© 2012 Mavis Natasha Rodrigues

TIMING CHANNEL CODES FOR COVERT COMMUNICATION AND NETWORK FLOW TRACING

BY

MAVIS NATASHA RODRIGUES

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2012

Urbana, Illinois

Adviser:

Assistant Professor Negar Kiyavash

ABSTRACT

The goal of this thesis is to achieve covert communication, i.e. to covertly relay information using resources that are traditionally unintended for data transmission. In a covert channel, only the sender and receiver are aware of the communication and capable of reading the transmitted message, and any other party would not be aware of the existence of this transmission. The timing channel has been studied as a medium for covert communication in a computer network. It offers the means to communicate a message by relaying information encrypted within the timings formed between consecutive packets.

This thesis covers two schemes that utilize the timing channel for covert communication – to send a message and to trace network flows. The former, is the design of the first spyware circuit that covertly leaks spied information using an efficient encoder and for a network employing the CSMA/CA. This protocol modifies the inter-packet delays of the network flow using an exponential back-off rule. The second scheme does not relay any message but adds a signature called a watermark to enable flow tracing through the network with dependent deletion and substitution errors, and unlike earlier schemes, it is capable of withstanding desynchronization caused by packet drops. We will verify that our schemes are detectable by the receiver, and robust to the noise introduced by the network. In addition to their reliability, they are both difficult to detect and remain invisible to an observer.

To my parents and brother, for their love and support

ACKNOWLEDGMENTS

Writing this after the completion of my thesis, I find it to be the most difficult part – not because of the lack of names or stories to write but due to the fact that I can't find the words to express my gratitude and also because I don't think I will be capable of accurately expressing my thanks. Therefore expect nothing fancy.

I have to start by expressing my gratitude to my research advisor, Negar Kiyavash, for her time and her constant support and patience. She has been a source of inspiration and guidance during my research and has encouraged me during the difficult parts of my graduate years.

I would also like to thank Todd Coleman, for introducing me to research back in my undergrad, and my research group – Chris, Daniel, Farzaneh, Jalal, Sachin and Xun for the times we had at CSL, especially Xun who mentored and advised me through part of my thesis. Siva, for his company in the lab and his ideas and suggestions during this thesis, and Laura for taking the time to edit, in spite of my short notice.

To all my friends who are too many to list! Kayo, Naren, and Saurabh for ensuring that I still enjoyed my college life. Jane for being my roommate and friend and supporting through all my ups and downs. Sarah, my first roommate and friend when I came to this university, who never fails to listen to my very random calls especially when I need her the most.

My love, hugs and thanks to my family. Dad and Mom for investing in

my goals and dreams. My brother for believing in me more than myself. They have always been there from my earliest memories, and it is their love, scoldings and support that made me who I am now.

And finally, to God for His blessings throughout my life, without whom none of this would be possible. Before I came to this university, I was told that making true friends and gaining support would be difficult, but all these years have been filled with memories and friendships I will cherish throughout my life.

TABLE OF CONTENTS

LIST OF TABLES viii

LIST OF FIGURES ix

CHAPTER 1 INTRODUCTION 1
1.1 Covert Communication 1
1.2 Outline and Motivation 4
1.2.1 Timing Channel Spyware for CSMA/CA Protocol . . . 4
1.2.2 Watermarking 5
1.3 Notation 6

CHAPTER 2 LITERATURE SURVEY 8
2.1 Covert Timing Channels 8
2.2 Flow Tracing and Watermarks 10
2.3 Detecting the Timing Channel 12
2.4 Malicious Hardware 13

CHAPTER 3 PRELIMINARIES 14
3.1 Timing Channel 14
3.1.1 Exponential Server Timing Channel 15
3.1.2 Capacity 16
3.1.3 Service Times 17
3.2 Detecting Covert Communication 18
3.2.1 Regularity Test 19
3.2.2 Kolmogorov-Smirnov Test 19
3.2.3 Entropy Detection 20
3.2.4 Multi-Flow Attack 21
3.3 Low-Density Parity-Check Codes 22
3.4 Sum-Product Algorithm and Factor Graphs 24
3.5 Practical Codes for Timing Channels 25
3.5.1 Encoder Structure 26
3.5.2 Decoding Structure 29
3.6 CSMA/CA in IEEE 802.11 30

CHAPTER 4 A TIMING CHANNEL SPYWARE FOR CSMA/CA PROTOCOL 32
4.1 Introduction 32
4.2 Mechanism and Coding Scheme 35
4.2.1 Triggering the Spyware 35
4.3 Covert Communication with Timings 36
4.4 Implementation 38
4.5 Spyware Detection 40
4.5.1 Detection of Spyware Circuitry 41
4.5.2 Detection of Covert Communication 42
4.6 Experimental Results 43
4.6.1 Transparency of the Spyware Circuitry 45
4.6.2 Robustness of the Communication Scheme to Heterogeneous CSMA/CA Effects 46
4.6.3 Transparency of the Covert Communication Scheme . . 49
4.7 Conclusion and Future Work 51

