national military, non-governmental organization or perhaps a group which operates in an environment in which wired network infrastructures are very limited or do not exist such as in developing communities, disaster areas, war damaged regions. For reasons of expediency, maintenance, cost, connectivity and survivability the organization in question has deployed an 802.11 wireless network infrastructure for it's mobile felid agents working outside the confines of the main office complex. The local population regularly organize open air markets in which technology items such as CD/DVDs, MP3 players and memory sticks are often sold for amazingly low prices. One of the field agents from the organization, Alice, can not resist getting a good bargain and purchases a memory stick and immediately begins using it with the company wireless computer.

For the purposes of this research, we assume that the memory stick contains a software Trojan which transfers itself to the company computer upon first contact. The software Trojan then collects private network information from the compromised wireless computer such as the WPA encryption key. The collected key is then placed in a covert communication channel which beacons the key information to a foreign intelligence collection agent eavesdropping on local wireless network traffic as depicted in figure 1.1. Finding the signature of the covert channel in the captured network traffic transmitted by the compromised computer, the intelligence agent is able to reproduce the wireless network security password and gain access to the secure network. Once the intelligence agent has secretly joined the network and overcome the link en-

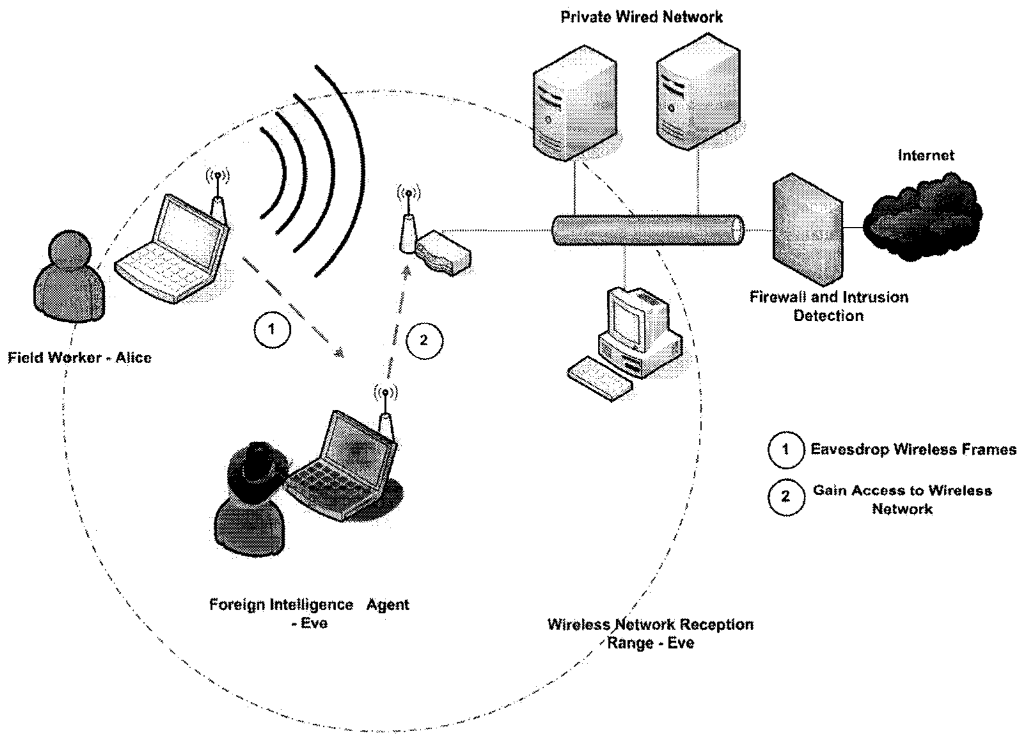

Figure 1.1: Covert Channels in Wireless Networks - Working Scenario.

cryption, any and all private information residing on the organization's network is available for subversion.

1.6 Research Objective

The principle goal of this research was to design, implement and test a working covert communication system on a secure wireless Internet based system. In order to achieve this goal, research activities were organized into three basic phases: 1) Developing an effective coding/decoding scheme, 2) Designing a covert communication system, and 3) Implementation of the Design.

In the first phase, a coding/decoding scheme was developed based on Forward Error Correction (FEC) techniques and validated against network traffic typical of a wireless Internet based system. Subsequently, in the second phase, the successful coding/decoding scheme was then incorporated into the design of a covert communication system which was designed as two distinct and separate components namely a software Trojan and covert receiver. Finally in the third phase, the resulting two components of the covert communication system were implemented using Microsoft Visual C++ and MATLAB respectively. The software Trojan was able to beacon modulated Internet Protocol (IP) traffic with encoded private information from the compromised wireless workstation. The modulated wireless traffic was then captured by a covert receiver passively monitoring the wireless network. The captured information was then demodulated and corrected to reliably reproduce the private network information.

The ability to reliably reproduce private network information despite the presence of protocol encryption validates the research hypothesis and exposes a security hole in contemporary wireless computer networks. In addition to validating the research hypothesis and achieving the overall research objective, the resulting communication system also produced useful performance data in terms of covert communication reliability and bandwidth in the presence of noise.

1.7 Thesis Organization

In order to appropriately articulate the successful validation of the research hypothesis, this thesis is arranged into four subsequent chapters. Chapter 2 will first provide an overview of the background material employed to investigate the problem space defined by the threat of covert channels in secure wireless networks. Chapter 3 will then present the theory employed to solve different aspects of the research problem. Subsequently, Chapter 4 will take the new ideas and approaches presented in Chapter 3 and discuss how they were applied in several experiments. Furthermore in Chapter 4, experimentation results are discussed and validated in terms of the design criteria as well as demonstrating the success of the wireless covert channel capability. Finally, the resulting covert communication system will be further discussed in Chapter 5 in terms of design performance and significance with respect to the research problem. In addition, Chapter 5 will provide some thoughts into recommendations for future work as well as concluding remarks to complete the thesis.

Chapter 2

Literature Survey

2.1 Introduction

Chapter 1 provided a introduction to the research topic area of this thesis by briefly outlining the motivation and research hypothesis that led to the principle goal of designing, implementing and testing a working covert communication system on a secure wireless network. This chapter will further expand on the ideas and technologies that served as a foundation for the *working scenario* and influenced the research into covert channels in secure wireless networks.

The background discussion in this chapter will cover several aspects of this research including the motivation for the *working scenario*, the TCP/IP and 802.11 technology models, covert channels as well as aspects of coding theory. The unique combination and application of these ideas and technologies will be further refined in Chapter 3 as part of the research toward designing a successful *working scenario*.

2.2 Fidelity of the Working Scenario

Intelligence and information collection is an activity no longer limited to the intelligence services of foreign states but has also expanded to sovereign individuals who operate in cells or on an individual basis for financial gain, glory or ideological reasons [25]. Due to the very profitable exchange of private information, former Russian and Chinese hacking experts are acting as as mercenaries selling their skills on the black market to political and religious extremists for cash [25].

The activity of gleaning information from open or secret sources through the use of computers or computer networks also known as Computer network intelligence (CNI) [18], provides foreign states, criminals or terrorists with highly reliable insider information that greatly assists in the achievement of their respective goals. Since the dispersal of al Qaeda elements from Afghanistan in 2001, terrorist groups have turned to the use and exploitation of IP based networks to continue their activities [5]. Not only are terrorist groups becoming more Internet and networking technology savvy for the purposes of planning and training, they are also educating and grooming followers in the art of network hacking for the purposes of conducting "electronic jihad" [5, 15]. Given the relative low costs associated with modern computing resources, the asymmetric threat of CNI posed by highly trained, mobile and determined groups is real and significant [25].

An important aspect in the conduct of CNI operations is the requirement to remain covert [18]. Consequently, CNI activities must be concealed amongst the regular activates of the operating environment, and the location of the information recipient must remain undetectable [18]. In addition, the act of transmitting captured data must be done without creating unnatural traffic patterns and therefore must be done slowly over longer periods of time [18]. The *working scenario* incorporates these approaches by suggesting that a compromised wireless computer slowly signals private information to a silent eavesdropping intelligence collection agent who is able to extract the covert message from the noise of regular network traffic.

The corrupted wireless computer in the *scenario* gathers information through the use of a software Trojan working on the behalf of the intelligence collection agent. The assumption that the software Trojan is implanted on a third-party computer through social engineering techniques is not unrealistic based on evidence [14] that this type of activity is already occurring on a regular basis. The article [14] reported that UK authorities uncovered evidence of foreign intelligence service agents compromising UK business and government agencies with hacking tools implanted by unsuspecting employees through the use of nefarious e-mails and USB memory sticks. This report has also highlighted that the information targeted by nefarious groups is not limited to government state secrets, but also included items that were scientific and economic in nature. Therefore, the threat of CNI operations is no longer limited to state government agencies but also extends to any organization with private information that serves the purposes foreign states, criminals or terrorists if compromised. Consequently, secure wireless networks provide nefarious individuals with a highly accessible means to target any group or organization with desirable information while at the same time remain hidden from detection.

2.3 TCP/IP Protocol Suite

The 7-layer Open Source Interconnect (OSI) architecture model developed by the International Standards Organization (ISO) in 1984 represents the foundation by which Internet based systems are designed [45]. The OSI model (figure 2.1) employs a hierarchical layered approach to computer communications in which each layer provides defined communication services to the layer directly above. In addition, each layer in the architecture is capable of employing the services from the layer directly below. For example, a particular layer of one computer can exchange information with the corresponding layer on another computer by utilizing the sub-layer communication services of the OSI framework.

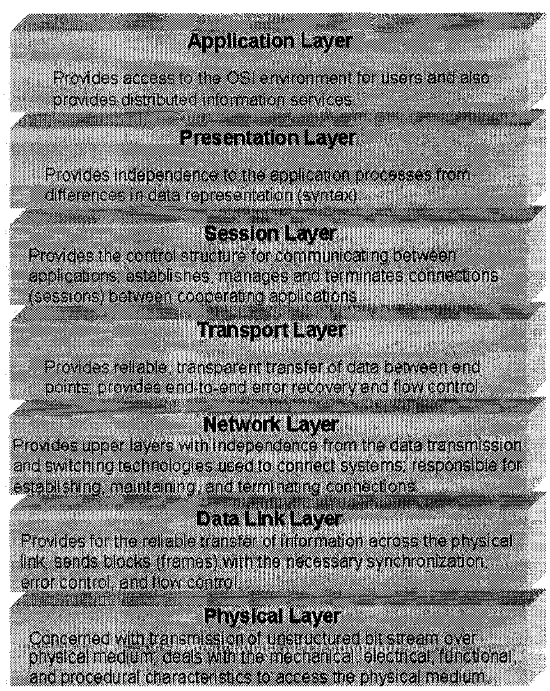

Figure 2.1: OSI Model from [45]

Although the 7-layer OSI model is often promoted as the standard architecture

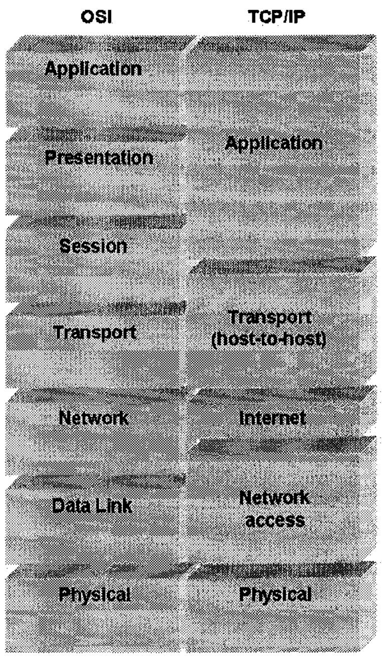

Figure 2.2: OSI and TCP/IP model comparison from [45]

for computer communication, the vast majority of Internet based systems employ the more simplistic 5-layer TCP/IP protocol architecture (figure 2.2) developed by US DoD Defence Advanced Research Projects Agency (DARPA) in 1969 [45]. Subsequently, the collection of protocols based on this less complex architecture is generally referred to as the TCP/IP protocol suite and represents the communication standards for the Internet and all Internet based systems. The following sub-sections are brief descriptions of the communication tasks contained in each of the five layers of the TCP/IP suite.

2.3.1 Physical Layer

The Physical Layer of the TCP/IP protocol suite deals with the physical interface between the data transmission device (e.g. network card, modem) and a transmission medium or network. Typical information employed at this layer of the suite includes transmission medium characteristics, signal metrics and data rates.

2.3.2 Network Access Layer

The Network Access Layer focuses on the exchange of information between the computer system and the connected network. The software employed at the Network Access Layer is specific to the connected network and thus may change depending on the specific standards associated with the connected network such as packet switching frame relay, Ethernet LAN or others. Regardless of the employed software variation at this layer of the protocol suite, the services provided to the next layer in the hierarchy (Internet Layer) are identical. This allows software functioning at higher levels of the suite to execute properly despite the unique characteristics of the attached network.

2.3.3 Internet Layer

The responsibility of the Internet Layer is to provide end-to-end delivery of data packets to a corresponding Internet Layer on another computer system. In order to provide this service, the Internet Layer employs the Internet Protocol (IP) to define logical network addressing and provides information routing to a remote host over multiple networks.

2.3.4 Transport Layer

The Transport Layer of the TCP/IP suite is responsible for providing reliable host-to-host transfer of Application Layer data. The most commonly used Transport Layer protocol within the context of the TCP/IP suite is the Transport Control Protocol (TCP). TCP is a connection-oriented protocol that provides a reliable means for the passage of application data through logical association. To establish a logical connection between two hosts, the protocol executes a three-way handshaking procedure [46, 42]. If the handshaking procedure is successful, a host-to-host connection is established. With a host-to-host connection established, the TCP Protocol then employs sequence numbering, acknowledgement numbering as well as window fields to provide flow and error control to reliably transfer data.

In addition to TCP, another commonly employed Transport Layer protocol in the TCP/IP suite is the User Datagram Protocol (UDP). In contrast with TCP, UDP is a connection-less protocol that does not guarantee delivery, protect against duplication or preserve sequence numbers for reordering upon receipt [45]. Since UDP is connection-less, data can be sent between hosts in a "fire and forget" manor with a minimum of protocol mechanisms.

2.3.5 Application Layer

The Application Layer is the top layer of the TCP/IP protocol suite which contains application specific logic to support the communication needs of user applications [45]. By calling to services in the Transport Layer, individual applications in the application layer can initiate tailored network communication activities in order to achieve the design goals of a particular application. In the context of the thesis *working scenario*, the covert communication channel established by the Software Trojan will originate from this Layer.

2.4 802.11 Wireless Networks

The subject area of 802.11 wireless networks covers a large range of topics from Radio Frequency (RF) fundamentals, spread spectrum technology, antenna principles to network infrastructure and protocols. In order to achieve an appreciation for the secure wireless network component of the thesis *working scenario*, the following broad overview to 802.11 wireless networks and its integration with the network communications architecture is presented.

802.11 wireless network standards were developed by the Institute of Electrical and Electronics Engineers (IEEE) as part of the 802 family series of specifications for local area network (LAN) technologies that focus on the lower two layers of the OSI model [12]. The individual network specifications in the 802 family series are identified by the decimal subcategory. For example, 802.3 is the specification for a Carrier Sense Multiple Access network with Collision Detection (CSMA/CD) or

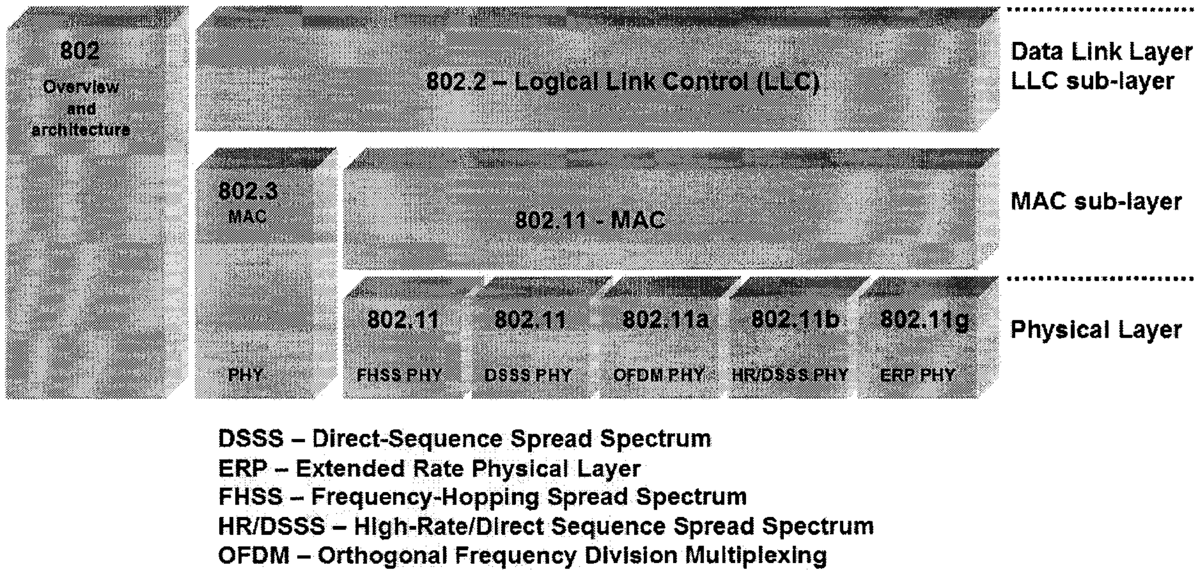

Figure 2.3: IEEE 802.11 and its relation to the OSI model from [12]

more commonly referred to as Ethernet while 802.2 specifies a common link layer and Logical Link Control (LLC). In the case of wireless networks, 802.11 standards define the Medium Access Control (MAC) and Physical (PHY) component specifications for wireless connectivity of fixed, portable and moving stations within a local area [22]. Similar to Ethernet, the 802.11 MAC sub-layer interacts with the LLC sub-layer of 802.2 to form the complete Data Link Layer of the OSI model or a portion of the Network Access Layer of the TCP/IP model. The relationships between 802.3, 802.2 and 802.11 and their respective place in the OSI model can be seen in figure 2.3.

2.4.1 802.11 Medium Access Control

The IEEE 802.11 MAC layer is responsible for servicing higher level protocols to reliably deliver data and apply optional security methods. Similar to other 802 link layers, 802.11 can support any network layer protocol communications such as IP. To accomplish the transport and delivery of higher-level protocols, the network layer packet must be encapsulated into an 802.11 data frame. 802.11 relies on the 802.2 LLC to achieve this encapsulation on its behalf. Employing the sub-network access protocol (SNAP), a 802.2 LLC frame is produced with the original un-encrypted IP

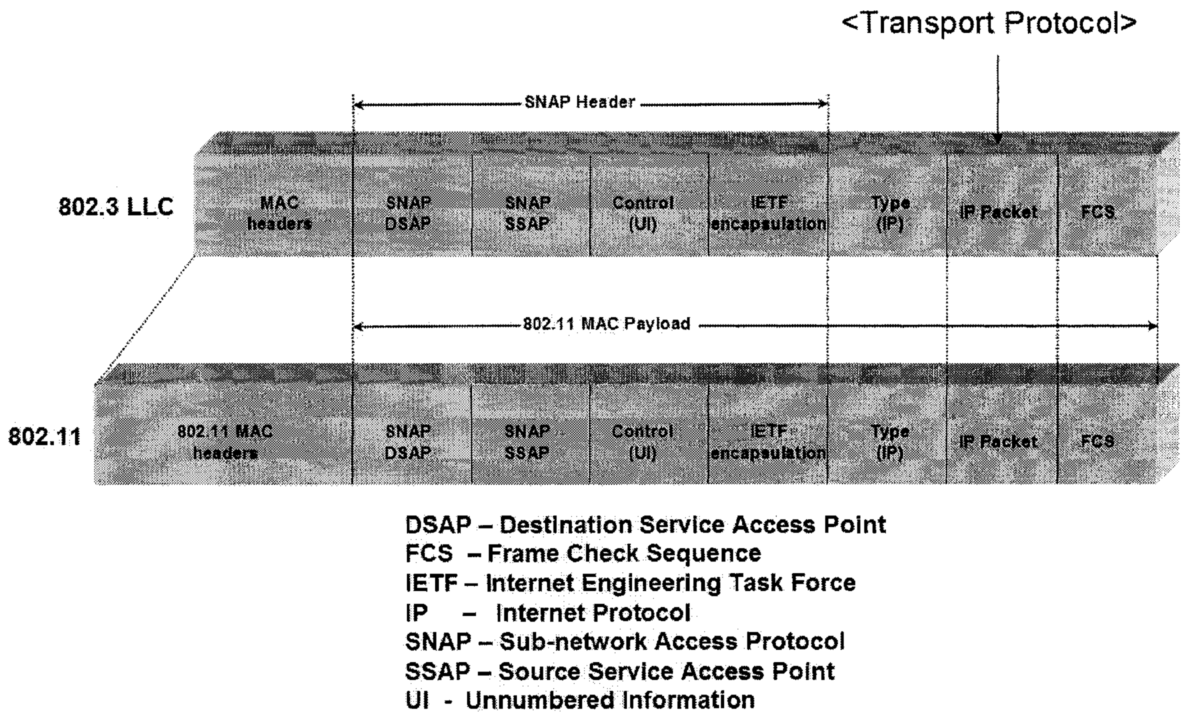

Figure 2.4: Encapsulation of Higher layer Protocols within 802.11 from [12]

packet as its payload. The resulting frame is then passed to the 802.11 MAC sublayer where it is then formatted into a 802.11 MAC data frame and then passed to the PHY for transmission. A graphical representation of this relationship is shown in figure 2.4.

If an 802.11 security method is employed, the 802.2 LLC frame is encrypted before it forms the payload portion of the 802.11 MAC frame. This process in turn obfuscates all information (packet headers and payload) originating from the Network Layer into an encrypted frame body. 802.11 supports two authentication and encryption methods, namely Wired Equivalent Privacy (WEP) and Wi-Fi Protected Access (WPA). WEP was the original authentication and encryption method supported by the 802.11 standard. WEP uses a pseudorandom number generator (PRNG) along with an RC4 stream cipher to encrypt data [2]. Due to significant weaknesses in the implementation of WEP, Temporal Key Integrity Protocol (TKIP) was implemented as a temporary fix to WEP. TKIP would attempt to address the weaknesses in the

WEP RC4 encryption implementation through the use of per-packet key mixing and automatic re-keying [2]. In an attempt to develop a longer term solution to wireless security, IEEE developed the 802.11i standard (2004) which included the WPA authentication and encryption method. As well as supporting WEP-TKIP standards, WPA also supports AES encryption which is considered more secure than previous encryption methods for wireless networks. The WPA version of the AES cipher is implemented with the Counter Mode with CBC-MAC Protocol (CCMP) as the mechanism to encrypt the 802.11 frames [2]. AES encryption is considered virtually unbreakable with current computing resources [2]. Similarly, WPA with AES is currently accepted as the most secure security standard for secure wireless networks and thus will be the focus of this research. In terms of the research goal, the proposed covert communication system must operate despite the implementation of WPA with AES encryption.

The 802.11 wireless MAC frame, regardless of type (Data, control, Management, etc.) is always composed of a MAC header, optional frame body and Frame Check Sequence (FCS) field. The sizes of each field within the MAC data frame shown in figure 2.5 are represented in brackets and reported in octets/bytes. Although the maximum size of a MAC data payload field is 2304 bytes, the MAC frames observed as part of this research will most likely never reach their maximum length of 2346 bytes. This is due to the fact that IP based networks employ a TCP/IP stack that usually implements an IP maximum transmission unit (MTU) of 1500 bytes [2]. Since the *working scenario* will employ the TCP/IP suite, it is unlikely that a packet larger than 1500 bytes will be passed to the MAC Layer. The wireless MAC data frame is of particular interest to the research of wireless covert channels. More specifically, research activities will focus on the resulting changes in frame attributes due to actions taken by an automated process operating from the Application Layer of the TCP/IP suite.

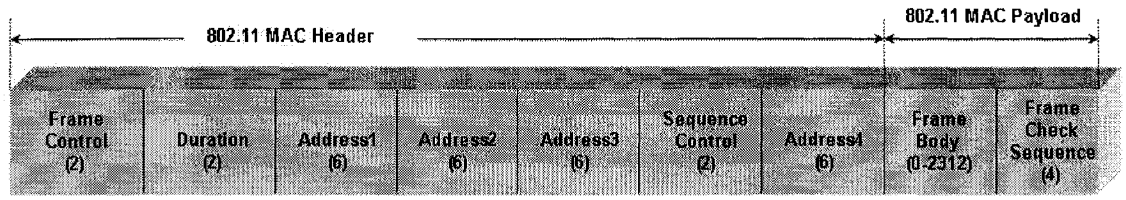

Figure 2.5: 802.11 Data Frame from [22]

2.4.2 802.11 Physical Layer

The Physical Layer of the 802.11 specification deals with the RF transmission and reception aspects of the wireless communication. Each variant of the 802.11 specification is primarily characterized by the changes in the physical layers. The 802.11 specification currently includes five different physical layer variations within the specification. Initially, the base 802.11 specification included two physical layers consisting of frequency hopping spread-spectrum (FHSS) and direct-sequence spread-spectrum (DSSS). Later revisions of the 802.11 specification added a high-rate direct-sequence (HR/DSSS) physical layer commonly referred to as 802.11b. Additionally, 802.11a and 802.11g are characterized by their respective orthogonal frequency multiplexing (OFDM) and extended rate PHY (ERP) physical layers. Although the 802.11 physical layer variations form an integral part of the 802.11 specification, aspects and details of the 802.11 physical layer are outside the scope of this thesis. The research activities of this thesis relating to 802.11 wireless networks will focus primarily on the MAC sub-layer and it's interaction with the higher layers of the TCP/IP suite.

2.4.3 802.11 Nomenclature

The four physical components that form the majority of all 802.11 wireless networks are *stations*, *access points*, a *wireless medium*, and a *distribution system* as seen in figure 2.6. *Stations* are the computing hosts which communicate over a network. In the context of wireless networking, *stations* are typically highly mobile battery operated laptops or hand-held computers. Although, if the infrastructure of a wired network does not exist or is impractical due to the working environment, *stations* can also be regular desktop computers with wireless network interface cards. In terms of the *working scenario*, the *stations* are represented by field agents who work in an environment in which wired infrastructure is inappropriate due to the hostile working environment.

Access points are the network devices that act as bridging tools between the wired and wireless network infrastructure. *Access points* not only perform wired-to-wireless network communication conversion, but also act as a wireless network gatekeeper by establishing security options such as wireless access security passphrases and MAC address list filtering. *Access points* also define the type of encryption, if any, that the wireless network will employ. If a type of link encryption is selected, the security passphrase is used to generate encryption keys particular to the selected encryption method.

The *wireless medium* represents the free space between stations and access points in which 802.11 MAC frames are passed. In order to communicate over the *wireless medium*, the 802.11 protocol employs one of the RF techniques defined in the PHY standard to signal information to a corresponding PHY of another wireless network device. Typical wireless devices can communicate over the *wireless medium* at distances ranging from 100m to 300m [45].

The *distribution system* represents the wired "backbone" network to which the *access point* is connected. Typically, the backbone network is a wired Ethernet configuration [12] that connects wireless users to the Internet (via an access point) and potentially an organization's private resources.

2.4.4 802.11 Architecture

When the physical components of an 802.11 wireless network are combined together as a communicating group, they form the basic building block of a wireless network commonly referred to as a Basic Service Set (BSS). Each BSS can be further characterized into one of two types of wireless networks, namely *independent networks* and *infrastructure networks*. *Independent wireless networks* are formed when stations

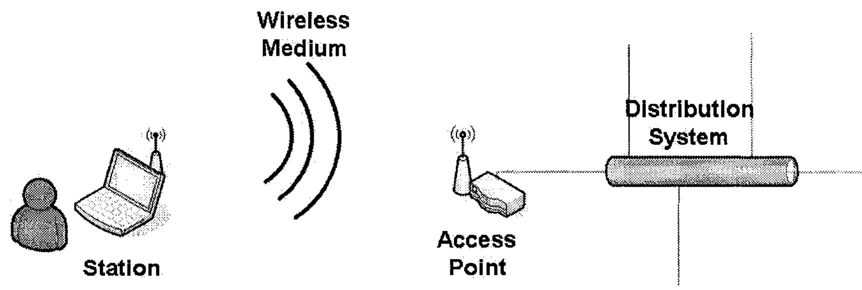

Figure 2.6: Components of 802.11 LANs from [12]

communicate directly with each other without the assistance of an access point. Typically, *independent networks* are employed when users of stations wish to create a simple network to pass information for a short duration such as in a meeting or class room setting. These types of networks are commonly referred to as ad hoc BBSs or ad hoc networks [12]. See figure 2.7.

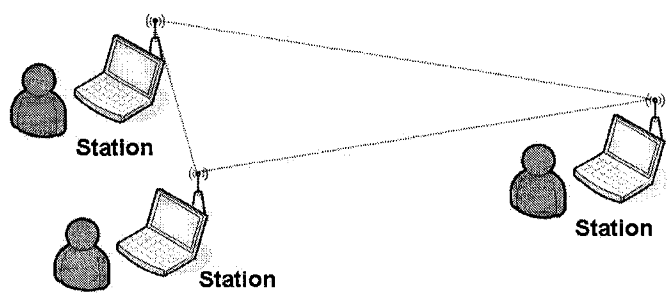

Figure 2.7: 802.11 Ad hoc Wireless Network from [12]

On the other hand, *infrastructure networks* are BBSes that employ an access point to assist in the communication between stations. More specifically, any communication between two stations in *infrastructure networks* must first go to an access point and then be relayed to the second station as shown in figure 2.8. Although a multi-hop transmission appears to be less efficient in terms of delay and channel capacity, an *infrastructure network* can increase the communication range between stations by acting as a radio relay effectively doubling the range between stations.

Furthermore, in an *infrastructure network*, a station must request an access point's permission to join the network in order to obtain network services. For a station to

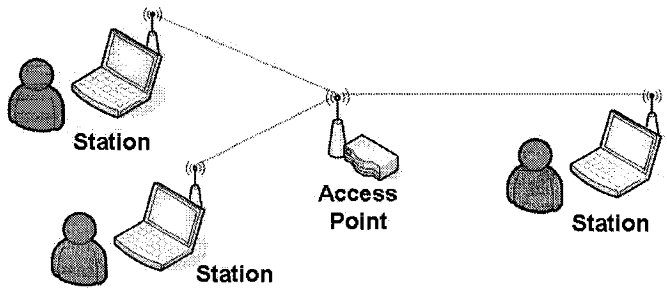

Figure 2.8: 802.11 Infrastructure Network from [12]

join a wireless network, it must first associate itself with the access point. The association process in 802.11 wireless networks establishes the MAC address of the requesting station in order to send and receive frames on a wireless LAN. Once the station has been associated with the access point, it then must be authenticated to establish that the requesting station possesses the authority to connect to the network. If an *infrastructure network* is employing open authentication, a station can join the network without providing any proof of authority. Although, in the case of secure wireless networks, the authority for a station to join a wireless network is proven through the use of a pre-established security password. Once the station has proven its authority to join the network with the security password, the wireless link is encrypted with one of the 802.11 security methods. For the purposes of this research, the *working scenario* utilizes an *infrastructure network* with the strongest security method available in the 802.11 standard WPA with AES encryption.

2.4.5 Wireless Eavesdropping

Wireless eavesdropping is the covert process by which a wireless computer can capture 802.11 frames from other wireless stations or BBSs that are within the reception range of the employed antenna. The attraction with eavesdropping from an attacker's point of view is that it leaves no trace of the attacker's presence since an eavesdropping computer does not need to join or be on the targeted network to capture its MAC frames [2]. The tools to perform an eavesdropping wireless LAN attack are widely available [51] and relatively simple to configure. In addition, with the use of a high gain directional antenna, an attacker can extend the reception range of an attack far beyond the typical ranges associated with commercial omni-directional antennas [51]. This makes the threat of an eavesdropping attack even more significant since the attacker need not be in close proximity but instead could be positioned several kilometres away from the targeted wireless network. Once again, this only reiterates the significance of the *working scenario* and the exposure of wireless network traffic to potentially malicious individuals.

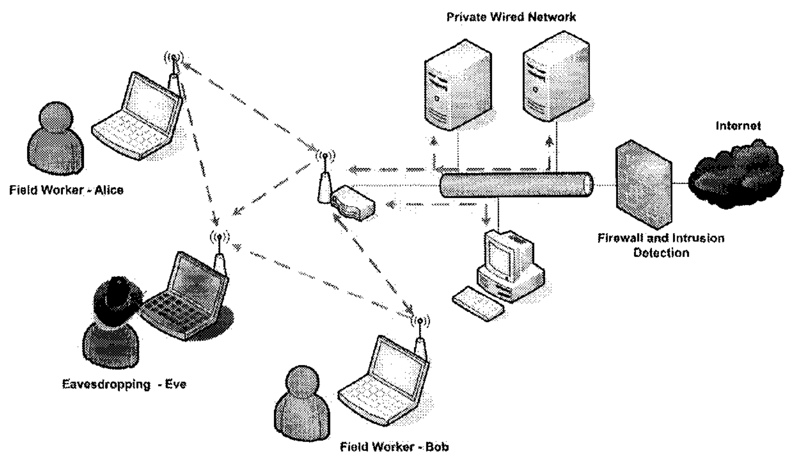

Figure 2.9: Wireless Eavesdropping Scenario

2.5 Covert Channels

In order to design and implement a working covert communication system on a secure wireless network and fulfil the principle goal of this research, a basic definition and understanding of covert channel implementations is required. This section will provide a working definition of covert channels and how it relates to the *working scenario* of this thesis.

2.5.1 Definitions

The concept of covert channels was first introduced by Butler W. Lampson in 1973 as part of an investigation into program execution confinement within an operating environment. As part of his research, Lampson noted that one potential category of data leakage from within an confined program could be classified as a Covert Channel if the leakage "was not intended for information transfer at all, such as the service program's effect on the system load" [19]. Since the time of Lampson's research which first noted the existence of covert channels, several definitions for covert channels have been used to further describe the hidden or indirect passage of information within a computing environment, such as the following.

> Covert Channels are the "... indirect (covert) transmission by storage into variables that describe resource status." [38]

> "Covert Channels, in contrast, use entities not normally viewed as data objects to transfer information from one subject to another." [16]

Although these definitions of covert channels are correct, they fail to explicitly articulate the malicious intent to violate the systems security policy as part of the transfer of information [13]. One definition which attempts to encompass the malicious intent of the covert channel users is as follows.

> "Given a nondiscretionary (e.g., mandatory) security policy model M and its interpretation I(M) in an operating system, any potential communication between two subjects I(S_h) and I(S_i) of I(M) is covert if and only if any communication between the corresponding subjects S_h and S_i of the model M is illegal in M." [48].

Although more comprehensive, the definition used at [48] is somewhat abstract. In an attempt to employ a similar but easier to convey characterization for the purposes of this research, the following definition was considered:

> "Covert channels are a means of communication between two process that are not permitted to communicate, but do so anyway, a few bits at a time, by affecting shared resources." [28]

Merging the simplicity of the definition from [28] with the network-based environment discussed by the authors at [23], the definition employed in this thesis when discussing covert channels is as follows.

> Covert channels are unexpected and hidden communication paths embedded within a communication system that violate the system security policy. By manipulating the property of a communication system, a covert channel can convey information to designated recipients while remaining very difficult to detect by the legitimate users of the system.

Although covert channels use information hiding to pass data, they should not be confused with the very similar art of steganography. Steganographic channels also entail the hiding of information within an environment in such a way that only the intended recipients are aware of its existence, such as digital watermarking [6, 29, 31, 34]. Nevertheless, the one distinguishing characteristic of covert channels is the malicious intent on the part of the parties employing the channel to violate a system security policy [13, 33] and communicate the security measures prohibiting the activity. If any steganographic techniques are employed with the intent of violating a security policy, this would be considered covert channels communications. On the other hand, a steganographic channel occurs when the two communicating parties employing a steganographic technique are allowed to talk and communicate within the confines of the system's established security policy [28]. This is not to say that nefarious characters do not employ steganographic channels. It has been reported at [15] that terrorists have used steganographic channels to plan terrorist attacks by hiding instructions within digital photographs on the popular auction web site e-Bay. Despite any similarities, steganographic channels are not the focus of this research.

Since covert channels are usually very hard to detect and not the product of a computer environment's original design, some may still doubt their ability to exist in a real-world context. Given the wealth of research and examples of covert channels at [30, 39, 41, 43], there should be no uncertainty that covert channels do exist and represent a significant security threat to organizations, business and governments.

The following sub-sections will further define the different types covert channels and the environments in which they exist.

2.5.2 Covert Channel Classifications

2.5.2.1 Storage and Timing Channels

Covert channels are not limited to a particular method or approach but in fact exist in many forms. Since any communication path that violates a security policy can be considered a covert channel, there exist a number of communication conduits that can be classified as a covert channels despite any potential differences in the approach of a particular exploit. Although, most covert channel exploits can be classified into two separate scenarios that either involve storage or timing [13] to achieve a successful implementation.

Covert storage channels transfer data between sender and recipient through the use of storage variables [13]. The sender alters the value of a storage variable and the recipient, observing the storage variable, detects and interprets the changes [16]. These variables may be covertly passed directly or indirectly between the sender and receiver as long as they employ a shared resource. In addition, the storage variables may contain overt information such as a message within a particular field [7] or represent information for the purposes of "signalling" a message across the shared resource [16].

Covert timing channels, by contrast, transfer data between sender and recipient through the use of an observable system property and a common time reference [13]. The sender modulates the observable property with respect to the common time refence with information from a covert message. The receiver then interprets the delay or lack of delay with respect to the common time reference as the covert message [39].

2.5.2.2 Noisy and Noiseless Channels

Covert channels can be further sub-classified as noisy or noiseless channels depending on the environment in which the exploit exists. A channel is considered noiseless if the data transmitted by the sender is received by the receiver with a probability of 1 [13]. In this case, the sender and receiver have exclusive use of the shared resource and do not experience disruptions in communication that would require error detection and correction. The noiseless channel can be considered an error-free environment.

On the other hand, a channel is considered noisy if the data transmitted by the sender is received by the receiver with a probability of less than 1 [13]. Furthermore, it is possible that the receiver might also contend with additional data that did not originate from the sender. This additional data represents a type of noise that interferes with the covert message within the channel. In this case, the sender and receiver share the common resource with other processes which produce similar data. Consequently, the sender and receiver in this channel must incorporate some form of error detection and correction capability [8, 21, 52] in order to effectively recover the covert message from the noise of the shared resource.

2.5.3 Network-Based Covert Channels

Network-based covert channels have been known to exist within several different Layers of the TCP/IP suite [43, 32, 47]. A number of them employ steganographic techniques which piggy-back on protocols to pass information fields [4]. To be a successful covert channel, these mechanisms (table 2.1) rely on the unauthorized passage of information through a network firewall and intrusion detection system in order to communicate to a remote location on the Internet. Covert channels based on network protocol exploits that communicate to remote Internet locations can be difficult to detect and block. The main success in mitigating such channels is by first identifying the exploit and then translating that identity into a known signature that the network firewall and IDS resources recognize [4, 43].

In the context of secure wireless networks, storage channels can often be defeated with the use of encryption. In terms of the *working scenario*, we must identify covert channels that exhibit promise of survivability with respect to the 802.11 MAC level encryption. For storage channels to be effective in this context, the storage variable must exhibit some observable property other than it's contents (ex. variable size) which, despite the application of an encryption process, will represent information for the purposes of "signalling" a covert message.

Table 2.1: Summary of Network-Based Covert Channels from [43]

Layer	Protocol	Field	Remarks
Application	HTTP	HTTP request	Text field
Application	HTTP	HTTP request entity-body	Text field
Application	HTTP	HTTP response	Text field
Application	HTTP	HTTP response entity-body	Text field
Transport	DNS	ID	16-bit field
Transport	DNS	QDCOUNT	16-bit field
Transport	DNS	ANCOUNT	16-bit field
Transport	DNS	NSCOUNT	16-bit field
Transport	DNS	ARCOUNT	16-bit field
Transport	DNS	QNAME	Text field
Transport	TCP	TCP Sequence Number	32-bit number
Transport	TCP	TCP Ack Number	32-bit number
Network	ICMP	Payload	Data field
Network	IP	IP Identification	16-bit field
Network	IP	IP Flags	3-bit field
Network	IP	IP Options	24-bit field
Network	IP	IP Padding	8-bit field

2.6 Digital Communication System

The basic function of a digital communication system is to transfer data from an information source to a destination [24]. Based on the *working scenario* and the accepted definition for covert channels, it is expected that the desired covert communication system will behave like a digital communication system and pass information between a source and destination processes. For this reason, the basic model for a digital communication system [9, 24, 44, 40] will be adopted as the basis for the covert communication design. The following are brief descriptions of each component of the model as seen in figure 2.10.

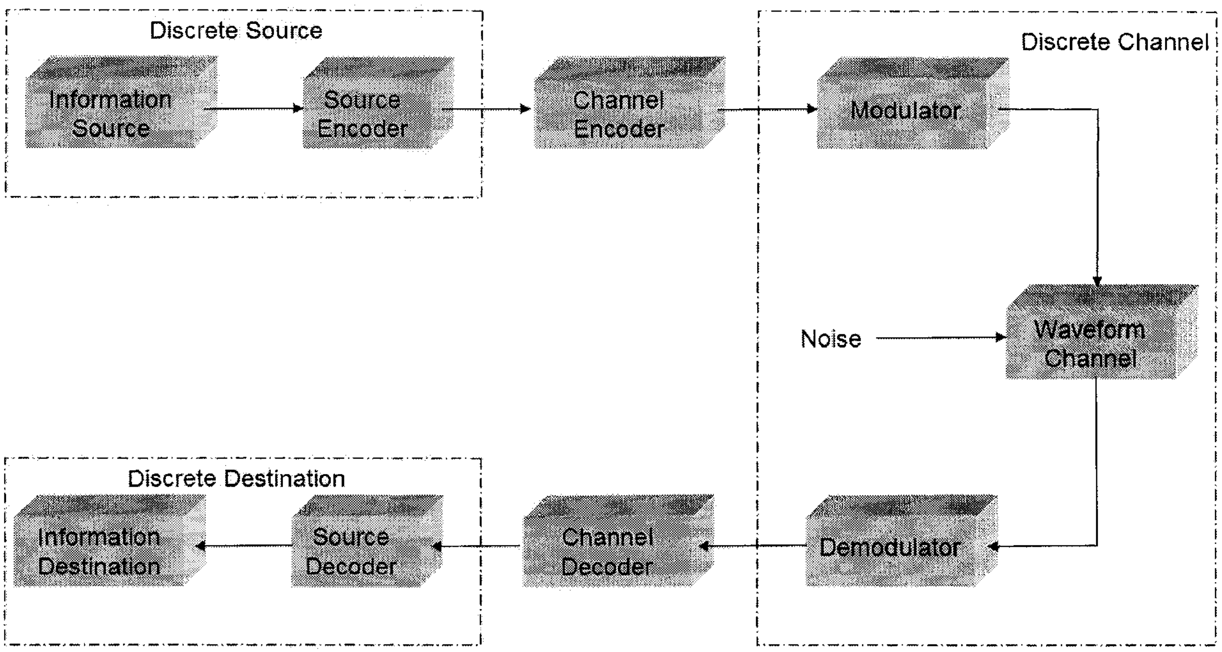

Figure 2.10: A Digital Communication System from [24, 9]

2.6.1 Discrete Data Source

The discrete data source is a process that produces a sequence of discrete symbols from a finite alphabet. The information source in this context is a person or process that originates the initial message. In the context of the *working scenario*, the information source will be a process that outputs the ASCII text that composes the wireless security passphrase. The Source Encoder accepts the output from the Information Source and converts it to a binary sequence referred to as the information sequence $\mathbf{u}$ [24].