CHAPTER 5 WATERMARKING: TRACING TRAFFIC FLOWS THROUGH THE NETWORK 53
5.1 Introduction 53
5.2 System Model 56
5.2.1 Sparsifier 57
5.2.2 QIM Embedder and Extractor 58
5.2.3 HMM Decoder 59
5.3 Evaluation 65
5.3.1 Effects of Watermark Length and Quantization Step-Size 66
5.3.2 Robustness against Network Jitters 68
5.3.3 Robustness against Packet Losses 68
5.4 Watermark Visibility 70
5.4.1 Kolmogorov-Smirnov Test 71
5.4.2 Multi-Flow Attack 71
5.5 Conclusion 73

CHAPTER 6 CONCLUSION 74

APPENDIX A INFORMATION THEORY CONCEPTS 76

APPENDIX B HIDDEN MARKOV MODEL AND FORWARD-BACKWARD ALGORITHM 78

APPENDIX C DERIVATIONS 79
C.1 Shaping 79
C.2 Little's Law 80

REFERENCES 82

LIST OF TABLES

4.1 The overhead introduced by the encoder implementation on a WARP node for three encoding matrix sizes. 45
4.2 Average SER of the spyware decoder vs. number of congesting nodes. The study is done for different probabilities of sending packets over the wireless medium, and the matrix sizes. 48
4.3 Regularity test, using a window of size 100 packets, fails to detect the presence of covert communication. 49
4.4 Detection (P_D) and false alarm (P_{FA}) probabilities for using the corrected conditional entropy (CCE) on the packet inter-arrival times. 50
4.5 Detection (P_D) and false alarm (P_{FA}) probabilities for using the entropy test (EN) on the packet inter-arrival times. . 50
4.6 Detection (P_D) and false alarm (P_{FA}) probabilities for using a Komolgorov-Smirnov test on the packet inter-arrival times. 51

5.1 False positive rates of tests with varying parameters for a threshold set to have less than 1% false positives. 67
5.2 True positives rates of tests with varying parameters for a threshold set to have less than 1% false positives. 67
5.3 True positive (TP) watermark detection rates for a Laplacian modeled network jitter with varying standard deviation (S.D.). 68
5.4 True positive (TP) watermark detection rates for various deletion ratios P_d. All false positive (FP) rates are restricted under 1%. 70
5.5 Average K-S distances for varying watermark lengths and step-sizes. 71
5.6 Average number of empty intervals over the first 500 packets for watermarked and unwatermarked flows. 73

LIST OF FIGURES

3.1 Overview of a timing channel that embeds a hidden message within the inter-packet delays of the packet flow 14
3.2 Forney factor graph representation. 24
3.3 The stages involved in the encoding process that maps the encoded symbol to an inter-packet delay 26
3.4 Forney factor graph of $\mathbf{Hx}=\bar{\mathbf{s}}$. 30
3.5 A flow chart of the CSMA/CA protocol that uses exponential back-offs 31

4.1 Scenario of a warden monitoring network traffic that may include a cover timing channel 37
4.2 Setup of an infected transmitted leaking data using a timing channel 38
4.3 Ways of interaction between the CSMA/CA (MAC) and the physical layer (PHY) 39
4.4 Block diagram of the encoder circuit 39
4.5 Boxplots of the symbol errors across multiple trials, using the large encoding matrix with burstiness probabilities $B_p \in \{0.4, 0.6\}$ with 200 runs. 48

5.1 Overview of our watermarking scheme. 56
5.2 An example for substitution errors caused by IPD jitters. '$\mathbf{x}$'s are '0' quantizers and '$\mathbf{o}$'s are '1' quantizers. The bit embedded on I_i is 1, but the decoded bit from $\hat{I}_i$ is 0. 59
5.3 Merging of IPDs when packets are dropped. 61
5.4 The HMM of the watermark-over-network channel. 61
5.5 A histogram of the correlations between the extracted watermark $\hat{\mathbf{w}}$ and $\mathbf{w}$ of 5000 marked flows (solid curve) and 5000 unmarked flows (dotted curve). 69
5.6 Histogram of empty intervals in an aggregate of 10 flows . . . 72

CHAPTER 1

INTRODUCTION

1.1 Covert Communication

The aim of this thesis is to present methods to achieve covert communication. Covert communication means disclosing unauthorized information or relaying a message, especially by using resources that are not intended for the purpose of communication [1]. In a covert channel, only the intended parties are aware of the communication and are capable of reliably decoding the hidden data. Any unintended parties should not be able to recover the transferred data, and, even better, should be unaware of the existence of this communication. The analysis of covert communication has attracted a large amount of interest, such as in military communication. It is also means to thwart attacks made by malicious hackers who could use these covert channels to attack systems and organizations to leak personal and secured information. Computer hackers sometimes launch attacks on organizations by using 'stepping stones', i.e. redirecting their traffic, by using the systems of unaware network users, in order to hide the source of the attack. In this case, covertly inserting a marker in the network flows would enable linking these flows and thereby identifying flows belonging to the same chain. In short, the study of covert communication is important for security, not only for big businesses, but also for the individual user.

Communication over a network involves sending a series of packets from

one host system to another, where each packet consists of some part of the message. One method used to ensure that only the intended parties are able to decode the message from these packets, is to encrypt the information itself. The content of the packets could require some form of decoding to read the data and the overall message. Yet if the goal is not only encryption, but also to hide the presence of the communication, then communicating packets that are obviously encrypted will not suffice.

Covert communication uses system resources that are not intended for communication for the purpose of relaying information. Security analysts categorize covert channels into two forms [2]:

- Storage channels which involve reading and writing from a storage location.
- Timing channels which use system resources such as CPU times for the sending and receiving processes to communicate with each other.

The second item, the timing channel, is the means of covert communication that is the subject of this thesis. In a network, a system resource that can potentially be used to relay information is the time between packets. Any message a user sends via the Internet is broken into packets that are efficiently handled by the network. In each of these network flows, there are spaces between the packets – each packet has a small amount of space between itself and the next packet. These spaces are referred to as inter-packet delays (IPDs) or inter-arrival times, defined as the time between two consecutive packets. These IPDs always exist in any network flow, and though they are not intended for communication, research in the past few years has looked at encoding messages within these IPDs. A simple case of encoding a binary message would be to map an even inter-packet delay to a bit '0' and an odd

inter-packet delay to a bit '1'.

Using IPDs between packets has many advantages. It offers a new degree of freedom to encode a message that is independent of the size of the packets and present in every network flow. It is also a resource that is not intended or considered for communication. As long as there are not any adverse changes to the IPDs, the network traffic and its packets remain unaffected. Inter-packet delays thus provide an excellent method to hide a message, as long as there is not much difference between the IPDs of a normal network flow and IPDs of flows with a hidden message. However, there are some challenges with using IPDs to communicate a message – the IPDs within a network flow are always perturbed by the network. Each network flow goes through various devices (routers, switches, etc.) which modify or perturb the IPDs; therefore, the packet timings, at the receiver's end, are no longer the same as the original timings when encoding the message. These devices that change the IPDs of the network flow are modeled as a *queuing channel* since it processes each packet in the order of arrival.

This thesis will look into methods of relaying information covertly via the timing channel. The goal is to relay some data from one system to another using the timing channel in a network. Chapter 4 explains a scheme that sends data encoded within the IPDs to create a spyware for a network that employs the commonly used CSMA/CA protocol (Carrier Sense Multiple Access with Collision Avoidance), which as its name implies, attempts to avoid collisions between various flows by using exponential back-offs. Chapter 5 describes the addition of a signature, called a watermark, to each flow that will help trace it through the network. The noise added to the timing channel is created by a network that has Laplacian jitters, and randomly drops packets. The following sections of this introduction will further describe these

chapters.

1.2 Outline and Motivation

Chapter 2 provides a survey on the research in the past few years on timing channels. Some researchers have analyzed the timing channel and others have provided schemes that utilize it. The chapter then explains the research in flow tracing and detecting stepping stones, including tracing flows by embedding a watermark in the timing channel. This chapter will also list some of the detection techniques that are capable of detecting the presence of the covert timing channel.

Chapter 3 offers a detailed explanation on some of the algorithms and techniques used in the research covered in later chapters. It starts by describing the timing channel model, and the model for the network and its essential characteristics. It also describes a technique that encodes a message within the IPDs of the network flow. Later it lists the detection techniques that will be used to test the visibility of the schemes proposed in Chapter 4 and 5. It concludes by giving a short explanation on some of the concepts used in the later chapters, such as the low-density parity-check codes, the sum-product decoding algorithm and the CSMA/CA protocol.

The following subsections will describe the motivation behind the subjects covered in Chapter 4 and Chapter 5.

1.2.1 Timing Channel Spyware for CSMA/CA Protocol

Advances in technologies have lead to the increasing importance of data security and privacy. It is essential that security vulnerabilities of systems are studied and analyzed to prevent the leakage of vital or personal information.