2.6.2 Channel Encoder

The Channel Encoder adds redundancy to the information sequence $\mathbf{u}$ and transforms it into a discrete encoded sequence $\mathbf{v}$. The purpose of this transformation is to provide additional information within the message to allow the Channel Decoder to correct for any errors in the encoded sequence that may have occurred during its passage through the transmission channel.

2.6.3 Discrete Channel

The Discrete Channel accepts a sequence of discrete symbols as an input and then outputs a sequence of discrete symbols that may or may not be identical to the input sequence. The Channel consists of a Modulator, Channel, and a Demodulator. The modulator accepts symbols from the Channel Encoder and converts each symbol value into a different waveform that is suitable for transmission. Therefore, the number of different waveforms is equivalent to the number of different symbols. The Channel is a physical medium that allows for the transmission of the waveform signals. As the waveforms enter the Channel they are intermixed with the noise of other similar waveforms. The Demodulator recovers the series of intermixed waveforms and converts them to a corresponding sequence of received symbols $\mathbf{r}$.

2.6.4 Channel Decoder

The Channel Decoder accepts the sequence $\mathbf{r}$ from the Demodulator and transforms it into an estimated binary sequence $\hat{\mathbf{u}}$. The term decoding in this context also includes an error detection and correction process in addition to decoding the received encoded symbols. This entails developing a error correction strategy based on the types of errors that will occur in the targeted channel and melding it with a symbol decoding process. Ideally, the resulting estimated binary sequence $\hat{\mathbf{u}}$ is the same as the initial binary sequence $\mathbf{u}$.

2.6.5 Discrete Data Destination

The Discrete Data Destination is the target audience for the data sent by the Discrete Information Source. In order to review the received data, the Source Decoder converts the estimated sequence $\hat{\mathbf{u}}$ into an estimate of the original output message from the Information Source and delivers it to the Destination. In relation to the *working scenario*, the Destination is an ASCII text display on the intelligence agent's computer that reveals the security passphrase for the secure wireless network.

2.6.6 Error Control Strategy

Another item to note from the *working scenario* is the unidirectional nature of the covert communication system. This means that the eavesdropping intelligence agent does not communicate with the software Trojan but only listens for its signal. Therefore, if there are errors present in the received transmission, the Source Decoder must be able to perform the error correcting process without any help from the software Trojan. Error control in this uni-directional context must be accomplished through the use of Forward Error Correction (FEC) [24]. FEC also allows the system to perform error correction on the received data without waiting for the entire sequence of symbols that composes the message. The functionality of FEC can be achieved through the employment of error correcting codes.

2.7 Error Control Coding

In an effort to design a covert communication system that will operate in a noisy environment and reliably reproduce a desired message, a review of current digital communication system terminology and error control methods is required. Inherent to the primary research goal, the desired communication is to be "covert" thus difficult to detect by way of concealment amongst other network traffic. Error control techniques represent an essential component in the effort to achieve reliable communications within the targeted wireless networking environment.

Coding should not be confused with encryption or encipherment. These techniques are specifically used to conceal the contents of a message. On the other hand, the term coding in this context refers to the process by which redundant information is systematically introduced into a message prior to transmission for the purposes of faithfully recovering the original message at the receiver despite the introduction of errors [9]. When making reference to coding, we also refer to the area of research known as Coding Theory or more specifically, Channel Coding, a class of Coding Theory which deals with the minimizing of received message error by employing a code for the purposes error detection and correction. Channel coding and the use of codes for error correcting techniques has been successfully implemented and used in applications as diverse as correcting Compact Disc (CD) audio errors due to dust and scratches to removing noise from NASA deep space mission communications with the Voyager spacecraft.

2.7.1 Types of Codes

The two basic types of error control codes in use today are block codes and convolution codes. The primary differences between block codes and convolutional codes lies in the way an encoder utilizes each code type to produce an encoded message. An encoder that employs block codes divides the binary input stream into message blocks of k information bits each. The message block is represented by the binary k-tuple $\mathbf{u} = (u_1, u_2, ..., u_k)$ called a message. The encoder can therefore only process 2^k different possible input blocks. As binary message $\mathbf{u}$ arrives in the input stream, the encoder then independently translates the message into a corresponding binary n-tuple $\mathbf{v} = (v_1, v_2, ..., v_n)$ from a look-up table of discrete symbols of length n bits called code words. Each code word represents an individual symbol of an alphabet containing 2^k different output symbols. Therefore there are just as many individual code words as number of possible input blocks. The resulting code is thus defined by the length of the input block k and the length of the output symbols n as a (n, k) block code. Table 2.2 is an example of a block code look-up table for a (7,4) block code [24]. The rate of a block code is calculated as $\boldsymbol{R} = k/n$ where $k \leq n$. Since the output symbol (code word) from a block code encoder depends only on the input symbol (input message), the encoder is considered memory-less.

An encoder that employs a convolutional code also accepts k-bit blocks and also produces an n-bit output symbol [24]. Although, in a convolutional encoder, the encoded output symbol is not just dependent on the input symbol at the same time reference. An output symbol of a convolutional encoder is dependant on the previous m input symbols, where m is the order of memory within the encoder. Therefore, the set of n-bit output symbols produced by k input bits with m encoder memory registers is called a (n,k,m) convolutional code. The rate of a convolutional code is

Table 2.2: Look-Up Table for a (7,4) Binary Block Code from [24]

Messages	Code Words
0000	0000000
1000	1101000
0100	0110100
1100	1011100
0010	1110010
1010	0011010
0110	1000110
1110	0101110
0001	1010001
1001	0111001
0101	1100101
1101	0001101
0011	0100011
1011	1001011
0111	0010111
1111	1111111

also calculated as $\boldsymbol{R} = k/n$ where $k \leq n$. Convolution codes take their name from the linear superposition (modulo-2 addition) process of an impulse response characteristic of the encoder. The modulo-2 (mod2) addition of time shifted impulse responses is an overlapping process which results in a coded output symbol [20].

Another way to express how convolution encoders perform mod2 addition is through the use of generator polynomials (g_n). Since each bit of the output symbol from a convolutional encoder is a result of mod2 addition from different memory resisters within the encoder, the generator polynomial is a way of articulating which memory register contributed to the addition of a particular bit in the output symbol (ex. $g_1 = (1, 1, 1, 1)$) as seen in figure 2.11. Therefore, generator polynomials are unique code parameters that convey the behaviour of a particular convolutional code.

Given the parameters and the generator polynomials that uniquely describe a particular code, a look-up table can then be produced that also articulates the convolutional code's unique behaviour as seen in table 2.3. The convolutional encoder look-up table consists of four items: The Input Bit(s) to the encoder, the Current State of the encoder, the Output Bits and the Next State of the encoder. The states

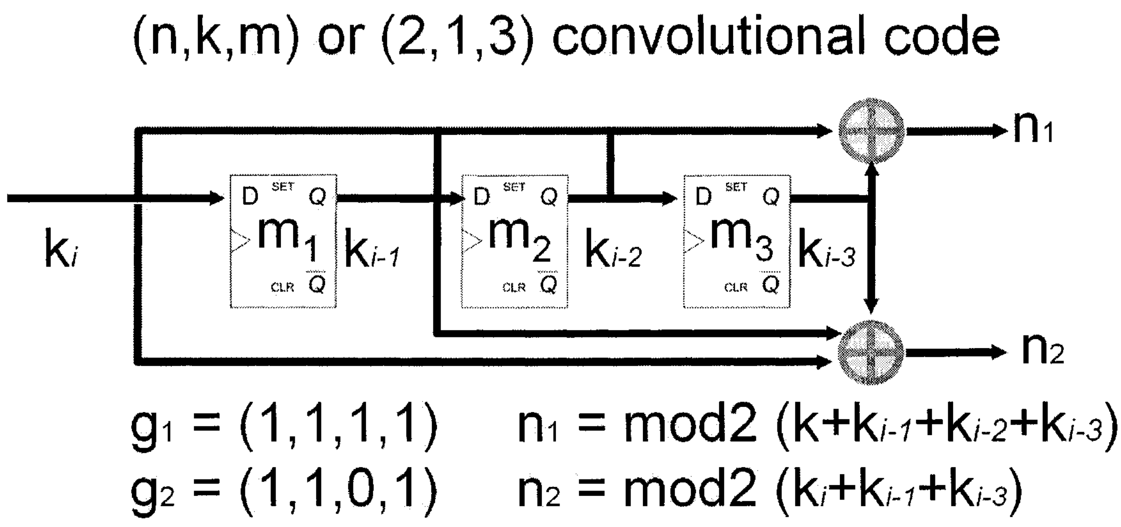

Figure 2.11: Convolutional Code Encoder and Generator Polynomials

of the convolutional encoder are represented by the sequence of the bits within each the memory registers (ex. if $m_1 = 0$, $m_2 = 1$, $m_3 = 1$ then $s_1 = 0$, $s_2 = 1$, $s_3 = 1$ and the current state is 011). For example, if there are 3 memory registers ($m = 3$), then the encoder will possess 8 (2^m) states. Thus the convolutional encoder is a finite state machine which emits a coded symbol during each state transition. The trigger to make the transition between states is the value input bit sequence.

A convolutional encoder can therefore be implemented with a series of memory storage registers and mod2 addition operations or by just referencing the look-up table. The employment of a look-up table by an encoder can simplify the design and implementation of an encoder by reducing the requirement to perform the necessary mathematical functions to produce encoded symbols [20]. It is expected that the Software Trojan from the *working scenario* will employ a look-up table vice memory storage and adders to generate the coded symbols for the covert communication system.

Since convolutional code output symbols are dependant on the previous receipt of input symbols, they would appear to have a better error detection and correction characteristic over block codes. Furthermore, the expected errors in the targeted networking environment will be coded symbol insertions and deletions, the use of

Table 2.3: Look-Up Table for a (2,1,3) Convolutional Encoder from [20]

Input Bit	Current State	Output Bits	Next State
I_1	$s_1s_2s_3$	O_1O_2	$s_1s_2s_3$
0	0 0 0	0 0	0 0 0
1	0 0 0	1 1	1 0 0
0	0 0 1	1 1	0 0 0
1	0 0 1	0 0	1 0 0
0	0 1 0	1 0	0 0 1
1	0 1 0	0 1	1 0 1
0	0 1 1	0 1	0 0 1
1	0 1 1	1 0	1 0 1
0	1 0 0	1 1	0 1 0
1	1 0 0	0 0	1 1 0
0	1 0 1	0 0	0 1 0
1	1 0 1	1 1	1 1 0
0	1 1 0	0 1	0 1 1
1	1 1 0	1 0	1 1 1
0	1 1 1	1 0	0 1 1
1	1 1 1	0 1	1 1 1

convolutional codes would appear to more practical [44] than block codes. In addition, having the knowledge of which symbols are expected next in a coded message based on the previous received trend is a distinct advantage for detecting and correcting errors.

Given that the output symbols of a convolutional encoder follow a pattern, convolutional codes possess limited self-synchronizing capability [44] and may not require any message framing to allow for a reliable decode of the covert message. Furthermore, since the convolutional encoder is a finite state machine for a discrete communication system [40] the output symbols are not directly related to the input symbol such as in block codes. Therefore, the decoding of the covert message, if intercepted by unintended recipients, would be more challenging. Given the success potential convolutional codes possess to deal with the unique error modes expected in the wireless channel along with other potential side benefits of synchronization and obscurity, convolutional codes will be selected as the error correcting code to be used in the design of the covert communication system.

2.7.2 Convolutional Code Graphical Representations

Graphical representations are often used by convolutional encoder designers in order to get a better appreciation for the operation or behaviour of a particular convolutional code. The three graphical representation methods that are often employed are: State Diagram, Tree Diagram and Trellis Diagram.

2.7.2.1 State Digram

A State Diagram is a collection of nodes and arrows that represent the transitions between states of a convolutional encoder as shown in figure 2.12. Each individual state is represented by a node with the state value articulated inside. The arrows represent the state transitions initiated by the input bits of the encoder. Typically, the state transition arrow is annotated with the value of the input bit(s) that give rise to the transition as well as the output bits in the form of input bits/output bits.

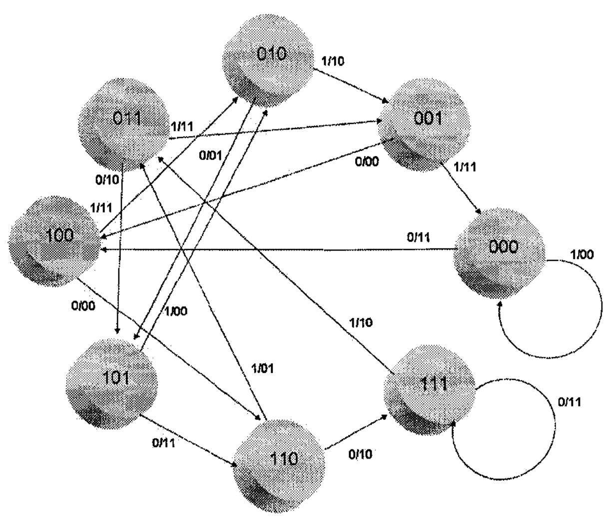

Figure 2.12: State Diagram Example for (2,1,3) code from [20]

2.7.2.2 Tree Diagram

A Tree Diagram is a binary decision tree that represents the decisions taken by the convolutional encoder over time based on the encoder input sequence as shown in figure 2.13. Although this method is helpful in representing the encoder decision behaviour based on the input bits, it was found to be of minimal use compared to the other graphical methods in the design of the covert communication system. For this reason, Tree Diagram representations will not be further employed in the remainder of this thesis.

2.7.2.3 Trellis Diagram

To better appreciate the behaviour of the convolutional encoder's behaviour over a particular time period, the generally preferred graphical representation is the Trellis Diagram as shown in figure 2.14. The x-axis of the diagram represents the progression of a discrete time reference while the y-axis represents the different states of the convolutional encoder. Lines that represent state transitions are then drawn from a particular state to all the next possible states depending on the number of input bits to the encoder. For any convolutional encoder with k input bits, there will be 2^k possible transitions from the current state. Once again, each state transition is typically annotated with the values of the input and output bits to convey which bit(s) initiated the transition and the emitted output bits for the particular transition. This annotation is typically in the form input bits/output bits. In addition, Trellis Diagrams typically begin at the "zero" state and progress to any number of desired time periods.

2.8 Summary

This chapter has presented the background material employed to design a covert communication system. The background topics discussed in this chapter focused on the *working scenario* fidelity, TCP/IP and 802.11 technology models, covert channels as well as aspects of information theory. The combination and application of these ideas and technologies will be further discussed in Chapter 3 as part of the research activities toward designing a successful *working scenario*.

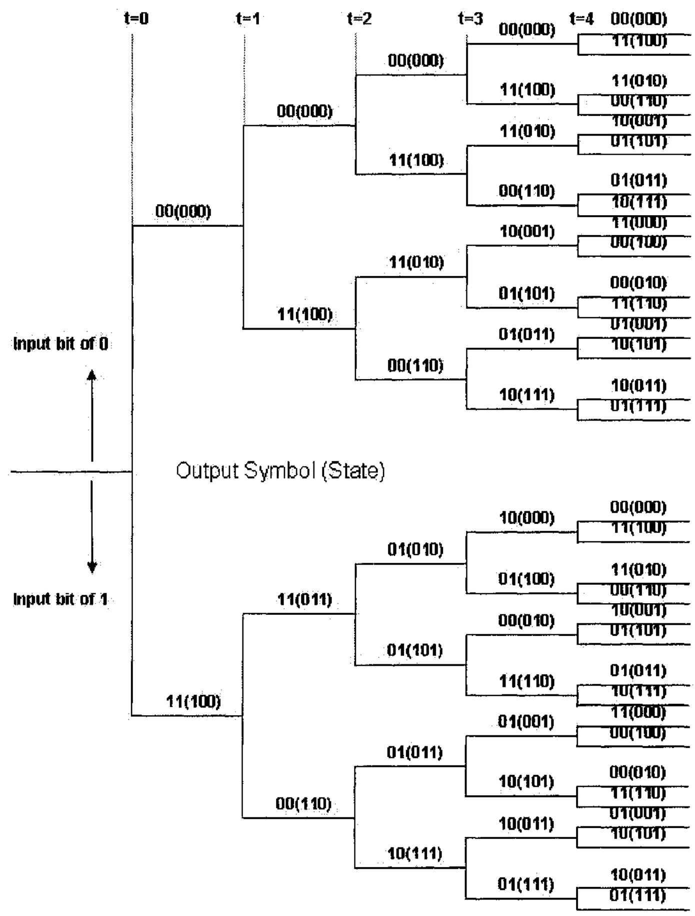

Figure 2.13: Tree Diagram Example for (2,1,3) code from [20]

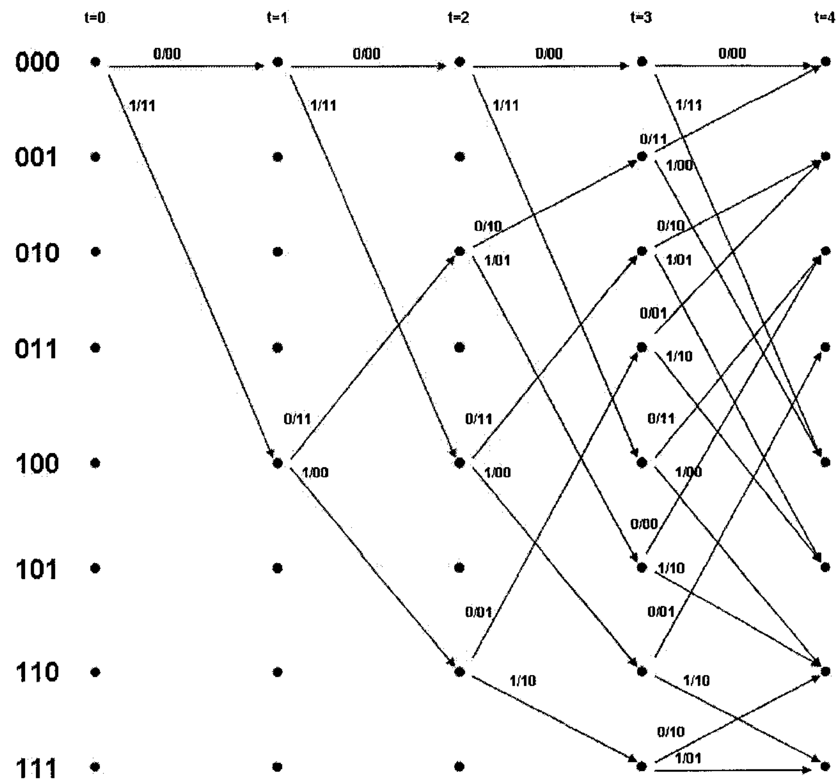

Figure 2.14: Trellis Diagram Example for (2,1,3) code from [20]

Chapter 3

New Theory and Design

3.1 Introduction

Previously in Chapter 2, ideas and technologies that formed the foundation and motivation for the research into covert channels in secure wireless networks were presented. Chapter 3 will discuss the unique combination of ideas and technologies previously presented along with some new theory that was utilized to design a working covert communication system on a secure wireless network.

In an effort to design a covert channel that conceals its communication amongst the noise of the shared wireless resource, this chapter will outline how convolutional coding along with a new error correcting methodology can be combined to detect and correct potential channel errors. In addition, this chapter will provide the details of the covert communication system design and the approach used to defeat the encryption inherent to the WPA security standard. The experimentation and validation of the design concepts presented in this chapter will be further discussed in Chapter 4 as part of the implementation of the covert communication system on a secure wireless network.

3.2 Wireless Network Covert Channel Model

The first step in designing a wireless covert communication system is to model the target environment and identify the shared resource by which to communicate. Transposing the generic model for a digital communication system presented in figure 2.10 to the problem space of an 802.11 secure wireless network and the *working scenario*, we can make direct associations between the components of the model and resources for each actor in the *working scenario* shown in figure 3.1.

The software trojan on the compromised station will perform the functions of

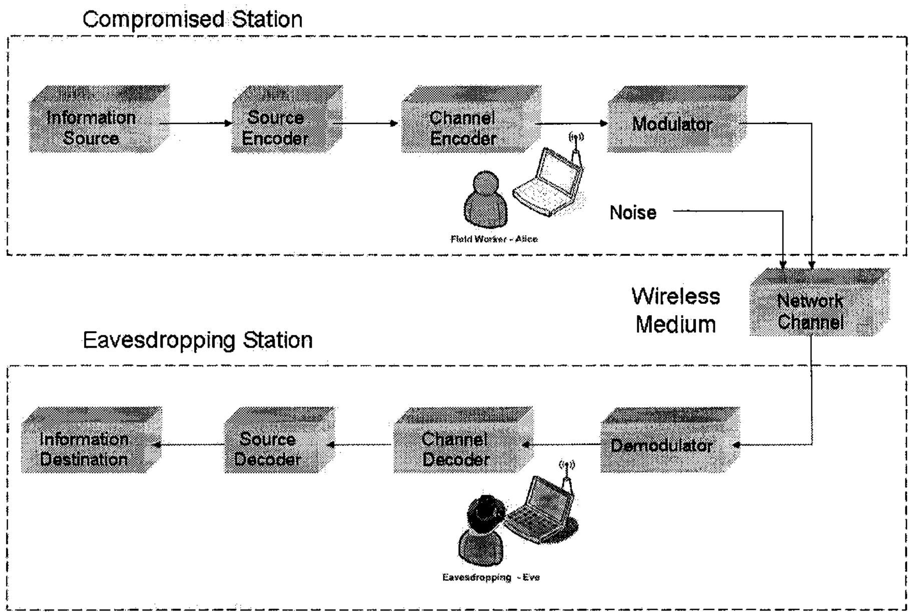

Figure 3.1: Wireless Network Covert Channel Model

the Discrete Source, Channel Encoder and Modulator. The physical Channel will represent the wireless passage of encrypted 802.11 MAC data frames between the software Trojan on the compromised station and the eavesdropping station of the intelligence agent. While the intelligence agent station conducting CNI operations will perform the functions of Demodulator, Channel Decoder, Source Decoder and Information Destination. The noise in this context will be the presence of other encrypted 802.11 MAC data frames originating from the same compromised station but generated by other legitimate processes.