Chapter 4 will discuss a new covert communication that leaks secured and private chip data using the timing channel. It utilizes a coding scheme which embeds the data within the IPDs between consecutive packets. The network noise is modeled by a $\cdot/M/1$ queuing channel, which adds random independent, identically distributed (i.i.d) exponential or geometric values to the IPDs of the original flow. It uses sparse-graph linear codes followed by a shaping technique to transform uniformly distributed random variables to one with the required distribution.

The noise that perturbs the IPDs is introduced by the IEEE 802.11 CSMA/CA protocol. This protocol is a carrier sensing scheme where a node attempts to avoid collision by first sensing if the channel is available. In case the channel is busy, the node inserts an exponential back-off. This chapter will discuss the first reliable covert communication system, based on timing channels, capable of being robust to perturbations introduced by the CSMA/CA's back-off rule.

Section 4.1 will introduce the topic and state the goals and contributions and Section 2.4 will survey literature on malicious hardware and their detection. The scheme proposed in Chapter 4 will use the timing channel explained in Section 3.1 and the coding scheme explained in Section 4.3. An explanation of the CSMA/CA protocol is given in Section 3.6.

1.2.2 Watermarking

Watermarking is a technique used to trace packet flows as they pass through the network either to find the source of an attack or to attack the anonymity of a network flow. Watermarks exploit the timing channel by perturbing the IPDs to embed some identifiable mark that is robust to any network effects

(jitters, packet losses) but at the same time remains invisible to an outside observer. Watermarks are divided into two types:

- Interval-based watermarks which perform transformations on an interval and are robust to packet losses, but detectable using the multi-flow attack [3].
- Inter-packet delay watermarks which modify each individual inter-packet delay, but are vulnerable to packet losses since they lead to desynchronization.

The research in the past few years is given in Sections 2.2 and 2.3. Previous research has mainly focused on robustness against network jitters that perturb the IPDs and disregarded the possibility of packet losses. In Chapter 5 we will present a novel IPD-based watermarking scheme that is shown to be robust to both network jitters and packet losses, while remaining invisible to the detection techniques explained in Section 3.2.

1.3 Notation

For consistency we will be standardizing the following notations throughout this thesis.

- Random variables will always be noted using italic upper case letters, and values using italic lower case letters.
- A vector will be denoted by bold lower case letters, and matrices by bold upper case letters.
- Sequences are represented by italic lower case letters with the length as an exponent. For example, $x^n = \{x_1, x_2, x_3, \ldots, x_n\}$ where n is its

length. Each element of x^n is denoted by x_i given an index $\{i : i \in \mathbb{Z}, 0 < i \leq n\}$.

There is no difference between the representation of a vector $\mathbf{x} \in \mathbb{R}^n$ and $\{x^n : x_i \in \mathbb{R}, i \in \mathbb{Z}_+\}$. The decision to represent a notation by either form would depend on the context in which it is being used; for mathematical equations, the vector representation is preferred.

CHAPTER 2

LITERATURE SURVEY

2.1 Covert Timing Channels

Historically, timing channels are synonymous with covert channels [1, 4, 5, 6, 7, 8, 9]. Lampson [1] studied the problem of preventing a program from communicating with another program that has not called it and classified the various possible leaks. His is one of the first papers that defines the timing channel and categorizes covert communication. The Department of Defense [4] published a paper that states that the evaluation of the reliability and security of computer systems requires the analysis of timing channels. Kang et al. [6] noticed that a covert channel is formed by the timings of ACKs during the communication between a 'High' receiver and a 'Low' sender. This channel could be used to send messages from 'High' to 'Low', i.e. the intended receiver 'High' could send messages to the intended sender 'Low'; therefore, the communication is not in the permitted direction.

Timing channels have been studied to understand their characteristics [10, 5]. Anantharam et al. [10] studied the capacity of queuing channels where the information is encoded within the IPDs of the packet flow and the receiver sees a noisy version of the input. A single server queue with deterministic service time has infinite capacity since the packets could be sent at a rate greater than the service time. For non-deterministic service times, finding the capacity is a challenge due to the channel being non-memoryless, non-

stationary, and non-linear. They show that even with departures that are a random transformation of the input to the channel, reliable decoding is possible as long as the input flow rate is below capacity. Among these single-server queues, the capacity is the lowest for those with exponential service times and this capacity cannot be increased by feedback.

Bedekar et al. [11] studied the discrete case version of Anantharam et al. [10] studied and solved for the capacity of discrete time single-server queues where the service times are i.i.d integer values. They found that among all queues with such a model, the queue with geometrically distributed service times has the smallest capacity.

Sundaresan et al. [12] are among the first to find the existence of good tree codes with sequential decoding for an exponential server timing channel. They were able to improve on earlier decoding techniques and provided a finite upper bound on the expected number of computations to move one step ahead. However, the complexity was reduced at the expense of the rate, and their scheme could only achieve rates half that of capacity.