In order to achieve covert communications between the software Trojan and the covert receiver of the intelligence agent, information must somehow be imbedded within the properties of the encrypted 802.11 MAC data frames. Since all higher level protocol information is encrypted as part of the data frame payload body, a covert channel technique other than steganographic must be employed to effect the shared resource of the 802.11 MAC data frame to permit communication. The proposed solution for this problem is the implementation of an appropriate covert storage channel. Utilizing a covert storage channel, the software trojan would signal information to the covert receiver through changes in the sizes of received 802.11 MAC data frames consistent with a modulated coding scheme. This signal would be unidirectional in nature as there would be no way for the covert receiver to provide feedback to the software trojan.

In addition, in order to achieve a certain level of activity concealment, the wireless covert channel must also coexist amongst the noise of other legitimate communications. This means that the sizes of the encrypted 802.11 MAC data used to transfer the encoded message must be similar to other observed legitimate encrypted traffic as not to bring attention to its irregular activity. The proposed solution for this design goal is to implement FEC error control coding techniques within the wireless covert communication system in order to detect and correct for errors within the covert message without feedback to the software trojan.

3.3 Selecting an Effective Coding/Decoding Scheme

A core idea in this research into wireless network-based covert channels is the application of proven error control coding methods for digital communication systems to the unique new problem of MAC Layer frame communication. Unlike regular digital communications systems which incur error within the discrete channel at the bit level of each individually transmitted waveform from, the noise within the covert network channel of a 802.11 MAC Layer frame communication system is represented by the addition or loss of frames where each frame represents a possible encoded symbol. Therefore, the desired coding scheme must be able to detect symbol errors within the covert channel data stream as well as take the appropriate steps to resolve the error and reliably recover the intended covert message.

Some of the research into channels with insertion and deletion errors [21, 36] suggest methods that implement marker or watermark codes and deal with the problems at the bit level. Unfortunately, these methods are somewhat impractical for this problem space. Although, the employment of marker and watermark codes in this context could potentially improve issues of synchronization and error detection, they would most definitely compromise the concealed nature of the covert system with a consistent and repetitive activity pattern.

On the other hand, convolutional codes present a viable approach to solve the problem of concealing the transmission pattern of the message while providing error detecting and correcting functionality. The non-cyclical nature of convolutional codes along with the ability to perform FEC provides the covert communication system with a reliable capability that will be harder to detect in the presence of noise.

For a convolutional code to be effective for use within a covert communication system, the decoder employing this code must be able to detect symbol errors quickly from within a sequence of symbols. In order to accomplish this task successfully, a convolutional code encoder must exhibit significant memory distances to easily distinguish breaks in the expected symbol sequence. Notwithstanding the search for a convolutional code with large output symbol distances, one must also consider the possible state errors identified at [44] of memory loss, equal and alternate paths and unequal and alternate paths as an indicator of encoder performance when employing a code in this context.

3.3.1 Memory Loss

Memory Loss in a state machine occurs when there exists the possibility for the current state to also be the next state in a loop-back transition as seen in figure 3.2. Since the encoded symbol emitted during this loop-back transition will always be the same, it could potentially be emitted several times in a row depending on the input symbol to the encoder. If this occurs, any memory of previous symbol emissions in the sequence of encoded symbols is lost as it becomes impossible to determine how many of the loop-back encoded symbols were noise or actually part of the encoded message.

In addition to the simplest form of memory loss which is manifested as a single self-loop, memory loss can also occur with loop back conditions occurring over multiple states (figure 3.3). The state transition example State A $\longrightarrow$ State B $\longrightarrow$ State A shown in figure 3.3 is also a case of memory loss due to two potential noise symbol

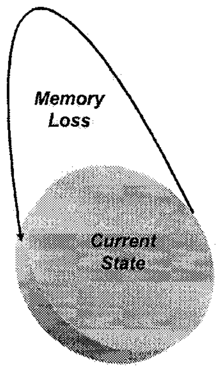

Figure 3.2: State Machine Memory Loss Error - Self-Loop

insertions which returned the state machine back to State A. Once again it becomes difficult to determine if the two loop-back encoded symbols were a result of noise or actually part of the encoded message. Even larger loop-back memory loss cases are possible, although the combinations of noise symbols required to manifest these conditions become less-probable to occur randomly in the channel. For this reason, the selected coding scheme should not contain any short loop-back state transitions to avoid the potential for memory loss due to noise symbol insertions.

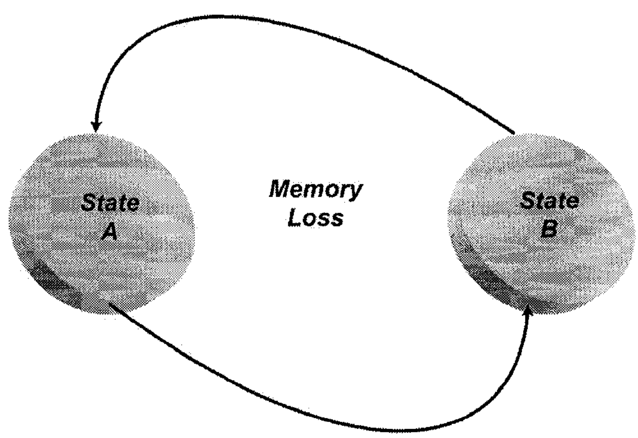

Figure 3.3: State Machine Memory Loss Error - Multiple State Loop

3.3.2 Equal and Alternate Paths

An equal and alternate path error refers to the condition in a state machine in which the transitions which lead to a particular state can be achieved through two or more paths of equal length as seen in figure 3.4. This means that the legitimate sequence of encoded symbols that leads from one particular state to another can also be replicated with an equivalent number of non-legitimate symbols in a separate sequence. This type of error posses a problem in the error detection and correction process of the Channel Decoding component of the covert channel model. Short equal and alternate paths may give way to the improper acceptance or recreation of an incorrectly encoded symbol sequences due to insertion error noise in the channel since there is no way to decide which path is the correct decoding. The selected encoding scheme must then exhibit equal and alternate paths long enough to provide the decoder the flexibility to correct for several insertion or deletion errors.

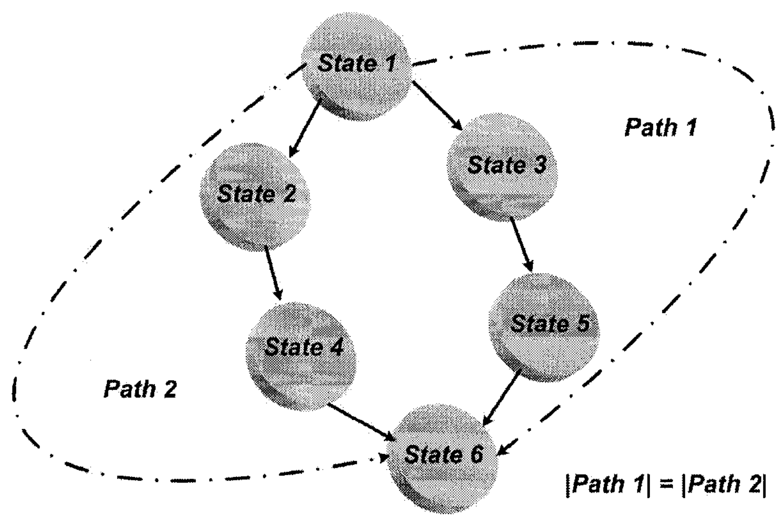

Figure 3.4: State Machine Equal and Alternate Path Error

3.3.3 Unequal and Alternate Paths

Unequal and alternate path errors refer to the condition in a state machine in which the transitions which lead to a particular state can be achieved through two or more paths of unequal length as seen in figure 3.4. This means that the legitimate sequence of encoded symbols that leads from one particular state to another can also be replicated with a smaller or larger number of illegitimate symbols in a separate sequence. This type of error also posses a problem in the error detection and correction process of the Channel Decoding component of the covert channel model as smaller unequal and alternate paths may give way to the improper acceptance or recreation of an incorrectly encoded symbol sequence due to noise in the channel. The selected encoding scheme must then exhibit unequal and alternate paths with a significant length to once again provide the decoder the flexibility to correct for several insertion or deletion errors.

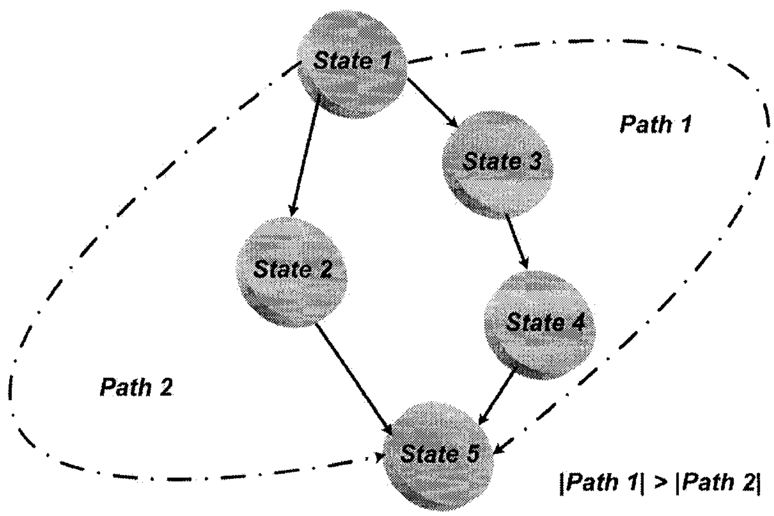

Figure 3.5: State Machine Unequal and Alternate Path Error

3.3.4 Trellis Codes

Trellis codes, like convolutional codes, can be articulated graphically through the use of state or trellis diagrams. Although, unlike convolutional codes, trellis codes are not based on an impulse response or generator polynomials to define their behaviour. Instead, a trellis code is defined by a finite state machine structure, quite often hand-designed [44, 49], which exhibits a behaviour that can be captured within a trellis diagram. Trellis codes allow for the development of codes with good output symbol memory distances while at the same time reducing the impact of the possible state errors outlined in the previous sub-sections. In addition, a trellis code's behaviour can also be represented in a look-up table that is identical in structure to those of traditional convolutional codes.

Based on the above error criteria and a decoder's ability to detect a symbol error quickly, a convolutional style code will be selected for implementation within the covert communication system. Tests results relating to the code selection will be presented further in the following chapter 4.

3.4 Error Correction Methodology

In order to correct for symbol level errors, the Channel Decoding component must first detect that an error exists in the sequence of encoded symbols, then take the appropriate action to then correct for the detected error. Once the error has been corrected, the estimated sequence of encoded symbols can then be decoded based on the chosen coding strategy. This section will discuss the methodology that was employed to correct for detected errors within the received sequence of encoded symbols.

Knowing that the errors that will occur in the 802.11 wireless covert channel will either be the noise of additional of MAC Layer frames from other processes or the loss of MAC Layer frames due to anomalies of the wireless channel, then the expected errors will be either an insertion or deletion of received encoded symbols. Given that the expected errors are symbol-level versus bit-level in nature, regular methods of error correction for convolutional decoders must be modified to adapt to this new environment. More specifically, we must consider a correction methodology that accounts for the insertion and deletion of symbols versus the corruption of individual bits from the received sequence. Therefore, another design objective for the covert communication system was to formulate a new error correction methodology. It was proposed that this new error correction methodology would be based on a series inductive steps to detect and correct for expected symbol errors from the wireless covert channel.

The first step in developing an error correction methodology was to establish a method to detecting errors in a sequence of received symbols. One proven method for decoding bit streams encoded with convolutional codes is through the use of a decoder employing a Viterbi algorithm [50]. Although a Viterbi based decoder also attempts to correct for bit-level errors based on maximum likelihood decoding, the use of such a decoder in this context was only to detect errors in the received encoded symbol sequence.

3.4.1 Error Correction Assumptions

Given an established method for error detection within an encoded symbol sequence, the next task was to develop a series of inductive steps that would attempt to correct for symbol level errors. To accomplish this task a series of assumptions were made as a framework from which the inductive steps could be produced.

Like the author at [36], it was assumed that all insertion and deletion errors that occur are random events. Since the quantity of legitimate 802.11 MAC data frames (noise) is dependant on user actions and/or communication processes active during any period of time on a wireless computer, it was expected that the level of legitimate 802.11 MAC data frames emanating form a typical wireless computer would be fairly low for the majority of a particular observation period. Given this expectation, the following five assumptions were made in the development of the inductive decision steps.

Assumption 1. The Viterbi decoder can detect single symbol deletion errors within one encoded symbol.

Assumption 2. The Viterbi decoder can detect single symbol insertion errors within two encoded symbols.

Assumption 3. The Viterbi decoder can detect double symbol insertion errors within three encoded symbols.

Assumption 4. The a symbol in a sequence of symbols $abcd$ to be evaluated is always legitimate and in the correct position.

Assumption 5. Within a sequence of four symbols ($abcd$), there can only exist a maximum of two symbol errors. These errors can range from single to double insertions errors or single deletion errors or a combination of both types of error.

3.4.2 Inductive Steps

Next in the process of developing the error correction methodology was the development of a series of inductive steps that would attempt to correct for symbol level errors. The inductive steps would outline how the received sequence of symbols would be inspected for errors as well as define the action taken to correct errors in the received sequence based on either a pass/fail result of the inspection.

A sliding window approach of six symbols (a,b,c,d,e,f) was adopted as the basis for the inductive step inspections. The inductive step inspections that formed a decision tree (figures 3.6 and 3.7) focused on identifying and eliminating one to two possible symbol errors from within a four symbol sequence of *abcd* in order to prove that the symbol that followed immediately after a in a sequence *abcd* was in fact legitimate and in the proper position. If this was accomplished during any step of the decision tree, the a symbol would be added to an estimated sequence of received symbols considered "error free" and awaiting final decode. The symbol that followed immediately after a, that was deemed "good", then became the new a symbol and the sliding window was shifted to incorporate the successfully tested error correction changes as well as to add the next symbols in the raw sequence to form a new window of symbols *abcdef*. Once the sliding window has been reformed, the process restarts with a check for errors within the new version of the *abcd* sequence.

The decision tree was divided into two sections based on the expectation that symbol insertion errors (noise) were more likely to occur than symbol deletions (frame losses). In addition, the process of correcting symbol insertions was considered less costly in terms of computing resources, compared to correcting symbol deletions. Symbol deletion correction requires that every possible symbol within the coded alphabet be attempted to complete one deletion verification. Thus, for a code with a rate $\boldsymbol{R} = k/n$, one symbol deletion may require 2^k attempts. The result was a "cost effective" decoding strategy that first looked for symbol insertion errors (figure 3.6) and then progressed to a mix of insertion and deletion errors (figure 3.7) if the sequence of the previous checks failed.

In the first stage of the decision tree as shown in figure 3.6, the sequence of *abcd* is decoded with the Viterbi decoder to deduce the sequence was error free. If that check passes and proves that *b* is legitimate and in the proper position, the *b* symbol is deemed “good” and *a* is added to the bank of “error free” symbols. The *b* symbol then becomes the new *a* symbol and the process restarts with the sliding window shifted in the sequence by one symbol. If that check fails, then the sliding window is then reconfigured to check for a double insertion error where the *d* and *e* symbols are considered “noise” ($\bar{d}$ and $\bar{e}$) and removed from the window sequence. If that check passes, then the symbol *b* is once again proven “good” and *a* is added to the bank. In addition, the process is restarted with the sequence *bcfg* assuming the new role of *abcd*. On the other hand, if the check failed then the process continues to the next step in the tree to perform another verification.

Figure 3.6: Symbol Error Correction - Inductive Step Inspections Part 1

In the second stage of the decision tree as shown in figure 3.7, the sequence of *abcd* is checked for single symbol insertions as well as possible symbol deletions (i). In the first part of this stage, the verifications focus on mixed symbol insertion and deletions and then progress to single symbol deletion errors. Similar to the first stage, if a check was successful the symbol next to a was deemed "good" and a was added to the bank. The corrected sequence then becomes the new sequence *abcd* and the process is repeated. If the process continually fails and the bottom of the tree is reached, then one of the design assumptions is considered in error. If this case is reached, then the d symbol in the sequence *abcd* is considered the first "noise" symbol ($\bar{d}$) in a burst of several other noise symbols that follow in the received sequence. The d symbol is then removed from the sequence and the process restarted with *abce* becoming the new *abcd*.

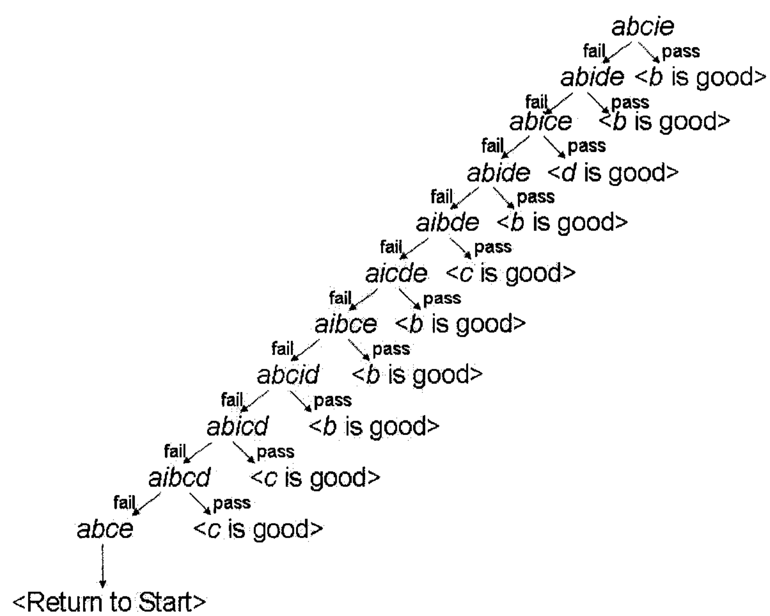

Figure 3.7: Symbol Error Correction - Inductive Step Inspections Part 2

Once the window reaches the end of the symbol sequence it must then adopt a slightly different approach as seen in figure 3.8 to correct the remaining symbols in the sliding window *bcdef*. Due to the few remaining symbols in the sequence, the previous error correction methodology can not be applied. Instead, this approach checked each symbol individually along with the error-free bank for insertions ($\bar{b}$) or deletions (i) before being considered "good" and consequently added to the bank.

If a symbol failed the initial validity check (Bank+b), a symbol deletion check was conducted between the bank of error-free symbols and the tested symbol (Bank+ib). Upon the failure of both tests, the symbol was deleted from the sequence and the next symbol (c) was tested in a similar fashion. On the other hand, if a symbol passes either the insertion (Bank+b) or deletion (Bank+ib) test, the symbols used to achieve a successful test (b or ib) are added to the bank and the process continues with the next symbol (c). When the final individual symbol test finishes at the end of the sliding window (f), the error detection and correction phase is considered complete. The resulting bank of estimated "error free" symbols is then decoded with the appropriate decoding scheme.

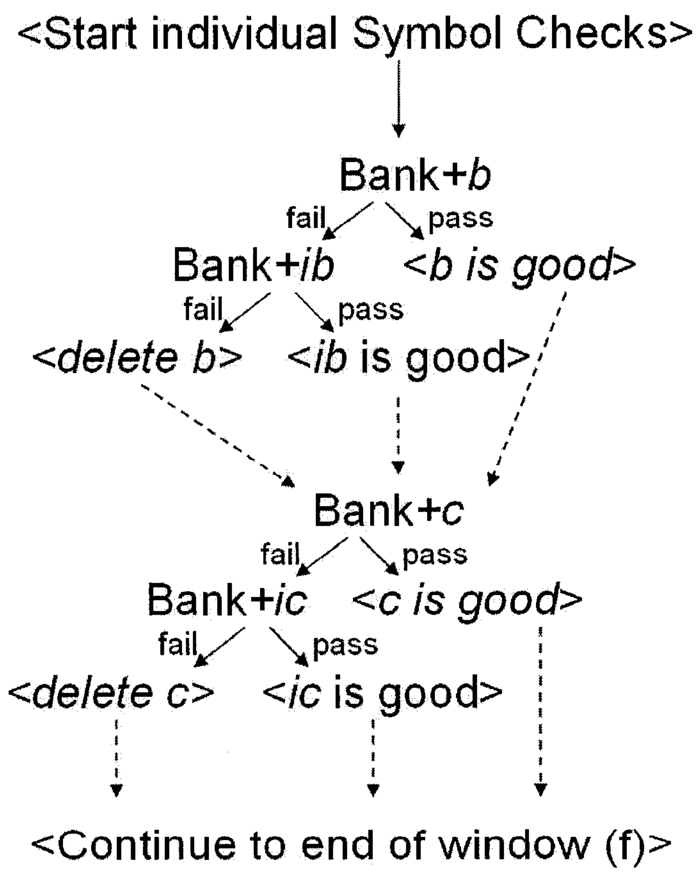

Figure 3.8: Symbol Error Correction - Final Window Inspection

3.5 Covert Communication Design

The final step in the design of the wireless covert communication system is the unique combination of all the previously discussed design concepts into one final system based on available technology and techniques. Employing a systems approach design, a selected coding/decoding scheme along with the error correction methodology will be incorporated into the wireless network covert channel model (figure 3.9) with the intention of implementation based on commercially available computing resources. The sub-sections that follow further articulate the design and operation of the wireless covert communication system in terms of its two principle components: Software Trojan and Covert Receiver.