Some research focuses on creating and designing practical network timing channels. Padlipsky et al. [13] used an ON/OFF scheme to covertly communicate a message where a packet in a predetermined interval indicates a bit '1' which is otherwise a bit '0'. Cabuk et al. [14] provided an IP timing-based design that uses a similar technique on the IP protocol using a special starting sequence for synchronization. Luo [9] proposed a TCP-based timing channel against network jitters and packet losses by encoding a message into TCP packet bursts and showed that their timing channel has a higher decoding accuracy compared to the IP timing channel.

2.2 Flow Tracing and Watermarks

The ability to trace a network flow as it travels through the network is essential in preventing malicious users from masking the source of an attack or impeding anonymous traffic. Staniford-Chen and Heberlein [15] proposed a thumbprint that would embed a signature in the packet contents and compared flows in different connections; but since most traffic is encrypted, the contents are inaccessible. Recent works have shown that similarities in communication patterns such as packet sizes and timings may be used for flow linking [16, 17, 18, 19]. There are two methods to analyze traffic – *active* or *passive*.

Passive analysis such as [15, 16, 20, 21] uses the original characteristics of a packet flow. Zhang and Paxson [16] examined the ON/OFF periods, i.e. the period of activity and inactivity of the traffic, by measuring the coincident transitions from the ON state to the OFF state. Yoda and Etoh [20] detected flows that belong to the same chain by comparing the deviations defined as the difference between the average propagation delay and the minimum propagation delay between two connections. Wang et al. [21] showed that the IPDs are preserved over router hops and stepping stones, and, since they are unique to the user, can be used as a basis for a correlation measure. These schemes rely only on the original characteristics in the observed flows, which can be easily weakened by timing perturbations during network transmissions. Therefore, successful detection using these schemes often requires analysis of a large number of packets.

Active analysis schemes inject a signature or a pattern into the IPDs of the flows. They are further divided into interval-based schemes and IPD-based schemes. Pyun et al. [18] is a case of the former, where when a '0'

is embedded, all packets in a selected interval are squeezed into the subsequent interval. Kiyavash et al. [3] showed that interval-based watermarking schemes are vulnerable to the *multi-flow attack* whereupon observing a few flows, the attacker can detect the watermark as an abnormally large number of empty time periods that were created during the embedding process. In IPD-based schemes [17, 22, 19], watermark bits are embedded into the inter-arrival times of packets. Wang and Reeves [17] first introduced using watermarks within network flows. They suggested introducing small perturbation within the IPDs by using a quantization encoding scheme, where each bit of the watermark is embedded into $m > 1$ packets. They demonstrated that their scheme was robust to timing perturbations and proved that they could theoretically achieve close-to-perfect detection rates for sufficiently long flows. Later, Wang et al. [22] used IPD-based watermarks on encrypted peer-to-peer Voice over IP, using several milliseconds of timing perturbations to create unique flows. Wang et al. [23] also showed that low-latency anonymous systems are not able to disguise the network flows against sufficiently long watermarks in spite of various flow transformations such as bogus packets, traffic padding, flow mixing and splitting, natural or intentional jitter, or packet drops. Earlier schemes mainly dealt with robustness against network jitters and did not consider the possibility of packet losses that lead to desynchronization. Haumansadr et al. [19] extended their IPD-based watermark to handle packet losses by performing selective correlation where packets that did not have a corresponding match in the incoming flow are removed. However, this scheme is *non-blind*, requiring the storage of the incoming flow's timings.

2.3 Detecting the Timing Channel

The covert timing channel that either communicates a message or simply traces the flow must be invisible to an outside observer, i.e. the observer should not be aware of the use of the timing channel, or able to extract the encoding parameters.

Earlier work has been mainly focused on disrupting or completely eliminating covert timing channels [24, 25]. Kang and Moskowitz [24, 26] moderated the ACKs in order to decrease the capacity of the covert timing channel by decoupling the ACKs sent from a 'High' to 'Low' and adding random noise. Giles et al. [25] limited the capacity in a game between the sender and receiver versus a jammer. More recent work has focused on the detection of covert timing channels [14, 27]. Cabuk et al. [14] detected highly regular behavior in the IPDs of a covert flow as a result of static encoding of frames, whereas normal flows do not have any regular patterns. Berk et al. [27] noted that in a binary covert channel, the mean IPD is in between two spikes, formed by a histogram of the flow's IPDs, compared to a normal flow which has one large spike about the mean IPD. Gianvecchio et al. [28] provided a technique that uses the entropy of the IPDs of network flow to indicate if the flow contains information within its IPDs. Peng et al. [29] studied the visibility of watermarks and the importance of careful design in order to prevent an attacker from being able to detect and recover watermark parameters and therefore remove the watermark or discard watermarked flows. They used the technique proposed by Wang et al. [17] and showed the attacker needs only to have access to the time stamps of the packets. They noted that embedding a watermark forms packet delays of either a uniform distribution or a combination of uniform and normal distributions. They were able successfully to

employ the Kolmogorov-Smirnov test [30, 31] that compares the statistical distribitions of the IPDs to effectively distinguish between watermarked and unwatermarked flows.