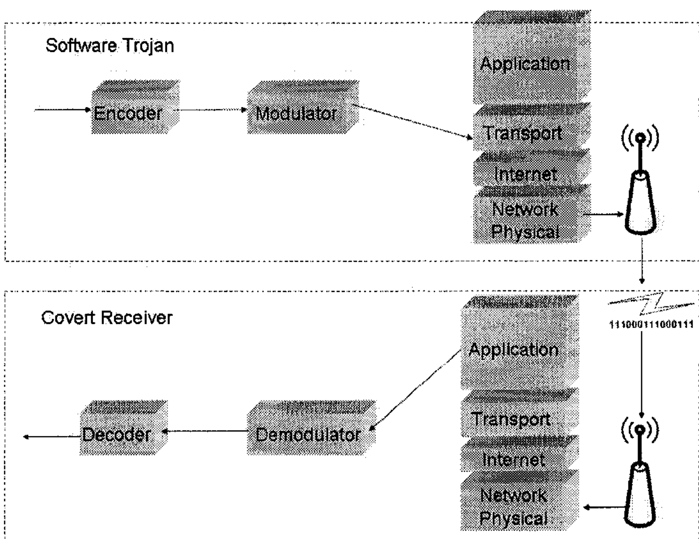

Figure 3.9: Wireless Covert Communication System Model

3.5.1 Software Trojan

The Software Trojan will be a software executable designed for use in a Microsoft Windows operating system environment similar to the widely used baseline configurations employed by both Canadian and US Militaries [1, 11, 10]. The Trojan will be designed based on the model shown in figure 3.10 and implemented with Microsoft Visual C++ software [27] using object oriented design techniques.

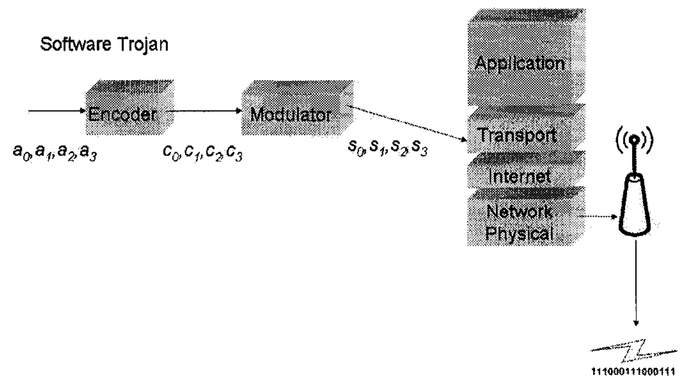

Figure 3.10: Software Trojan Model

For demonstrative purposes, the Trojan will accept ASCII text from the keyboard of the compromised computer and convert it to a stream of equivalent binary values. The individual bits of those binary values are represented in the model by the sequence a_0, a_1, a_2, a_3. The Trojan encoder will then take the stream of binary values and produce a sequence of encoded symbols (c_0, c_1, c_2, c_3) based on the selected coding strategy. The value of each individual encoded symbol c_0, c_1, c_2, c_3 is then used by the modulator to generate a sequence of data payloads (s_0, s_1, s_2, s_3) with sizes corresponding directly with the value of the encoded symbol. More specifically, there is a direct mapping between encoded symbol c_i and the data payload size s_i of the network data packet (and resulting 802.11 MAC frame size). The data payloads are to consist of a series of ASCII characters combined in such a fashion to not bring attention to its purpose if intercepted. The data payload will then be included in the crafting of a Transport Layer UDP echo request [37] as the data portion of the UDP packet [1]. The resulting UDP echo request packet will then be transmitted across the wireless channel as an encrypted 802.11 MAC frame to the wireless access point of the associated compromised station. The wireless Access point will be utilizing the

[1]Note: We have chosen to implement the covert communication system using UDP echo packets. However, any network packets may be used to create the desired covert storage channel and the employment of UDP echo packets does not result in any loss of generality.

WPA security standard with AES encryption. The sequence of UDP echo requests that represented the modulated covert message will be proceeded and terminated by framing symbols which are also size modulated into UDP echo requests. Since the covert message will be beaconed in a continuous fashion once the Trojan has been started, the framing symbols will provide the covert receiver a reference from which to decode each iteration of the beaconed message.

3.5.2 Covert Receiver

The Covert Receiver as shown in figure 3.11 represents a collection of computing resources used to covertly intercept and process 802.11 MAC data frames for the purposes of recovering the covert message sent by the Software Trojan. This means that the Covert Receiver need not be implemented all in one programming environment or even all on one computer/operating system. The Covert Receiver could be a composed of a series of applications and process which when combined together perform the task of interception and reliable reproduction of the covert message. In relation to the *working scenario*, it would not be unrealistic for an intelligence agent conducting CNI operations to be using a high performance antenna as well as computing resources utilizing several different tools to recover a covert message. For the purposes of this research, the Covert Receiver will be designed based on readily available commercial computing resources.

The Covert Receiver will capture the AES encrypted 802.11 frames with a wireless sniffing application such as Kismet [17] and a wireless network card operating in promiscuous mode [51]. If Kismet is chosen, then a Linux operating system such as Ubuntu [3] will be used. It is crucial that the chosen operating system contain wireless network card drivers that operate in promiscuous mode to effectively eavesdrop and capture 802.11 MAC frames without associating with the access point.

Once the 802.11 MAC frames are captured by the sniffing application and stored in a capture file, the frame lengths from a targeted station can be filtered by individual MAC addresses. This data manipulation and filtering of the captured dump file can be accomplished with the use of a protocol analyzing application such as OmniPeek

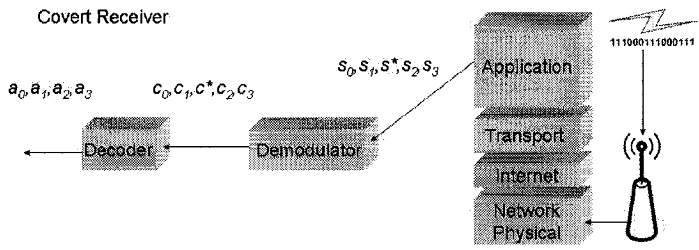

Figure 3.11: Covert Receiver Model

Personal [53]. The filtered MAC frame lengths of the targeted station will produce a sequence of received modulated symbols $(s_0, s_1, s^*, s_2, s_3)$ that may also include noise, s^*.

The remaining components of the covert receiver can be implemented in the MATLAB [26] programming environment. The sequence of modulated symbols s_0, s_1, s^*, s_2, s_3 can be loaded into MATLAB and demodulated to a sequence of encoded binary symbols $(c_0, c_1, c^*, c_2, c_3)$ based on MAC data frame lengths associated with the encapsulated UDP echo requests from the Software Trojan. Like the modulated symbol sequence, the sequence of encoded binary symbols may contain noise symbols (c^*) which are carried forward as part of the demodulation process. The sequence of encoded symbols c_0, c_1, c^*, c_2, c_3 are then sent to the decoder for error correction and decoding based on the selected encoding strategy. The decoder will use the Viterbi decoding function resident in MATLAB to detect the error, c^*, within the sequence. Once the error is detected, the error correction checks specified by the correction methodology will then be applied to the sequence to prune out the noise (c^*). The corrected sequence of encoded symbols c_0, c_1, c_2, c_3 is then passed to the Viterbi decoder to translate it into the estimated binary sequence a_0, a_1, a_2, a_3 of the covert message based on the coding strategy used in the Software Trojan. Finally, the binary sequence a_0, a_1, a_2, a_3 is then converted back to ASCII characters to reveal the contents of the private information leaked from the secure wireless network.

3.6 Summary

This chapter has presented the details of the covert communication system design and the approach used to defeat the encryption inherent to the 802.11 security standard. In addition, this chapter outlined how trellis coding along with a new error correction methodology can be combined to detect and correct potential channel errors. The experimentation and validation of the design concepts presented in this chapter will be further discussed in Chapter 4 as part of the implementation of the covert communication system on a secure wireless network.

Chapter 4

Experimental Analysis and Validation

4.1 Introduction

Earlier in Chapter 3, a unique combination of ideas and technologies utilized in the design of a covert communication system were presented. With the intent to realize the designs of the previous chapter, the focus of Chapter 4 will be to outline the experimentation, results and validation that led to the successful implementation and testing of a working covert communication system on a secure wireless network.

As part of the process to implement a working covert communication system on a secure wireless network, several experiments were conducted to validate the design concepts and specifications. This chapter will describe each of the experiments and their results that led to a successful achievement of the research goal. A more comprehensive discussion of the resulting covert communication system's implementation and performance will follow in the concluding fifth chapter that follows.

4.2 Error Detection and Code Selection

The first series of experiments of this research centred on detecting a symbol error in a sequence of encoded symbols as well as measuring the performance of a code's ability to assist in that task. It must be noted that the focus of this research was not to develop a particular code for this application, but to perform a survey of available convolutional and trellis codes and determine an appropriate code which is suited for implementation in a covert communication system.

These initial error detection tests were conducted in the MATLAB programming environment utilizing the MATLAB implementation of the convolutional encoder (convenc) as well as the Viterbi decoding algorithm (vitdec) to detect the symbol sequence errors. In order to perform the task of error detection within an encoded sequence with these two MATLAB functions, a binary encoded sequence of symbols

$(c_0c_1c_2c_3)$ would first be decoded into an estimated decoded stream of binary symbols $(\hat{a}_0\ \hat{a}_1\ \hat{a}_2)$ with the vitdec function. The same decoded binary stream $(\hat{a}_0\ \hat{a}_1\ \hat{a}_2)$ would be re-encoded with the convenc function employing the same coding strategy to produce an estimated representation of the original encoded sequence of symbols $(\hat{c}_0\ \hat{c}_1\ \hat{c}_2\ \hat{c}_3)$. If both encoded sequences matched (received and estimated received) then the original received sequence is considered error free.

On the other hand, if the two encoded sequences did not match, then there were symbol error(s) present in the original received sequence requiring correction. As mentioned in the previous chapter, Viterbi based decoders attempt to correct for bit-level errors based on maximum likelihood decoding. If the Viterbi decoder is provided with an encoded symbol sequence with errors $(c_0c_1c^*c_2c_3)$, the Viterbi decoder will perform bit level error correction in order to produce an estimated decoded stream of binary symbols $(\hat{a}_0\ \hat{a}_4\ \hat{a}_3)$ based on maximum likelihood bit error correction. When the estimated decoded stream of binary symbols $(\hat{a}_0\ \hat{a}_4\ \hat{a}_3)$ is re-encoded to reproduce an estimated representation of the original encoded sequence of symbols $(\hat{c}_0\ \hat{c}_1\ \hat{c}_5\ \hat{c}_4)$ the two encoded sequences will not match due to the bit level error correction performed by the Viterbi decoder. Therefore by applying the Viterbi based decoder to a received sequence of encoded symbols, symbol-level errors can be positively identified.

4.2.1 Single Symbol Insertion Experiment

The first experiment in this series of tests involved the detection of a single symbol insertion error in an encoded sequence. To simulate this type of error, a string of ASCII text (*The quick brown fox jumped over the lazy dog's back.*) was initially encoded with a particular coding strategy to produce a sequence of 416 error-free encoded symbols *abcdefg...*. The error-free sequence of symbols was then modified with the insertion of an individual symbol error $\bar{a}$, from the alphabet of 2^k possible encoded output symbols, into a position between the fifth and sixth symbols of the encoded sequence $(abcde\bar{a}fg\ldots)$.

The resulting sequence was then tested for symbol errors by initially testing the first four symbols in the sequence *abcd* which were known to be error-free. This initial testing window of four error-free symbols was then increased in size by one symbol *abcde* and retested for errors. This process continued until the inserted error symbol was added to the testing window ($abcde\bar{a}$) and the window failed the comparison check. At the point at which the window fails the comparison check, the inserted error symbol is considered detected.

Once an error was detected, the symbol position in the modified sequence at which the error was discovered was then compared with the actual position in the sequence of the symbol error insertion. The difference between the two sequence positions would then indicate the effectiveness of the code in question in detecting an error given the symbol employed ($\bar{a}$) for the insertion. On the other hand, if no error was detected and the testing window reached the end of the encoded sequence, the testing would cease for that particular inserted symbol taking note that the error was not detected.

At the end of testing for the inserted symbol ($\bar{a}$) in a particular position of the sequence (6^{th}), that same symbol would then be inserted in the next symbol position (7^{th}) of another error-free encoded sequence ($abcdef\bar{a}g...$) and the entire process re-executed. This incremental approach to symbol insertion placement and testing would continue until the insertion was placed in a final position, 20 symbols from the end of the sequence. The window of 20 symbols was selected to provide poorer performing strategies the ability to detect an insertion error before reaching the end of the encoded symbol sequence.

Once the symbol insertion tests for the first insertion symbol $\bar{a}$ were completed, the next symbol in the alphabet of encoded symbols was then selected to be a symbol insertion ($\bar{b}$) and the same series of insertion tests were re-executed starting once again from the 6^{th} symbol position ($abcde\bar{b}fg...$). These repetitive series of tests would continue until all the symbols in the 2^k alphabet of encoded symbols were verified. The error detection results for all the inserted symbols were then combined as both an average and a worst case maximum to provide an indication of the overall performance of the implemented code's ability to detect single symbol insertion errors. The results of the single insertion tests can be seen in table 4.1.

Table 4.1: Single Symbol Insertion Error Detection

Code Rate	Memory	Ave Detect in Sym	Max Detect in Sym	Missed Errors	Source
1/5	4	1.063	2	0	[44]
1/4	2	N/A	147	2	[24]
1/4	3	1.125	2	0	[44]
1/4	7	1.136	4	0	[24]
1/4	9	1.146	5	0	[24]
1/3	9	1.332	6	0	[24]
1/3	11	1.332	6	0	[24]
1/2	9	1.999	10	0	[24]
1/2	13	2.023	10	0	[24]
2/3	4	N/A	14	1	[24]
2/3	5	N/A	12	1	[24]
3/4	3	N/A	9	2	[24]

The tabulated results of the single symbol insertion error experiments appear to form a trend in which the ability to detect symbol insertion errors improved as the code rate was lowered from 1/2 to 1/5 rate convolutional codes. In addition, the increase in memory depth for a particular code rate gave no significant improvement in performance in detecting an insertion in the sequence of symbols. What is also interesting to note is that the two top performing codes (Rate 1/5 and 1/4) from [44] were not convolutional codes but rather trellis codes. Although the traditional convolutional codes taken from [24] were considered "good" codes, many of them contained error inducing artifacts within their respective state machines such as memory loss or short unequal and alternate paths. These artifacts along with the proper combination of legitimate and inserted symbols would cause cases of poor error detection, if errors were detected at all.

The author of [44] has attempted to develop more efficient codes for use in a network-based covert communication system based on a state machine structure. The resulting trellis codes behave similarly to regular convolutional codes but have been designed with a state machine structure that specifically reduces potential errors such as memory loss, short equal and alternate paths and shorter unequal and alternate paths as previously discussed on page 41. Since the trellis codes significantly outperformed the remaining field of convolutional codes in detecting single symbol insertions, they were the only codes retained for further comparison in detecting potential wireless covert channel errors. Furthermore, the trellis codes confirmed the second design assumption on page 45 by consistently detecting a single symbol insertion within two symbols.

4.2.2 Single Symbol Deletion Experiment

The second experiment in the series of encoded sequence error detection involved the detection of a single symbol deletion from an error free sequence of encoded symbols. Similar to the last experiment, a string of ASCII text (*The quick brown fox jumped over the lazy dog's back.*) was encoded with a particular coding strategy to produce a sequence of error-free encoded symbols *abcdefg....* To simulate this type of error, the resulting error-free sequence of symbols was modified by pruning out one of the legitimate symbols from the sequence. This removal of a legitimate symbol first occurred with the symbol that occupied the 6^{th} position of the sequence (*abcdeg...*).

This experiment also adopted an expanding testing window approach that started with an error-free sequence of four symbols (*abcd*). One should note that the testing window of four symbols relates back to the design assumptions outlined in Section 3.4.1. The testing window would continually expand by adding the next symbol in the modified sequence and then repeating the error detection procedure until the error was detected. Once the error was detected, the sequence position of the last added symbol would be compared with the position of the known symbol deletion to derive the detection metric in symbols.

Once the symbol error was detected in the 6^{th} position of the modified sequence, a new symbol deletion was created at the 7^{th} position (*abcdefh...*) of an error-free sequence and the test re-executed on the new deletion error. These tests would continue until the last iteration of a symbol deletion tests, 10 symbols from the end of the sequence, was completed. Once again, a window of 10 symbols was selected to provide poorer performing strategies the ability to detect a deletion error before reaching the end of the encoded symbol sequence. At the end of the testing sequence, all the detection metrics would then be combined to reflect the overall performance of a particular code's implementation at detecting symbol deletions from a sequence of encoded symbols. The results of this experiment are shown in table 4.2.

Table 4.2: Single Symbol Deletion Error Detection

Code Rate	Memory	Ave Detect in Sym	Max Detect in Sym	Missed Errors	Source
1/5	4	1.0	1	0	[44]
1/4	3	1.0	1	0	[44]

Given the results of the symbol deletion experiment in table 4.2, neither trellis code exhibited a performance significantly better than the other for detecting this type of error. The design of both codes allows for the immediate detection of a missing symbol from a sequence of encoded symbols. Although there is no clear winning code for this round of experimentation, both codes performed in a fashion that is not only desirable, but consistent with the first design assumption for the wireless covert communication system on page 45. For these reasons, both trellis codes were once again retained for further experimentation.

4.2.3 Double Symbol Insertion Experiment

The third and final experiment in this series involved the detection of two consecutive symbol insertions in an otherwise error free sequence of encoded symbols. The two symbol insertions would be a combination of 2^k possible output symbols for a particular code. Therefore, this experiment would generate $(2^k)^2$ possible combinations of coded symbols which would be inserted into error-free sequences similar to the single symbol insertion experiment. Due to the already significant number of test vectors generated from the possible combinations of symbols, the string of ASCII text used to produce the sequence of error-free encoded symbols was reduced to *Hello World* to achieve results in a more timely manor.

The double symbol insertion error was simulated by inserting a combination of error symbols ($\bar{a}\bar{a}$) into an error-free sequence *abcdefg...* starting at the 6^th position

($abcde\bar{a}\bar{a}fg...$). Similar to the previous experiments, an expanding testing window approach that started with an error-free sequence of four symbols ($abcd$) was also employed. The four symbol window would be tested for errors and then expanded. Once again, the next symbol in the sequence would be added to the testing window and checked for errors until an error was detected. When the error was detected, the symbol position in the sequence at the point of detection was then compared with the symbol position of the first symbol insertion to calculate the detection metric in symbols for a double insertion error.

Once the test was conducted with two symbol insertions starting at the 6^{th} position in the sequence, the same combination of insertions were then place at the 7^{th} position of a fresh error-free sequence ($abcdef\bar{a}\bar{a}g...$) and the detection process re-executed. This re-execution of the experiment would continue until it reached a final insertion positon, 10 encoded symbols from the end of the error-free sequence. Once the final double insertion test was conducted for the particular combination of insertion symbols ($\bar{a}\bar{a}$), another combination of symbols ($\bar{a}\bar{b}$) were then used to re-conduct the experiment. This would continue until all the possible combinations of $(2^k)^2$ symbols were attempted. Once the testing was complete, the detection metrics for all combinations of symbol insertions were then combined to reflect the detection performance of a particular code's implementation for this type of error. The results for this experiment can be seen in table 4.3.

Table 4.3: Double Symbol Insertion Error Detection

Code Rate	Memory	Ave Detect in Sym	Max Detect in Sym	Missed Errors	Source
1/5	4	N/A	3	0	[44]
1/4	3	N/A	3	0	[44]

Once again, both trellis codes performed admirably as all of the double insertion errors were detected from the multiple combinations of test vectors. Although the maximum detection window for this type of error was three symbols in the cases of both codes, this is not an indication of poor performance. In fact, this metric indicates the exact opposite since the minimum possible detection window for this type of error is three symbols. Since every possible combination of encoded symbol was inserted to simulate noise, it is possible that both inserted symbols achieve a combination that is in fact the next two legitimate symbols in the sequence. The only indication of a break in the legitimate sequence would be upon the arrival of the next legitimate symbol (ex. $abcde\bar{f}\bar{g}f\ldots$). Since this type of false symbol alignment infrequently occurs for all possible combinations of noise symbols, the average window of detection is potentially lower. Nevertheless, for the purposes of designing the wireless covert communication system, the maximum window of detection of three symbols will be used for this type of error. Furthermore, the error detecting performance of the two trellis codes for this type of error confirms the third design assumption made on page 45 of this thesis.

Both trellis codes ((4,1,3) and (5,1,4)) performed well at detecting encoded symbol sequence errors as seen in the previous three experiments. Due to their relative equivalence in performance for the expected environment of a wireless covert channel, both were retained for further observation in the next round of experimentation of integrating error detection with the error correction methodology.

4.3 Error Correction Capability

This next series of experiments focused on integrating the error detection capability established previously with trellis coding and the inductive steps from page 47 to form an effective symbol level error correction capability for the covert receiver portion of the covert communication system.

The error correction experiments were once again conducted in the MATLAB programming environment to leverage the work previously conducted in symbol sequence error detection. In this round of experimentation, a series of test cases (table 4.4) were derived from the expected types of symbol errors as articulated in the fifth error correction design assumption on page 45. These symbol error tests would once again be placed in different positions within an otherwise error-free sequence of legitimate symbols and the error correction capability would attempt to detect and correct for each error type. The following subsections will compare the performance of the error correction methodology as a function of the employed trellis code. Given the two remaining trellis codes being considered for implementation into the covert communication system, two sets of results will be presented.

Table 4.4: Error Correction Tests

Series	Error Type	Test
1	Error Free	$\boldsymbol{abcd}$
2	Single Insertion	$\boldsymbol{ab\bar{c}c}$
3	Single Insertion	$\boldsymbol{a\bar{b}bc}$
4	Double Insertion	$\boldsymbol{ab\bar{c}\bar{c}}$
5	Double Insertion	$\boldsymbol{a\bar{b}b\bar{c}}$
6	Double Insertion	$\boldsymbol{a\bar{b}\bar{b}b}$
7	Mixed Insertion/Deletion	$\boldsymbol{ab\bar{c}d}$
8	Mixed Insertion/Deletion	$\boldsymbol{a\bar{b}bd}$
9	Mixed Insertion/Deletion	$\boldsymbol{ac\bar{d}d}$
10	Mixed Insertion/Deletion	$\boldsymbol{a\bar{b}cd}$
11	Single Deletion	$\boldsymbol{abde}$
12	Single Deletion	$\boldsymbol{acde}$

4.3.1 Single and Double Insertion Correction Experiments

In these series of experiments, the error correction methodology was integrated with one of the two trellis codes and subjected to a series of test vectors to verify the performance of the resulting error correcting capability. The test vectors consisted of several encoded symbol sequence including an error free sequence as well as sequences that included single and double insertion errors.