2.4 Malicious Hardware

Research in hardware malware has recently emerged, especially after the Department of Defense's Defense Science Board comprehensive report on the subject in 2005 [32]. Much of this work has concentrated on detection of foundry-inserted Trojans (malware), which is typically done by invasive or noninvasive study of the post-silicon ICs and comparison with the original design files to detect the modifications to the manufactured chips [33]. A number of methods for protection of ICs and third-party IP cores against piracy have been invented [34]. The threat of insertion of spyware in communication and security hardware modules has been identified [35]. Recently, King et al. [36] have designed and implemented malicious hardware that can get hidden login access or can steal the security keys on a microprocessor. The malicious hardware is embedded at the micro-architecture level by addition of a block of gates and does not modify the internals of the architectural blocks.

CHAPTER 3

PRELIMINARIES

3.1 Timing Channel

In a network, any communication is typically relayed in the packet contents of a network flow, but the spaces or pauses between consecutive packets offer another medium for communication. The timing channel for this scenario is depicted in Figure 3.1. Consider two individuals – a sender and receiver who wish to send a series of bits through a channel. Each bit is encoded within the IPDs of successive packets. In between is a queuing channel, a channel that introduces noise by distorting the original IPDs.

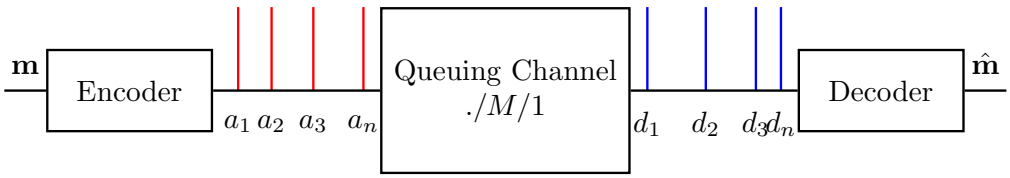


Figure 3.1: The message $\mathbf{m}$ is encoded within the arrival times $a_1, \ldots, a_n$. The $\cdot/M/1$ queue represents devices, such as routers, which introduce noise, leading to the modification of the timings. This results in the departure times $d_1, \ldots, d_n$ read by the decoder to produce the message $\hat{\mathbf{m}}$.

The sender wishes to encode a message $\mathbf{m}$ within the IPDs of the input flow with arrival times $a^n = \{a_1, a_2, \ldots, a_n\}$ where n is the total number of

packets and a_i is the the time at which the packet i is generated. The packets in the incoming flows arrive at an average rate of λ, therefore

$$E\left[\sum_{i=1}^{n-1}\frac{(a_{i+1}-a_i)}{n}\right]=\frac{1}{\lambda}$$

The IPDs are defined as the times between two consecutive packets. For the above case, the IPDs of the arrival flow a^n are given by

$$I_i^a \triangleq a_{i+1}-a_i, \quad i\in[1,n)$$

3.1.1 Exponential Server Timing Channel

The channel that distorts the IPDs of the original flow is modeled as a queuing channel. Similar to the everyday 'queue', the queuing channel services or processes each 'customer' (the packet) in the order of arrival. Since each packet is processed, there is a service time added to the IPDs of each packet. These increases or decreases in the inter-packet delays will be referred to as *service times*. A queuing channel is defined by the rate of the arrival process of its customers, the average number of customers that it serves and the number of servers.

In this thesis, the queuing channel is modeled as a $\cdot/M/1$ queuing channel, such as the exponential server timing channel (ESTC), which is the simplest queuing channel. It is a first-come-first-served channel that can handle Poisson-distributed arrivals of rate λ and processes each packet with independently and identically distributed (i.i.d) exponential service times of rate μ.

3.1.2 Capacity

Using the timing channels to encode information offers an interesting situation to analyze and to find its properties. The ESTC queuing channel is non-memoryless, non-stationary, and non-linear, and it is challenging to find the capacity of such channels. Anantharam's 'Bits through Queues' [10] studied the capacity of queuing channels that coded information within timings.