The insertion error sequences, similar to the error detection experiments, were created by modifying a sequence of error-free encoded symbols by inserting either one or two symbols in accordance with the particular test case (ex. $ab\bar{c}c$). Each individual test case was repeated throughout the length of the error-free encoded symbol sequence until the end of the sequence was achieved. In addition, each test case would be repeated for all possible output symbols for the particular code. The resulting corrected/estimated sequence of encoded symbols from each test vector were then compared with the original error-free sequence to determine if the error correction had effectively removed the fabricated error. If the estimated sequence did not match the original error-free sequence, the fault was recorded for the particular test case. The results of the single and double insertion correction experiments can be seen in table 4.5.

Table 4.5: Single and Double Insertion Correction Experiments

Series	Error Type	Test	Errors using (4,1,3)	Errors using (5,1,4)
1	Error Free	$\boldsymbol{abcd}$	0	0
2	Single Insertion	$\boldsymbol{ab\bar{c}c}$	0	0
3	Single Insertion	$\boldsymbol{a\bar{b}bc}$	0	0
4	Double Insertion	$\boldsymbol{ab\bar{c}\bar{c}}$	8	0
5	Double Insertion	$\boldsymbol{a\bar{b}b\bar{c}}$	33	122
6	Double Insertion	$\boldsymbol{a\bar{b}\bar{b}b}$	8	0
	Total Errors/Vectors		49/14401	122/56449

As expected, both trellis codes when combined with the error correction methodology, were able to inspect and properly correct all the error-free and single insertion test vectors. On the other hand, there were some significant differences between the performance of the two codes when they were faced with double insertion test vectors. The (5,1,4) trellis code out-performed the (4,1,3) trellis code in the 4 - 6 test series. Although it would appear that the (5,1,4) code had greater difficulty with the series 5 test vectors, it still achieved a failure rate of only (122/56449) or 0.216%. While the (4,1,3) code performed marginally worse with the series 5 test case by experiencing a (33/14401) or 0.229% failure rate. Therefore, in the case of this experiment, the (5,1,4) code is considered superior at resolving double insertion errors.

4.3.2 Single Deletion and Mixed Error Correction Experiments

These series of experiments continued the effort to verify the performance of the error correcting capability of the wireless covert communication system by subjecting the trellis codes along with error correction methodology to a new series of test vectors that included single deletions and mixed deletion and insertion tests.

The testing methodology for these experiments remained the same as the previous insertion error tests except the test vectors now included a symbol deletion as part of each test. For example in series 9 test vectors, the test sequence deletes a symbol (b)

as well as adds a symbol insertion ($\bar{d}$) to an error free sequence of *abcd* to eventually become $ac\bar{d}d$. If the error correcting capability could not properly resolve the induced errors for a series of test vectors, a fault would again be recorded for that particular case. The results of the single deletion and mixed error correction experiments can be seen in table 4.6.

Table 4.6: Single Deletion and Mixed Error Correction Experiments

Series	Error Type	Test	Errors using (4,1,3)	Errors using (5,1,4)
7	Mixed Insertion/Deletion	$\boldsymbol{ab\bar{c}d}$	8	9
8	Mixed Insertion/Deletion	$\boldsymbol{a\bar{b}bd}$	31	49
9	Mixed Insertion/Deletion	$\boldsymbol{ac\bar{d}d}$	57	54
10	Mixed Insertion/Deletion	$\boldsymbol{a\bar{b}cd}$	9	9
11	Single Deletion	***abde***	0	0
12	Single Deletion	***acde***	0	0
	Total Errors/Vectors		105/1152	121/2304

Although the error correcting faults were higher in this round of experimentation, the (5,1,4) trellis code once again out-performed its rival trellis code with a 5.252% fault rate (121/2304) as compared to a 9.115% fault rate (105/1152) for the (4,1,3) trellis code. Both codes were able to resolve single deletions without fault, but both encountered difficulty when resolving mixed insertion/deletion scenarios. Combining these results with the previous error correction experiments, the (5,1,4) trellis code exhibited an error correction fault tolerance of 243 faults/58753 test vectors or 0.414%. The (4,1,3) trellis code, by comparison, exhibited a error correcting fault tolerance of 154 faults/15553 test vectors or 0.990% which was over double that of the (5,1,4) trellis code. Given the error correcting performance of the (5,1,4) trellis code when integrated with the error correcting methodology, the (5,1,4) trellis code was selected as the code to be implemented in the wireless covert communication system.

In an attempt to interpret the results of the error correcting experiments and understand why the (5,1,4) code performed better than the (4,1,3) code we must look to the state structures of both codes for answers (figures 4.1, 4.2).

Knowing that the error correction methodology attempts to correct for potential errors within a sequence of symbols by suggesting alternative sequences which incor-

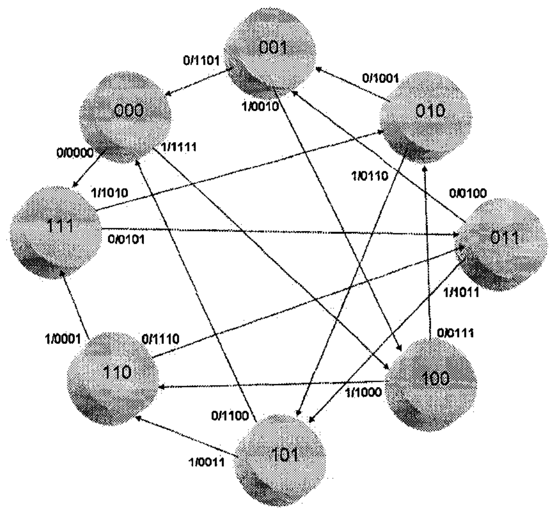

Figure 4.1: (4,1,3) Trellis code from [44]

porate symbol deletions as well as fabricated insertions to couter the expected noise and frame loss of a wireless channel, the process of making these suggestions assumes that the state structure will be diverse enough so that an alternate path will not be selected as part of the suggestion thus creating false positive and bogus data.

One reason the trellis codes performed better than the traditional convolutional codes in detecting symbol insertions was that memory loss errors did not exist in the structure and lengths of equal/unequal and alternate path errors were significantly large. Thus the introduction of one inserted symbol could not be confused for an alternate path through the state machine. Similarly, although the (4,1,3) code had increased path distances between equal/unequal alternate pathways, the increases were not sufficient. Resolving two sequence errors by suggesting two more errors often generated a sequence with four errors that selected an alternate path and created a false positive.

The (5,1,4) trellis code was developed to increase the alternate path lengths through the employment of a 3-dimensional structure [44]. The state machine is composed of two rings of eight states that are connected by a few state transitions. This serves to isolate the majority of state transitions to one of the two rings. It is through this isolation that the majority of sequence errors can be detected and

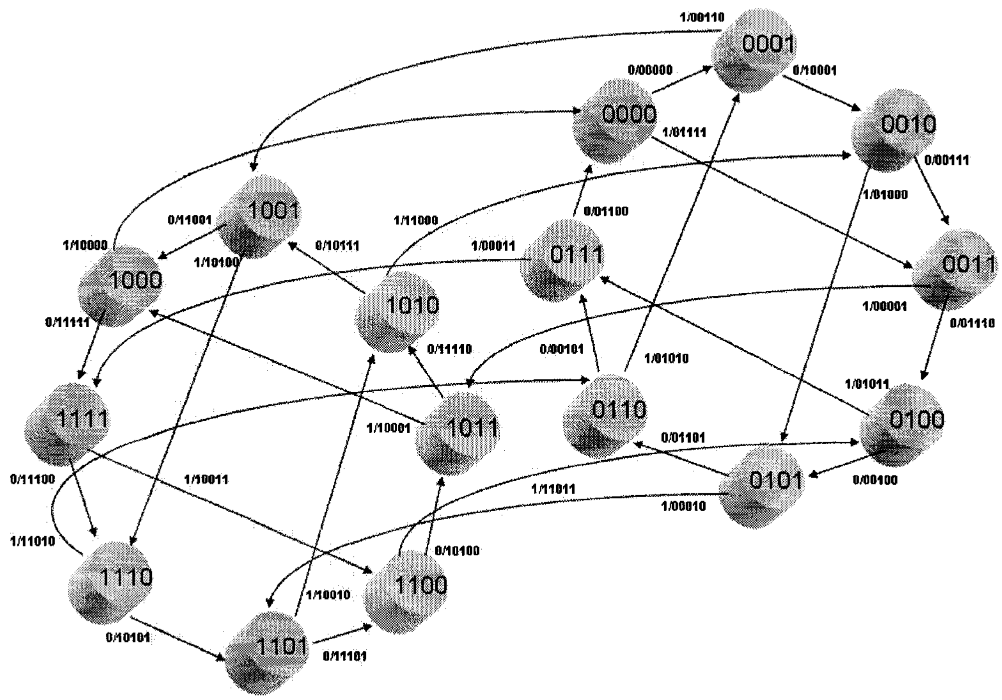

Figure 4.2: (5,1,4) Trellis code from [44]

resolved. Although the (5,1,4) trellis code can fail due to the accumulation of four sequence errors (2 x channel and 2 x correction suggestions), these cases occur less often due to the design of the state structure.

4.4 Wireless Channel Characterization

The final stage of experimentation before the implementation of the wireless covert communication system involved observing the wireless channel and attempting to characterize typical noise activity within the channel. By observing the behaviour of the 802.11 MAC data frames with respect to activity on the compromised station, the covert communication system can be implemented in a fashion that exploits the regular wireless network traffic for concealment as well as taking advantage of 802.11 frame encapsulation behaviour of higher-level protocol activity to communicate the covert message.

4.4.1 Noise Generation Experiment

The purpose of the noise generation experiment was to generate wireless network traffic from a wireless computer in a fashion that is typical of an average Internet user. Since the covert communication system will be using changes in the encrypted 802.11 MAC data frame length as a means to communicate a covert message, we are interested in the encrypted 802.11 MAC data frame activity originating from the wireless computer and directed at the access point. More specifically, we will observe the variations in the MAC data frame lengths that occur based on typical outbound Internet activity from the wireless computer.

This experiment was composed of two wireless computers: the first computer conducted several Internet based activates that would specifically generate outbound data traffic while the second computer eavesdropped on the first computer's wireless network activity. The 28 minute experiment saw the target wireless computer perform several activities such as surf web pages, search for information from an online search tool, send an e-mail with attachment from a web based mail provider as well as communicate with an instant messaging tool to chat and transfer a file. Concurrently, the eavesdropping computer covertly captured all the 802.11 frame traffic during the test period with a wireless sniffing tool Kismet. A histogram of the observed 802.11 MAC data frame traffic by frame lengths can be seen in figure 4.3

The histogram at figure 4.3 shows 802.11 MAC data frames ranging in size from 80 to 1544 bytes. Despite the appearance of activity across the whole spectrum of lengths, there are significant activity spikes at specific lengths. The spikes of activity for a few lengths were so large as to make it difficult to visualize the activity in the remainder of the spectrum of lengths. For this reason, some lengths were manually reduced to an artificial level of 40 occurrences in order to bring up the noise floor in the remainder of the spectrum. Most notably, the frame length of 92 bytes was reduced from 1827 occurrences to 40 for visual purposes. Other spikes included 154 occurrences of 100 byte frames and 299 occurrences of 1544 byte frames. The

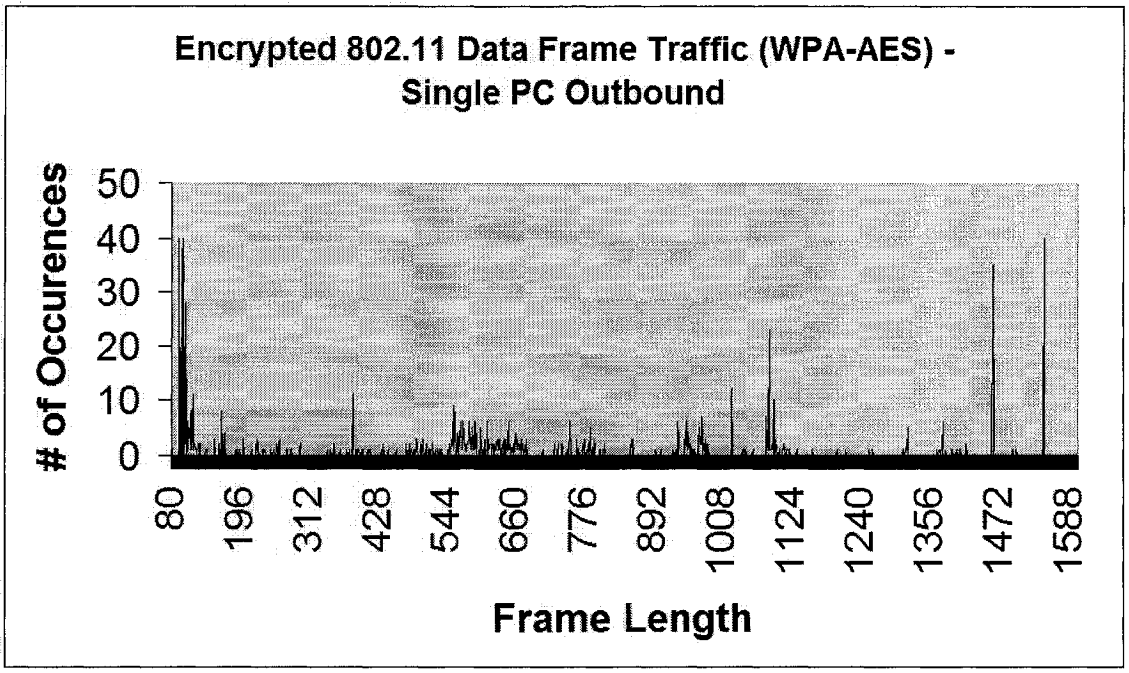

Figure 4.3: Noise Generation - Histogram of 802.11 MAC Data Frames

distribution of the observed MAC data frames are further represented in figure 4.4.

Since the wireless covert communication system is to operate in the presence of network traffic noise, the distribution of the MAC data frames by length as shown in figures 4.3 and 4.4 can be used to select the alphabet of coded symbols used to modulate the data packet lengths for use in the system. Although, despite the desire to operate amongst the noise of regular network traffic, care should be taken not to select overly used MAC data frame lengths. One can observe from figure 4.4 the distribution of observed MAC data frames that almost 70% of the activity was centred around the data frames with smaller lengths. In addition in figure 4.3, the three manually reduced frame lengths of 92, 100 and 1544 bytes composed 72.9% of the captured frame traffic. Although the covert communication system is to co-exist in the same channel as other legitimate traffic, these three traffic spikes would violate the error correction design assumptions from page 45. Therefore, the remaining frame lengths that exhibited moderate noise activity would be chosen for use with the covert communication system.

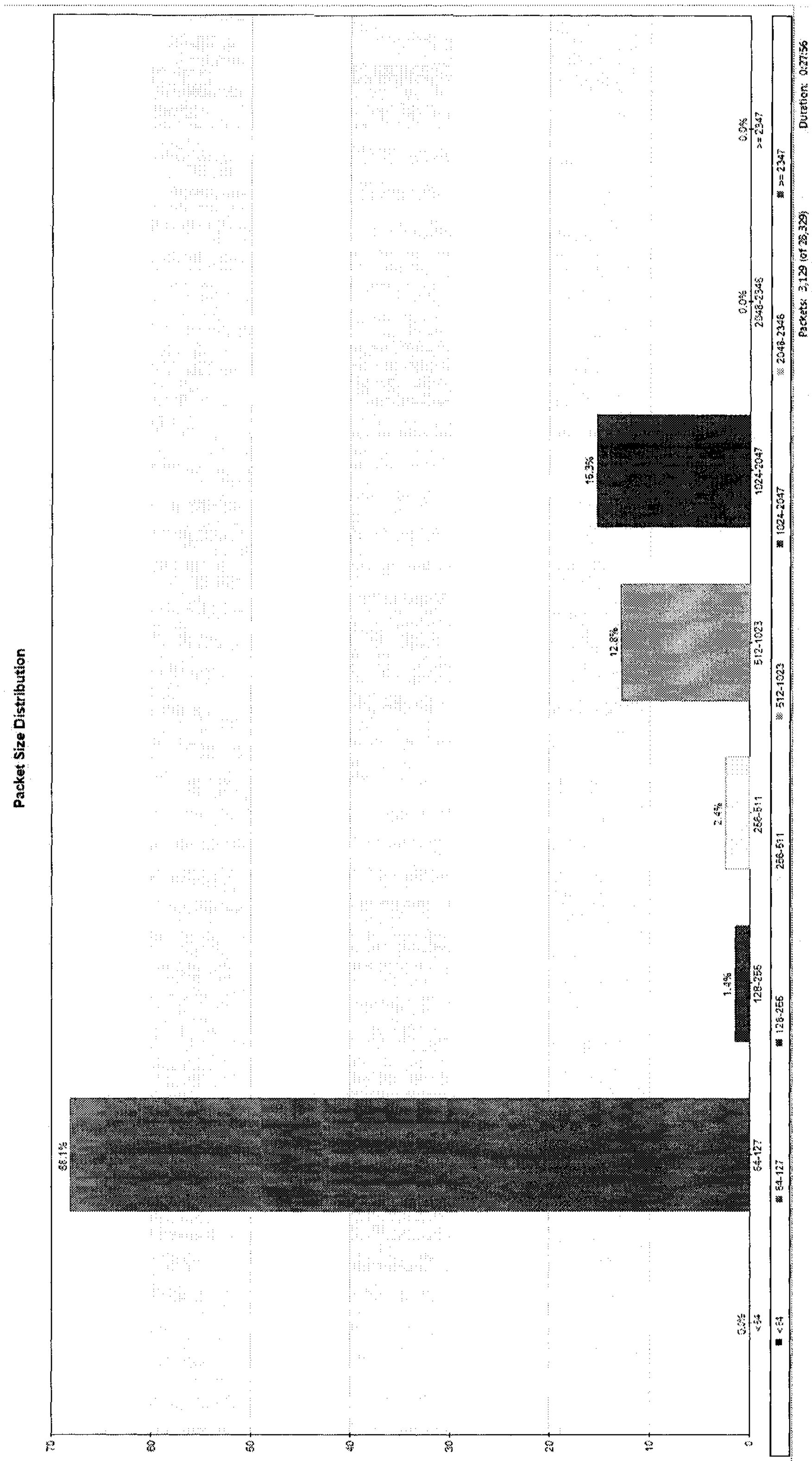

Figure 4.4: Noise Generation - MAC Data Frame Distribution

4.4.2 UDP Packet Size vs MAC Frame Size Experiment

This experiment was developed to determine the relationship between the size of an unencrypted UDP echo packet and the size of the same packet encapsulated inside an encrypted 802.11 MAC data frame. By conducting this experiment, it should become apparent if there is a direct relationship between the two sizes or if there are variations in the encapsulation and encryption process which will cause the transformation to be non-uniform over a spectrum of different sizes.

Using Microsoft Visual C++, a UDP echo packet was created with a data field filled with ASCII text and sent to the IP address of the wireless access point. Knowing that a UDP packet header is consistently 42 bytes in length, data fields ranging in size from 1 to 1460 bytes would be created and sent with the UDP echo request. The resulting series of transmitted packets would then be captured by an eavesdropping computer to observe the effect on the 802.11 encrypted MAC data frame lengths. In addition, one could observe if there are any significant changes in data frame behaviour as the Microsoft Ethernet MTU value of 1500 bytes is reached. The results of this experiment can be seen in figure 4.5.

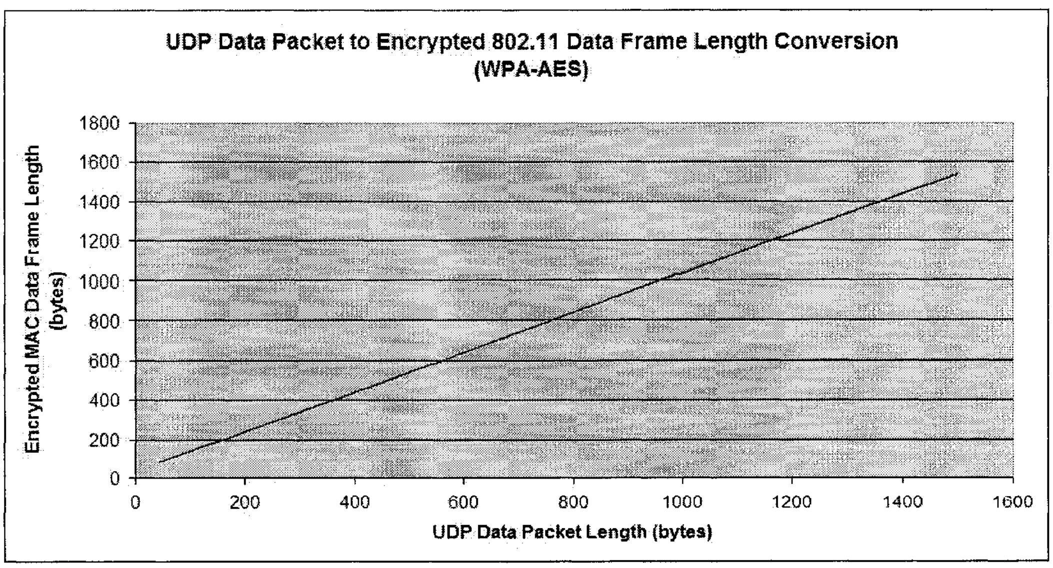

Figure 4.5: Observed Changes in MAC Frame Size

From figure 4.5 we can see that there is a direct relationship between a UDP data packet and the encrypted encapsulated 802.11 MAC data frame of that same packet. The direct relationship indicates a consistent increase of 38 bytes from a UDP echo request to the corresponding encrypted MAC data frame observed by the covert receiver. Accounting for the 42 byte header of the UDP packet, we can then devise a 80 byte direct relationship between the UDP data payload and the observed encrypted MAC data frame. This is of significance since the covert communication system can now modulate the size of UDP packets with specific encoded symbols and then demodulate the received 802.11 MAC data frames with a high degree of confidence that the received frames represent encoded symbols sent by the software Trojan.

4.5 Wireless Covert Channel Testing

Given the previous activities that characterized and tested individual components of the designed system, the final stage of this research combined the components into a functional wireless covert communication system. The covert communication system was able to take ASCII text from a compromised station and beacon that information over the wireless channel. An eavesdropping computer listening to the channel was then able to capture the modulated covert message in the presence of noise and effectively reproduce the original ASCII message.

The successful implementation of the covert communication system represents another achievement in the overall goals of this research. The following subsections outline the testing activities conducted with the final covert communication system in order to prove its functionality as well as to investigate the systems performance characteristics within a noisy wireless channel.

4.5.1 Live Testing Experiment

The live testing experiment was the first end-to-end test of the wireless covert communication system in which a phase of ASCII text (*The quick brown fox jumped over the lazy dog's back!*) was sent from the software Trojan over the encrypted wireless channel and then received by the covert receiver capturing the traffic from that channel.

Knowing the typical 802.11 MAC data frame lengths and frequencies of occurrence from previous experimentation, the covert communication system was configured to employ encrypted MAC frame lengths that were previously observed in figure 4.3 and thought to be inactive. By employing frame lengths that did not frequently occur, the covert system would first function with very little channel noise in order to prove its basic functionality. Using these symbols, the covert message was beaconed for six separate iterations and the covert receiver used to decode all six versions of the message to observe any irregularities. A snippet of the decoded message can be seen in figure 4.6.

```
Recovered Covert Text Message:
     5

This message contains the following number of symbols:
   426

The quick brown fox jumped over the lazy dog's back!

Recovered Covert Text Message:
     6

This message contains the following number of symbols:
   416

The quick brown fox jumped over the lazy dog's back!

>>
```

Figure 4.6: Live Test Results in Minimum Noise

Knowing that the error-free encoded symbol sequence of the covert message contained 424 encoded symbols, one can see if the received sequence of symbols from each message iteration contained either more or less symbols (noise or signal loss) as compared to the sent massage. One can see from figure 4.6 that despite the addition of two symbols in message #5 and the loss of 8 symbols in message #6, both messages decoded properly and without corruption. In fact, all six iterations of the beaconed message decoded without error. The results of this experiment provide initial proof that the implemented wireless covert communication system can operate effectively in the presence of minimal amounts of channel error.

4.5.2 Interlacing Experiments

Knowing that the implemented covert communication system can effectively operate in the presence of minimal channel error, the next stage of experimentation would attempt to explore the system's capability when faced with typical channel noise. Instead of re-conducting live capture experiments that contained noise, the typical channel noise previously presented on page 69 would be used to cause insertion errors in the covert message. Observing from previous experimentation the typical distribution of encrypted MAC data frame activity by length, frame lengths that exhibited moderate noise activity were chosen for use with the covert communication system as the new set of modulated encoded symbols.

4.5.2.1 Experiment #1

Employing the MATLAB programming environment, an error-free sequence of encoded symbols were created from the *quick brown fox* test phrase. Each encoded symbol was then assigned a relative transmission time of 28.3ms since this was the average relative time between transmitted encoded symbols observed during the previous live test experiment. The sequence of error-free encoded symbols was then interlaced with the previously captured noise based on the relative receipt times associated with each sequence. Since the noise capture experiment lasted for approximately 28 minutes, the interlacing of the encoded symbols sequence was repeated 144 times to simulate the beaconing of the message. Consistent with the actual operation of the software Trojan, a framing symbol was also used in the interlacing process to indicate the beginning of each iteration of the beaconed message.

Of the 144 messages that were interlaced with the noise of a typical channel, there were none that decoded improperly. The message which encountered the most severe channel noise is shown in figure 4.7 with a total of 457 symbols in the received message. This represents an increase of 33 noise symbols over the original error-free size of 424 symbols and a Signal to Noise Ratio (SNR) of 12.84.

Recovered Covert Text Message:
111

This message contains the following number of symbols:
457

The quick brown fox jumped over the lazy dog's back!

Recovered Covert Text Message:
112

This message contains the following number of symbols:
441

The quick brown fox jumped over the lazy dog's back!

Figure 4.7: Test Results in Noise

4.5.2.2 Experiment #2

Although this experiment provided some insight into the capability of the covert communication system to operate in the presence of noise, it did not expose the level of noise required in the channel to cause the decoder to fail. Therefore, another interlacing experiment was conducted to test the limits of the wireless covert communication system's performance in the presence of noise. In the first interlacing experiment, all the encoded symbols were sent with the minimum transmission interval of 28.3ms. In the next interlacing experiment, the transmission interval between encoded symbols would gradually increase to allow more noise between symbol transmissions.

Similar to the first interlacing experiment, the encoded symbols would be first transmitted with a transmission interval of 28.3ms for the entire length of the noise capture. The test would then be repeated with a transmission interval increased by the minimum interval to 56.6ms. This repetitive test would continue until only one message could be transmitted in the 28 minutes period of the noise capture.

Unlike the first interlace experiment, the encoded symbols would be created with a string of ASCII text only 63 characters long. This represents the maximum size of a WPA wireless security passphrase and the focus of the *working scenario*. Once the second interlacing experiment finished transmitting one single 63 ASCII character message over the 28 minute period, the phrase would be reduced by one character and the tests repeated. This reduction in the subject ASCII phase would continue until only an 8 ASCII characters were tested. 8 ASCII characters represent the minimum possible wireless security password for WPA encryption.

The resulting combinations of noise and message would be decoded and compared with original message for potential errors. If the estimated message was the same as the original text, the decode was successful. Otherwise, if there was an error in the decode, the recovery was considered a failure and recorded accordingly. In addition to the recording of success or failure of each decode, the transmission interval and the error-free encoded sequence length for each message were also recorded.

Given the recoded metrics of each message decode in the presence of noise, we are able to calculate several different performance measures for the covert communication system. Similar to the first interlace experiment, we can calculate the SNR ratio for each message knowing the encoded symbol sequence lengths before and after passage through the wireless channel. With a SNR metric for every message decode, we can then calculate the reliability for the covert communication system by calculating the percentage of successful message decodes within the bin of observation for a particular SNR. In order to achieve a statistically sound value for the system reliability at a particular SNR, the observation bin must be large enough to contain several hundred samples at a minimum. The wireless covert communication system's reliably as a function of the SNR of the covert message can be seen in figure 4.8.

The traces in figure 4.8 show the different performances characteristics of the covert communication system when decoding small 8 ASCII character messages compared to larger 63 ASCII character messages. In addition, the Average trace represents the overall performance of the system over the range of different message sizes. As expected, the smaller message sizes have a higher probability of successful decode in the presence of noise as compared to larger messages. Since smaller messages have shorter sequence lengths, their exposure to noise in the channel is limited. In addition, any addition of noise will cause a significant change in the SNR ratio due in part to the smaller length of the message. An interesting point to note here is the inability of the covert communication system to achieve 100% decode success for the entire

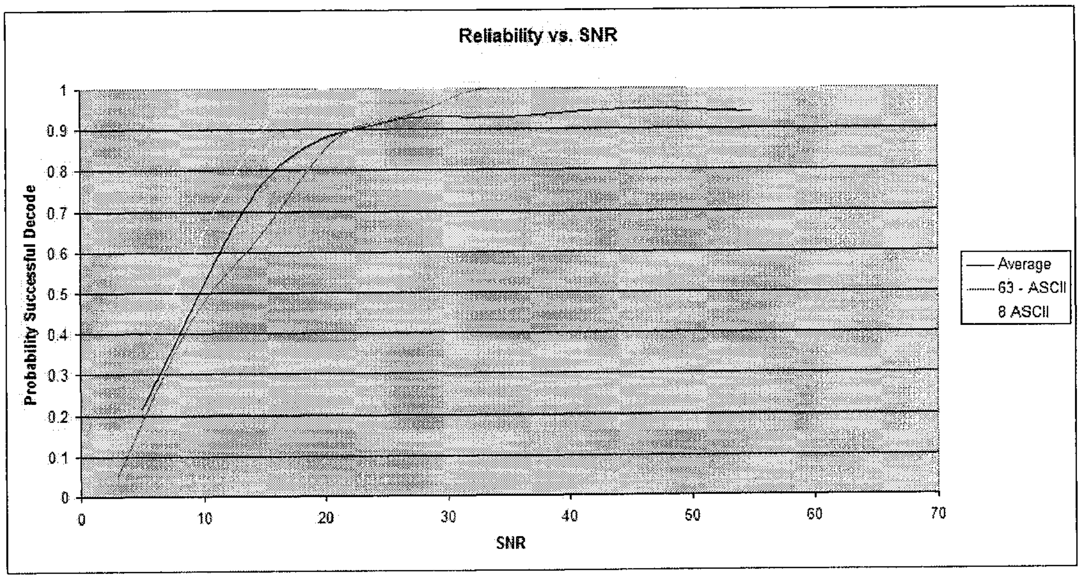

Figure 4.8: Reliability vs SNR

range of message sizes. This would seem to indicate that for larger messages at every SNR value there is always a message that will not decode properly. Although, this artifact should not overshadow the decoder's ability to successfully decode messages with a greater than 90% assurance beyond a SNR of 22.

The next performance measure compares the covert communication system's reliability to decode a message as a function of the transmission bandwidth. In this context, bandwidth is the measure of bits from the original un-encoded binary stream of ASCII text transmitted as a function of time. In the case of the covert communication system, each encoded symbol represents one bit of information from the original message. Therefore, to calculate bandwidth, it is the number of symbols in the encoded sequence divided by the sum of the transmission intervals between symbols. Similar to the last graph of reliability, each point of reliability is a percentage of successful decodes within a bin of observation for a particular bandwidth value. The wireless covert communication system's reliably as a function of bandwidth can be seen in figure 4.9.

Upon review of the calculated results, there would appear to be two interesting points worth noting from the reliability trace in figure 4.9. Firstly, for the conditions

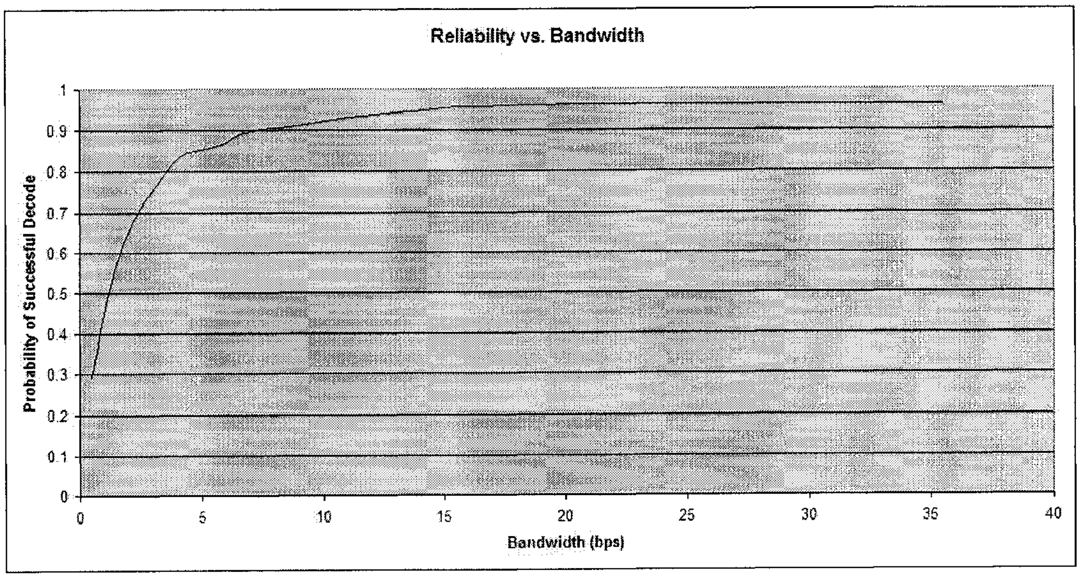

Figure 4.9: Reliability vs Bandwidth

of the experiment the covert communication system can consistently decode messages from a noisy channel with a success rate in excess of 80% for messages transmitted as slow as 4 bps. This would mean that if a pause is implemented in between transmissions of the modulated symbols, it should not by greater than 250ms in order to be effectively decoded without fault given the current noise profile. Although, since messages are beaconed across the wireless channel, the aggregate information obtained from possibly corrupted data should be enough to reconstruct the intended message. Therefore, the rate could even be lowered to one bit per second and achieve 30% reliability and still reconstruct the message from partial fragments of other corrupted decodes of the same message.

And secondly, the covert communication system could still not achieve 100% decode success for messages that were transmitted with the minimum transmission interval. It would appear that even given the increased transmission rates, some messages did not decode properly due to the utilized noise profile for the transmission channel.

Finally, we look at the relationship between SNR as a function of the bandwidth. Similar to the previous two graphs, each point of bandwidth is an average of all the observed bandwidths within a bin of observation for a particular SNR value. The wireless covert communication system's bandwidth as a function of SNR can be seen in figure 4.10.

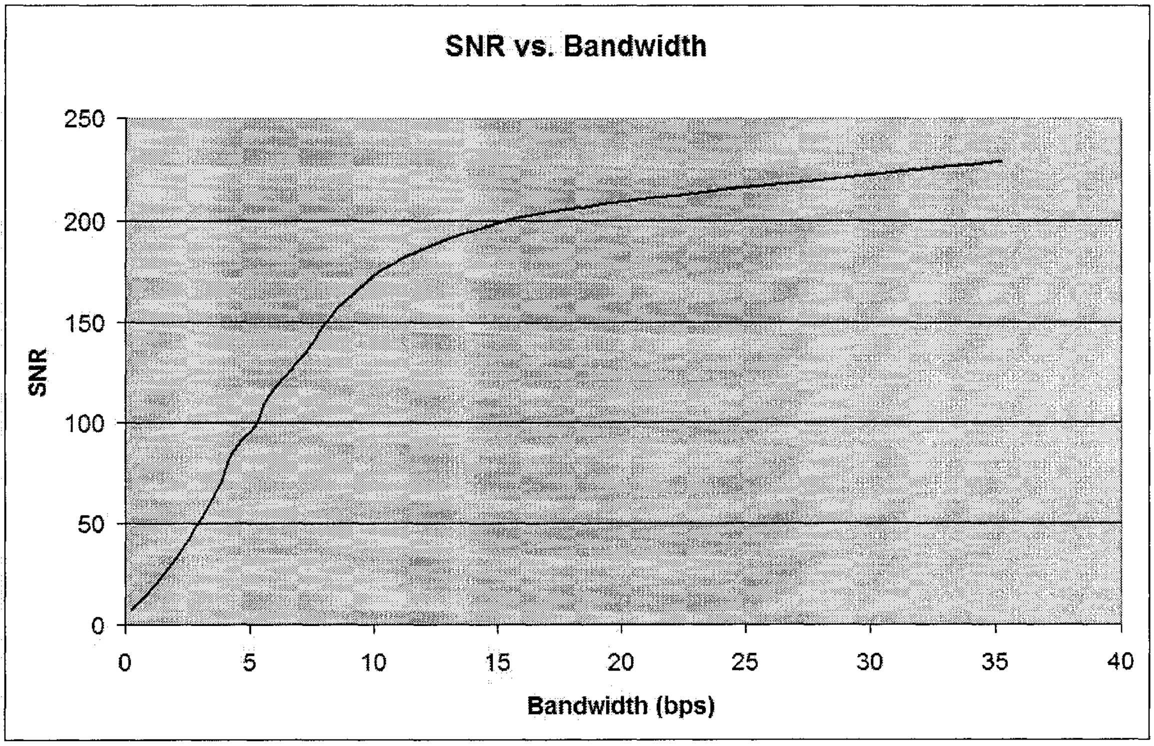

Figure 4.10: SNR vs Bandwidth

From the trace in figure 4.10 we can see that as the bandwidth of the covert message increased, the SNR associated with that particular bandwidth did not increase that significantly past 10bps. This relationship could be related to the effects of the larger length covert messages transmitted through the noisy channel. In order for a large message to be transmitted through the channel and retain a high SNR it must be done with a higher bit rate or very small inter-symbol transmission interval resulting in a higher bandwidth measure. Therefore, for significant increases in the bandwidth from 15 to 35 bps, one can achieve only moderate gain in SNR.

Furthermore, it is expected that one also exposes the covert activity to detection by increasing the bandwidth of communication. In this case, a balance must be struck in which a reliable bandwidth value is chosen (ex. 2 bps) which also allows for a reliable SNR (ex. 12) and thus ensures a higher probability of a successful decode (ex. 60%) of the covert message as seen in figure 4.11. Given the apparent interrelationship between Reliability, Bandwidth and SNR [44], one can adjust the design of the covert communication system by selecting a transmission bandwidth that will ensure a successful decode of a covert message despite the presence of noise in the channel.

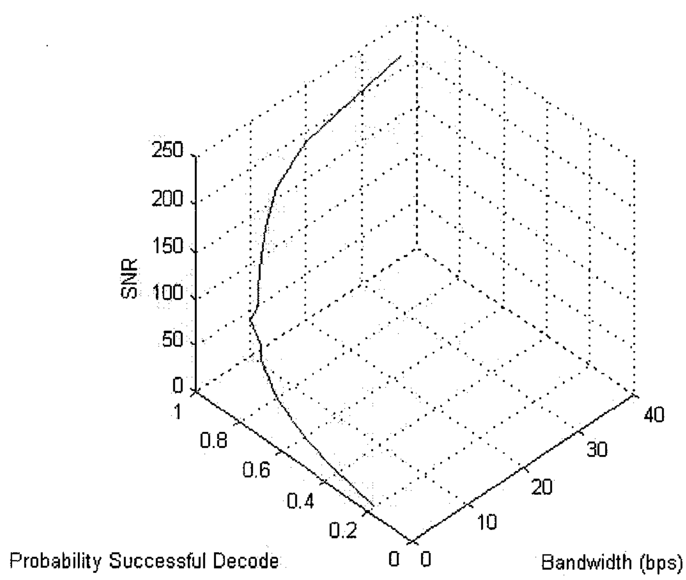

Figure 4.11: Reliability vs BW vs SNR

4.6 Summary

This chapter outlined the research activities that utilized the unique combination of ideas and system design concepts presented in Chapter 3 to implement a working wireless covert communication system. Furthermore, this chapter has presented the performance test results of the of the wireless covert communication system in fulfilment of the research goal presented in the first introductory chapter of this thesis.

The material presented in this thesis has introduced the concept of covert channels and the threat they represent to secure wireless networking. Given the possibility that a potential security hole in wireless networking could be exploited by nefarious individuals, this research set out to prove it is possible to implement a functioning wireless covert communication system by combining aspects of coding theory with a new inductive approach to error correcting. Employing a standard model for digital communication system design, this research integrated the new ideas into a functioning wireless covert communication system. Now that the principle research goal has been achieved, the concluding fifth chapter that follows will discuss the system's implementation and performance, areas of future research and some concluding remarks.

Chapter 5

Conclusion and Future Directions

5.1 Introduction

The previous chapters of this thesis presented the motivation and significance of researching covert channels in secure wireless networks. In addition, this thesis presented several new ideas that applied traditional coding theory concepts to detecting and correcting symbol-level errors typical of a wireless channel environment. The combination of these ideas led to the successful design and implementation of a working covert communication system on a secure wireless network. This final chapter will conclude the thesis with a discussion of the resulting covert communication system's implementation and performance. This last chapter will also provide some thoughts on potential areas of future work and some concluding comments.

5.2 Discussion

5.2.1 Importance of Wireless Covert Communication System

The successful implementation and demonstration of a wireless covert communication system has identified a significant security hole in the WPA authentication and encryption method of the 802.11 security standard. This security hole raises some serious concerns as to the possibility of such a system existing in the real world and the potential to detect such illegal communication activity.

In order for a covert communication system to be successful in the real world, it must operate in a concealed manor and be able to effectively pass information across the discrete channel with a high assurance. The wireless covert communication system that was developed as part of this research attempted to implement these characteristics in order to demonstrate the vulnerability of secure wireless networks to covert channels. The following sub-sections will explore the implementations of these characteristics and their significance to the overall operation of the wireless covert communication system.

5.2.1.1 Covertness

The implementation of covertness into the wireless covert communication system assumed two forms within the design, namely internal and external. The wireless covert communication system operated on two different sides of the security policy set forth by the WPA security standard for the 802.11 infrastructure, the un-encrypted (clear) network communications generated by the compromised station and the encrypted wireless link traffic. Therefore, internal covertness delt with the concealment of the clear network traffic generated by the Software Trojan and the external covertness which delt with the hiding of the network activity associated with the covert message amongst the regularly expected activity of a 802.11 wireless link.

The internal covertness of the wireless covert communication system was attempted through the employment of UDP echo requests. UDP traffic including UDP echo requests occur regularly within networks based on the TCP/IP suite of protocols. Although, this style of covert storage channel could have been achieved through the use of any other protocol that occurs regularly and can implement variable packet lengths. Based on their connection-less nature UDP packets can be transmitted across a network with a minimum of additional protocol interaction. Furthermore, by sending UDP packets only to the wireless gateway, the additional amount of clear network activity was limited to just the wireless link thus avoiding any IDS located further within the wired infrastructure of the network. Therefore, internal covertness was achieved through a combination of compartmentalization and the use of commonly expected network traffic.

Nevertheless, the use of UDP echo requests are not the only approach to producing covert communications. In fact, for this type of storage-based covert channel, any type of network packet could be employed as long as the software Trojan could modulate it's size based on a selected coding strategy. Therefore, there may be other exploits that are less obvious in clear network traffic but similarly produce the desired effect to 802.11 MAC data frames once encapsulated and encrypted. Furthermore, increased covertness could potentially be achieved if the software Trojan was able to modulate the network traffic streams produced by legitimate processes. By enabling the software Trojan to throttle the legitimate or "natural" occurring outbound wireless traffic, the activity of the covert communication system would not only be difficult to detect in the clear (internal covertness), but would almost be impossible to notice once transformed into encrypted 802.11 MAC data frames (external covertness).