For a single-server $\cdot/M/1$ queue with rate μ, the capacity given an arrival process with average rate $\lambda \in (0, \mu)$ is given by [10]

$$C(\mu, \lambda) = \lambda \log_2 \frac{\mu}{\lambda} \quad \text{(bits per second)}, \quad \lambda < \mu \tag{3.1}$$

The capacity is obtained by maximizing $C(\mu, \lambda)$ over all possible $\lambda < \mu$ and is given by

$$C(\mu) = e^{-1} \mu \log_2 e \quad \text{(bits per second)} \tag{3.2}$$

where the optimal rate is a Poisson process with rate $\lambda^* = \exp^{-1} \mu$. Since in a period of T seconds there are an average of $n = \lambda T$ packets, we get

$$\begin{aligned} \tilde{C}(\mu, \lambda) &= \log_2 \frac{\mu}{\lambda} \quad \text{(bits per packet)}, \quad \lambda < \mu \\ \tilde{C}(\mu) &= \log_2 e \quad \text{(bits per packet)} \end{aligned} \tag{3.3}$$

Bedekar et al. [11] studied the discrete time single-server queues where the service times are i.i.d integer values with mean $1/\mu$ slots. The discrete time analogue of the continuous time case as given by [11, 37] for rate $\lambda \in [0, \mu)$

$$C(\mu, \lambda) = H(\lambda) - \frac{\lambda}{\mu} H(\mu) \quad \text{(bits per slot)} \tag{3.4}$$

where $H(.)$ is a binary entropy function. The timing channel capacity obtained by taking the supremum of all λ over the timing capacities for $0 \leq \lambda < \mu$ is

$$\begin{aligned} C(\mu) &= \log_2(1+\nu) \quad \text{(bits per slot)} \\ \nu &\triangleq e^{-\frac{H(\mu)}{\mu}} \end{aligned} \tag{3.5}$$

where the maximum is achieved by a Bernoulli process with rate

$$\lambda^* = \frac{\nu}{1+\nu} \tag{3.6}$$

Similarly, for T slots there is an average of $n = \lambda T$ packets, therefore

$$\begin{aligned} &\tilde{C}(\mu,\lambda) = \frac{H(\lambda)}{\lambda} - \frac{H(\mu)}{\mu} \quad \text{(bits per packet)} \\ is \\ &\tilde{C}(\mu) = (1+\frac{1}{\nu})\log_2(1+\nu) \quad \text{(bits per packet)} \end{aligned} \tag{3.7}$$

3.1.3 Service Times

We can map each departure packet d_i to its arrival packet a_i as follows.

$$P(\mathbf{d}|\mathbf{a}) = \begin{cases} \prod_{i=1}^{n} P_{S_i}(s) & \text{if } d_i \geq a_i, \\ 0 & \text{otherwise} \end{cases} \tag{3.8}$$

where $S = \{s_1, s_2, \ldots, s_n\}$ represents the service times of the queuing channel and can be defined as

$$s_i \triangleq d_i - max(d_{i-1}, a_i) \tag{3.9}$$

The maximum term ensures that we consider that the queue might have to process the previous packet before the next successive packet. For example, the queue may take some time (s_1) to process the first packet, but if the

second packet arrives before the first is processed, i.e. $a_2 < d_2$, then the service time of the second packet is $d_2 - d_1$.

In continuous time the service times are i.i.d exponentially distributed with rate μ and thus have a probability density function $P_{S_i}(s) = \mu e^{-\mu s}$, whereas for the discrete case, they are geometrically distributed with probability mass function $P_{S_i}(s) = \mu(1-\mu)^{s-1}$.

3.2 Detecting Covert Communication

The countermeasures to covert timing channels could be divided into two categories: detection and disruption. The detection of covert timing channels is based on statistical tests that can distinguish between legitimate and covert traffic. The disruption (e.g. jamming) of covert timing channels is done by an active warden that aims to prevent the covert communication, usually with the undesired side effect of degraded system performance. While the focus of earlier work has been mainly on disrupting or completely eliminating covert timing channels [24, 25], recent works have focused more on their detection [14, 27]. This is mainly because the disruption of timing channels is usually costly – either in terms of resources used to actively monitor the traffic or the degradation of the system performance. One possible disruption is random delaying of the traffic, which reduces the performance possibly by reducing the capacity or increasing the error rates of covert timing channels. Another disruption would be to inject false packets. However, such a countermeasure is both costly in terms of the attacker resources and the lost capacity of the legitimate users of the channel. Therefore, it is highly desirable to employ disruption mechanisms only after one has verified the presence of a covert channel reliably.

Some special-purpose detection tests that are designed for specific covert timing channels include Cabuk et al.'s [14] ϵ-similarity for IP covert timing channels and Berk et al.'s [27] mean-max ratio to test for binary or multi-symbol covert timing channels. The mean-max ratio test assumes that the legitimate IPDs follow a normal distribution, which is often not true for real network traffic.