The wireless covert communication system achieved external covertness by modulating the encoded symbols to use packet sizes that would result in an encrypted 802.11 MAC data frame of similar sizes to those regularly expected as MAC data frames. By employing frame lengths that mimicked regular wireless traffic, the communication activity of the Software Trojan would be concealed as normal wireless traffic. A level of measure for this aspect of covertness was the SNR values obtained through experimentation in the previous chapter. The lower the SNR value, the more concealed the covert message was amongst the noise floor of the regularly expected traffic. The implemented covert channel was able to reliably produce the covert message with a 50% probability with SNR values as low as 8 for smaller messages. Even when the reliability descended below 50%, it is expected that the wireless covert channel could recover the message from the fragments of the beaconing message. Given this level of external covertness, the network activity generated by the software Trojan would be very difficult to detect by just looking at a capture of the 802.11 MAC data frames.

5.2.1.2 Bandwidth

The initial implementation of the wireless covert communication system modulated and transmitted encoded symbols continuously without delay to achieve a maximum bandwidth of approximately 35 bps. Although this level of bandwidth is desirable for the passage of large passphrases, the network activity at 35 bps is so significant that it can no longer be concealed by regular activity and the covertness of the system could be compromised. In addition, by operating the Software Trojan at such a high rate, a denial service attack situation is created in which the access point is flooded with UDP echo request packets. To reduce the network activity generated by the Software Trojan for the purposes of remaining internally and externally covert, an inter-packet delay was added between modulated encoded symbols. Thus, the final implementation of the wireless covert communication system incorporated the ability for the Software Trojan to throttle the modulation of the encoded sequence and induce a delay between modulated symbol transmissions.

The results obtained from Chapter 4 indicate that the bandwidth of the implemented covert communication system could be reduced to 2 bps (500ms inter-packet delay) for a SNR of 12 and still achieve an 60% reliability that the message will decode properly. This noteworthy result indicates the implemented wireless covert communication system can conceal its activity by reducing its bandwidth and still be effective in passing the covert message. At this rate the maximum size WPA passphrase of 63 ASCII characters which translates into 504 encoded symbols would require 252 seconds or 4.2 minutes to pass from the Software Trojan to the Covert Receiver. Given the noise generation experiment in Chapter 4 lasted almost 30 minutes, a five minute transmission of the covert message would produce six iterations of the same message and facilitate partial message reconstruction if required. A wireless traffic capture of 30 minutes is considered well within the capability of an attacker conducting CNI operations and thus reiterates the significance of the wireless covert communication system's performance.

Another aspect that could affect bandwidth would be to modify the code rate $\boldsymbol{R}$ ($\boldsymbol{R} = k/n$ where $k \leq n$) of the coding strategy implemented by the system to translate one input symbol to one output symbol. In the current implementation of the covert communication system, decreasing the code rate by increasing the length of the output symbol (n) of the trellis code would not decrease the systems bandwidth. Since each UDP echo request represents one encoded symbol, increasing n would not change the bandwidth of the system but only increase the number of possible encoded symbols to be used for FEC. On the other hand, by increasing the length of the input symbol (k) one could directly increase the bandwidth of the covert communication system.

Since the bandwidth of the covert communication system was measured in terms of the transmission of covert message bits per second, increasing k also increases the transmission of the covert message to k bits per encoded output symbol. Therefore, one could capitalize on potential gains in FEC by significantly increasing n while at the same time moderately increasing k to achieve a desired increase in bandwidth. Combining these changes to the values of k and n in this context would lower the code rate $\boldsymbol{R}$ while at the same time increase the effectiveness of the wireless covert communication system in terms of FEC and bandwidth.

5.2.1.3 Reliability

The implemented wireless covert communication system proved to be extremely reliable as a method to pass private information from the Software Trojan to the Covert Receiver despite WPA AES encryption and the presence of other legitimate network traffic. As seen in Chapter 4, the reliability of the system to successfully decode a covert message did not start significantly decreasing until the transmission bandwidth was reduced to a value below 4 bps given the employed noise profile.

Although, the ability for the covert communication system to reliably decode a covert message depends on the noise conforming to a pattern consistent with the design assumptions made on page 45. Given the noise profile of the typical noise generation experiment of Chapter 4, we can see that network traffic is bursty by nature and occurs with spikes of activity directly related to specific activities or processes (ex. web surfing versus file transfers). When the covert message was exposed to bursty activity, the design assumption would be violated and the decoder would most likely produce a faulty result. Therefore the selection of the encoded symbol lengths from the spectrum of 802.11 data frame lengths is important to ensure a highly reliable decode of a covert message yet remain concealed by the noise within the channel.

Finally, a major contributor to the reliability of the wireless covert channel was the development of an effective error detection and correction strategy. Despite the great performance of the selected trellis code combined with the error correction methodology proposed in Chapter 3, the decoder was limited in its ability to effectively correct for all possible errors due to increased noise levels within the channel. As previously mentioned, if the channel noise exceeded the design assumption of a maximum of two errors within a four symbol window the methodology would fail and the decode would be faulty. However, a typical wireless channel is rarely active with significant traffic volumes for any prolonged period of time. More specifically, the volumes of 802.11 MAC data frames originating from the compromised station and destined for the wireless access point were noticeably low during idle user periods. The wireless channel only experienced brief high intensity periods during very specific user initiated communication activities. Therefore, if the wireless covert communication system was implemented on a typical wireless station, it would experience a reliability well in excess of 80% due to prolonged periods of user inactivity during a regular daily cycle.

5.2.2 Impact of Research

The successful implementation of the wireless covert communication system serves as another example that covert channels are possible and can exist in a real-world context. Furthermore, this research has expanded the knowledge of network-based covert communications over noisy channels by implementing a successful error correction and detection methodology. Moreover, by implementing a functional wireless covert communication system, the performance metrics obtained during experimentation will serve to assist in the future research of this topic area as a baseline measure.

In addition, the threat of covert channels to secure wireless networks is real and potentially very costly to organizations who are deciding to implement wireless networks for the passage of private information. The gained mobility at the expense of exposure of sensitive information may be too great a cost for some to tolerate. Given the identified security hole in the WPA authentication and encryption methods of the 802.11 security standard through the application of covert channels, organizations should review their security policies and practises to defend against the implantation of a software Trojan as well a monitoring for irregular network activity from unauthorized processes. This research has shown that with a little social engineering, a strong cipher will not protect against intrusions by nefarious individuals seeking information.

5.2.3 Areas for Future Research

With the successful demonstration of 802.11 wireless network's vulnerability to covert channels, several areas of possible future research include investigations into improving the capability of the software Trojan, enhancing the capability of the covert receiver to decode messages in the presence of increased noise as well as defending against covert channels.

It was assumed that the private information transmitted by the software Trojan was readily available on the compromised computer. For demonstration purposes, only user keyboard input was implemented in the current design of the wireless software Trojan. Future research could investigate other ways to extract private information from either the system registry or other input devices in order to compromise network security.

In addition, the software Trojan in this research employed UDP echo requests as a mechanism to signal a message to the covert receiver. Although the UDP echo request was effective in this demonstrative context, it may not be the most covert exploit in comparison to other legitimate Internet activity. Another area of future research could pursue developing an exploit for the Software Trojan that more effectively conceals the beaconing activity within regularly expected network traffic. This would entail the development of other timing-based or more effective storage-based covert channels who's regular activity would be harder to detect from legitimate traffic. Examples of new storage based covert channels could include HTTP request packets with variable sizes to the creation of multiple mail messages of variable sizes using the Simple Mail Transfer Protocol (SMTP).

The Covert Receiver employed specific methods for detecting as well as correcting errors using a trellis code as the basis of the coding strategy. Further research to enhance the capability of the covert receiver could include the development of a more sophisticated method for coding the covert information to recover from greater numbers of channel errors. In addition, this newer coding scheme may also require the development of a decoder that is not based on a Viterbi algorithm but a different approach to reconstructing the correct encoded symbol sequence based on the selected coding scheme.

Finally, another area of potential research would focus on the detection and defence against such covert channels in secure wireless networks. This work could involve the development of software that would enhance the creation 802.11 MAC data frames to ensure constant frame length or random transmission intervals to protect against storage or timing-based covert channels. In addition, this research could also investigate the detection of the irregular network activities that would produce such communication systems.

5.3 Conclusion

The increased demand for easily configurable, low cost, mobile computing solutions in situations where conventional wired network infrastructures do not exist has fuelled the recent popularity of wireless networking products based on the IEEE 802.11 standard. The employment of wireless networks based on 802.11 technology also introduces a potential security risk to an organization by exposing network traffic to potential attackers anywhere within reception range of the wireless transmission. To defend against eavesdropping, the 802.11 protocol supports the WPA security standard with AES encryption. Despite the security policy introduced by the WPA standard, the radio frequency transmissions from 802.11 secure wireless networks present a shared resource that could be exploited by a covert channel. Covert channels are unexpected and hidden communication paths between two processes that are not permitted to communicate, but do so anyway by affecting a shared resource. By applying the concept of covert channels to a secure wireless environment, one could potentially compromise the security posture of the entire network by leaking out private information such as the wireless security password to an attacker conducting CNI operations.

The principle goal of this research was to design, implement and test a working covert communication system on a secure wireless network. The research goal was achieved through a series of activities that focused on the development of an error detection and correction strategy as well as the design and implementation of a covert communication system based on a digital communication model. Initially, a coding/decoding scheme was developed based on FEC techniques and validated against network traffic typical of a wireless Internet based system. Subsequently, the successful coding/decoding scheme was then incorporated into the design of the covert communication system which existed as two distinct and separate components, namely a software Trojan and covert receiver. The resulting two components of the covert communication system were then implemented using Microsoft Visual C++ and MATLAB respectively to produce a functioning wireless covert communication system. The resulting system was then tested against conditions typical of a wireless channel to demonstrate overall performance characteristics.

The resulting covert communication system was able to pass private network information from a compromised wireless computer to a covert receiver despite the implementation of a secure wireless standards such as WPA. This demonstration of capability by the covert communication system marked the successful achievement of the principle research goal. Furthermore, by demonstrating the vulnerability of wireless networks to covert channels, the research hypothesis outlined in Section 1.4 of this thesis was successfully validated.

The major contributions of this thesis are: 1) The implementation of a convolutional trellis code to provide improved stealth and reliability to a network based covert communication system, 2) The creation of a complete covert communication system on a 802.11 secure wireless network and 3) The identification of a security hole in the WPA authentication and encryption method of the 802.11 security standard by applying the threat of covert channels.

References

[1] DDCEI 3-5-2. An architecture for the deployment of 802.11 wireless LAN in DND, August 2004. version 1.0.

[2] J. Bardwell and D. Akin. *CWNA - Certified Wireless Network Administrator - Official Study Guide.* McGraw-Hill/Osborne, 3rd edition, 2005. ISBN: 0-07-225538-2.

[3] Canonical. Ubuntu 6.10, 2007. Available from `www.ubuntu.com` – URL verified 14 Mar 07.

[4] S. Chauhan. Project report: Analysis and detection of network covert channels. Technical report, Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Maryland, USA, December 2005. Available from `www.cisa.umbc.edu` – URL verified 28 Feb 07.

[5] S. Coll, S.B. Glasser, and J. Tate. Terrorists turn to the web as base of operations. Washington Post, aug 2005. Available from `www.crime-research.org` – URL verified 01 Mar 07.

[6] I.J. Cox, M.L. Miller, J.M.G. Linnartz, and T. Kalker. A review of watermarking principles and practises. *Digital Signal Processing in Multimedia Systems*, pages 461–485, 1999. Available from `www.adastral.ucl.ac.uk` – URL verified 1 Mar 07.

[7] daemon99 and alhambra. Project loki - icmp tunnelling. *Phrack Magazine*, 7(49), August 1996. Available from `www.phrack.org` – URL verified 07 Mar 07.

[8] M.C. Davey and D.J.C. MacKay. Reliable communication over channels with insertions, deletions and substitutions. *IEEE Transactions on Information Theory*, 47(2):687–698, February 2001.

[9] A. Dholakia. *Introduction to Convolutional Codes with applications.* Kluwer Academic Publishers, 1994. ISBN: 0-7923-9467-4.

[10] US DoD. Standard: Department of defence trusted computer system evaluation criteria, December 1985. Available from `http://security.isu.edu` – URL verified 1 Mar 07.

[11] US DoD. Directive 8100.2: Use of commercial wireless devices, services, and technologies in the department of defense (dod) global information grid (gig), April 2004. Available from `www.dtic.mil` – URL verified 1 Mar 07.

[12] M.S. Gast. *802.11 Wireless Networks - The Definitive Guide.* O'Reilly Media, Inc., 2nd edition, Apr 2005. ISBN: 0-596-10052-3.

[13] V.D. Gligor. A guide to understanding covert channel analysis of trusted systems. *National Computer Security Centre (NCSC-TG-030)*, 1(1), November 1993. Available from `www.fas.org` – URL verified 01 Mar 07.

[14] B. Goodwin. Foreign intelligence agents hacking UK businesses, government warns. ComputerWeekly.com, November 2006. Available from `www.computerweekly.com` – URL verified 7 Mar 07.

[15] J. Kelley. Terrorist instructions hidden online. USA Today, February 2001. Available from `www.usatoday.com` – URL verified 1 Mar 07.

[16] R.A. Kemmerer. Shared resource matrix methodology: An approach to identifying storage and timing channels. *ACM Transaction on Computer Systems*, 1(3):256–277, August 1983.

[17] M. Kershaw. Kismet, 2007. Available from `www.kismetwireless.net` – URL verified 14 Mar 07.

[18] S.-O. Krohné and S. Axberg. Internet makes terrorist action easier. Framsyn Magazine, January 2005. Available from `www.foi.se` – URL verified 7 Mar 07.

[19] B.W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973. Available from `http://citeseer.ist.psu.edu` – URL verified 1 Mar 07.

[20] C. Langton. Coding and decoding with convolutional codes. principles of communication tutorial 12., July 1999. Available from `www.complextoreal.com`.

[21] D. Leigh. Capacity of insertion and deletion channels. Technical report, Cavendish Laboratory, Cambridge, UK, July 2001. Available at `www.inference.phy.cam.ac.uk` – URL Verified 28 Feb 07.

[22] S. Leonard. Intrusion detection in wireless local area networks. Master's thesis, Royal Military College of Canada, April 2003.

[23] S. Li and A. Ephremides. A network layer covert channel in ad-hoc wireless networks. In *Proceedings of the First Annual IEEE Communications Society Conference on Sensors and Ad Hoc Communications and Networks* (IEEE SECON 2004, Santa Clara, CA, Oct. 4-7), pages 88–96. IEEE Press, Oct 2004.

[24] S. Lin and D.J. Costello Jr. *Error Control Coding: Fundamentals and Applications.* Prentice-Hall, Inc., 1983. ISBN: 0-13-283796-X.

[25] D.K. Matai. Organized crime, terrorism and the internet. University of Oxford, February 2005. Guest speaker transcript.

[26] The MathWorks. Matlab and Simulink, 2007. Available from www.mathworks.com – URL verified 14 Mar 07.

[27] Microsoft. Visual c++ 2005, 2005. Available from http://msdn2.microsoft.com – URL verified 14 Mar 07.

[28] J. Millen. 20 years of covert channel modelling and analysis. IEEE Symposium on security and Privacy, 1999.

[29] M.L. Miller, G.J. Doerr, and I.J. Cox. Applying informed coding and embedding to design a robust, high capacity watermark. *IEEE Transactions on Image Processing*, 49(3), June 2004.

[30] I.S. Moskowitz and H.K. Myong. Covert channels - here to stay? In *Proceedings of COMPASS '94* (COMAPSS '94, Gaithersburg, MD, USA, Jun. 27 - Jul. 1), pages 235–243. IEEE Press, Jul 1994. Available from http://chacs.nrl.navy.mil – URL verified 28 Feb 07.

[31] P. Moulin and R. Koettler. Data-hiding codes. In *Proceedings of the IEEE*, pages 2083–2127. IEEE Press, Dec 2005. Available from http://www.ifp.uiuc.edu – URL verified 26 Feb 2007.

[32] S. Murdoch and S. Lewis. Embedding covert channels into tcp/ip. In *Proceedings of Information Hiding Workshop* (IH05, Barcelona, Catalonia(Spain), June 6-8), pages 247–261. SpringerLink, Jun 2005. ISBN: 978-3-540-29039-1.

[33] M. Owens. A discussion of covert channels and steganography. March 2002. Available from http://gray-world.net – URL verified 28 Mar 07.

[34] F.A.P. Petitcolas, R.J. Anderson, and M.G. Kuhn. Information hiding – a survey. In *Proceedings of the IEEE*, number 7 in 87, pages 1062–1078. IEEE Press, jul 1999. Available from www.petitcolas.net – URL verified 28 Feb 07.

[35] P. Piret. *Convolutional Codes.* The MIT Press, 1988. ISBN: 0-262-16110-9.

[36] E.A. Ratzer and D.J.C. MacKay. Codes for channels with insertions, deletions and substitutions. In *Proceedings of 2nd International Symposium on Turbo Codes and Related Topics* (ISTCRT '00, Brest, France, Sep. 4-7), pages 149–156. IEEE Press, Sep 2000. Available from www.cs.toronto.edu – URL verified 28 Feb 07.

[37] D. Roberts. *Developing for the Internet with Winsock.* Coriolis Group, Inc., 1995. ISBN: 1-883577-42-X.

[38] M. Schaefer, B. Gold, R. Linde, and J. Scheid. Program confinement in kvm/370. In *Proceedings of the 1977 Annual ACM Conference* (Seattle, Washington, USA), pages 404–410, New York, Oct 1977. ACM Press.

[39] G. Shah, A. Molina, and M. Blaze. Keyboards and covert channels. In *Proceedings of the 15th USENIX Security Symposium* (USENIX '06, Vancouver, BC, Canada, Jul. 31 - Aug. 4), pages 59–75. The USENIX Association, Aug 2006. Available from `www.usenix.org` – URL verified 28 Feb 07.

[40] C.E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, October 1948.

[41] G. Simmons. The history of subliminal channels. *IEEE Journal of Selected Areas on Communications*, 16:452–462, May 1998.

[42] E. Skoudis and T. Liston. *Counter Hack Reloaded.* Prentice Hall, 2nd edition, 2006. ISBN: 0-13-148104-5.

[43] M. Smeets and M. Koot. Research report: Covert channels. Technical report, University of Amsterdam, Amsterdam, Netherlands, February 2006. Available from `www.gray-world.net` – URL Verified 28 Feb 07.

[44] R.W. Smith. *On the Design of Network-Based Covert Communication Systems.* PhD thesis, Royal Military College of Canada, May 2006. Draft.

[45] W. Stallings. *Computer Networking with Internet Protocols and Technology.* Prentice Hall, 2004. ISBN: 0-13-141098-9.

[46] R. Stevens. *TCP/IP Illustrated: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols*, volume 3. Addison-Wesley, 1996. ISBN: 0-201-63495-3.

[47] K. Szczypiorski. Hiccups: Hidden communication system for corrupted networks. In *Proceedings of the International Multi-Conference on Advanced Computer Systems*, pages 31–40, 2003. Available from `http://krzysiek.tele.pw.edu.pl` – URL Verified 26 Feb 2007.

[48] C.-R. Tsai and V.D. Gligor. A formal method for the identification of covert storage channels in source code. In *Proceedings of the IEEE Symposium on Security and Privacy* (Oakland, California, USA), pages 74–86. IEEE Press, Apr 1987.

[49] G. Ungerboeck. Trellis-coded modulation with redundant signal sets part ii: State of the art. *IEEE Communications Magazine*, 25(2):12–21, February 1987.

[50] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13(2):260–, April 1967.

[51] A.A. Vladimirov, K.V. Gavrilenko, and A.A. Mikhailovsky. *WI-FOO - The Secrets of Wireless Hacking.* Addison Wesley, oct edition, 2005. ISBN: 0-321-20217-1.

[52] Z. Wang and R.B. Lee. Capacity estimation of non-synchronous covert channels. In *Proceedings of the 25th IEEE International Conference on Distributed Computing Systems Workshops* (ICDCS '05, Columbus, Ohio, USA, Jun. 6-10), pages 170–176. IEEE Press, Jun 2005. Available from `http://palms.ee.princeton.edu` – URL verified 28 Feb 07.

[53] WildPackets. Omnipeek personal, 2007. Available from `www.wildpackets.com` – URL verified 14 Mar 07.

Curriculum Vitae

Curriculum Vitae

Born in Etobicoke, Ontario in 1968, Maj Paul E.C. Martin spent his formative years in the west end of Toronto, Ontario Canada. In 1987, Maj Martin joined the Canadian Forces and attended the Royal Military College (RMC) in Kingston as an undergraduate. Graduating from RMC in 1991 in Electrical Engineering, Maj Martin was pasted to the Communication Squadron in Lahr, Germany where he started his formative training as a Communications and Electronics Engineering Officer.

Maj Martin returned to Canada in 1992 to complete his military training in Kingston, On and assume his first post in 1993 as the Assistant Operations Officer in the 8 Air Communications and Control Squadron in Trenton, On. One year later in 1994, Maj Martin became the Systems Officer to the Canadian Mission Control Centre of the Search and Rescue Satellite program also in Trenton, On. In 1997, Maj Martin was posted to El Gorah, Egypt as the Force Information Systems Officer for the Multinational Force and Observers. Maj Martin returned to Canada in 1998 to perform the duties of Information Systems Officer for the Canadian Forces Recruiting Centre located in Ottawa, On.

In 2002, Maj Martin joined the Deputy Chief of Defence's Joint Staff as a J6 Operations Desk Officer supporting deployed CF personnel with strategic voice and data communications. As a staff officer, Maj Martin was intimately involved with communications issues for CF personnel deployed to Afghanistan, the Democratic Republic of Congo, Egypt, Israel, central Europe and other parts of the Middle East. Maj Martin was promoted to his present rank in 2005 and accepted into the Master's program in Computer Engineering.