The following sub-sections will explain some well-known tests that are employed to detect a covert channel and will be used in the later chapters to analyze the visibility of our schemes.

3.2.1 Regularity Test

Regularity tests are based on the assumption that the covert traffic is more regular than the legitimate traffic. One example of such a test is Cabuk et al.'s heuristic regularity test [14] which examines whether the variance in the inter-arrival times remains small. Specifically, the test traffic is separated into non-overlapping windows of packets. For each window i, the standard deviation σ_i of the inter-arrival times is computed. If the standard deviation of the pairwise differences between all pairs σ_i and σ_j for $i < j$ remains small, the traffic is considered highly regular and hence the presence of a covert channel is declared.

3.2.2 Kolmogorov-Smirnov Test

Statistical analyses are commonly used to decide if two sets of data belong to the same distribution. The Kolmogorov-Smirnov (K-S) test is one of the most commonly used statistical tests that can detect if two data sets belong to the same distribution or not. It differs from other tests by being a non-

parametric test, i.e. it does not depend on the distribution of the data. It evaluates the equality of one-dimensional distributions of probability. Peng et al. [29] used this technique to successfully distinguish between a flow that contained a watermark and a legitimate flow.

The Kolmogorov-Smirnov test evaluates the similarity between two sequences, by finding the maximum distance in their empirical distribution functions [38] and is based on the earlier works of Kolmogorov [30] and Smirnov [31]. An attacker can look at the flows arriving at a compromised host and calculate the maximum empirical difference between the flow and a known legitimate flow. If the difference is large, then the attacker declares that it contains a covert message.

The maximum distance between two cumulative distributions will be referred to as the K-S distance and is represented by d_K. The K-S distance between two flows is measured as

$$d_K = \sup_x(|F_A(x) - F_B(x)|) \tag{3.10}$$

where $F_A(x)$ and $F_B(x)$ are the empirical distribution functions of two data sets A and B that are to be compared, which in the later chapters will be the IPDs of two flows.

3.2.3 Entropy Detection

Developed by Gianvecchio and Wang [28] entropy detection is one of the latest detection techniques capable of identifying the existence of a timing channel, since the existence of the timing channel affects the entropy of the packet flow. They tested their design on three covert timing channels [14, 39, 40], and found that the combination of the *entropy* (EN) and the *corrected condi-*

tional entropy (CCE) on a sequence of binned data is effective at recognizing a covert timing channel.

The EN is simply the first-order entropy where a low entropy coveys the existence of a timing channel. The *entropy rate*, which can be used to measure complexity and regularity, is estimated on a finite data set using the corrected conditional entropy given by

$$CCE(X_\gamma|X_{\gamma-1}) = CE(X_\gamma|X_{\gamma-1}) + perc(X_\gamma)EN(X_1) \tag{3.11}$$

where CE is the condition entropy, $perc(X_\gamma)$ is the percentage of unique pattern of length γ and $EN(X_1)$ is the first-order entropy. The estimate of the entropy rate is the minimum of equation (3.11) over all possible values of γ.

To get the sequences, the data is binned into B bins where the suggested values of B for the corrected conditional entropy is $\mathsf{B} = 5$ and for the entropy test is $\mathsf{B} = 65536$. The authors provided an implementation of their design using a B-ary tree of height γ where the leaves are patterns of length γ and the children of the root are patterns of length $\gamma = 1$. Therefore a breath-first-search gives the corrected conditional entropy for each level.

3.2.4 Multi-Flow Attack

Kiyavash et al. [3] showed that using interval-based watermarking techniques leads to time dependent correlations. They provided a simple and effective technique that does not have much computational complexity, and not only will detect a watermark but is also capable of extracting hidden watermark parameters, resulting in its removal at a low cost. They had demonstrated this scheme on existing interval-based watermarks [23, 18, 41], successfully

locating cleared intervals and removing the watermark. It is shown to weaken against watermarks located at multiple positions [3].

The attacker, who is in control of a compromised host, could create flows, that will not be relayed, to monitor the flows at the compromised host. Needing as few as 10 watermarked flows, the attacker would simply line up the flows and observe the unusually high number of empty intervals caused by emptying out intervals when embedding the watermark. Using this technique, the attacker can then extract the required parameters, and remove the watermark from all flows that will be relayed through the compromised host.

The multi-flow attack takes a histogram of the sorted union of a few watermarked flows. The detection of a high number of empty intervals shows the presence of a watermark. Kiyavash et al. showed that if the same watermark message is present in all the flows, then as few as 10 watermarked flows are required to detect the watermark. If the watermark message differs in each flow, the attacker needs to aggregate a subset of flows that contain the same bit value to detect empty intervals.

3.3 Low-Density Parity-Check Codes

Gallager introduced low-density parity-check codes back in his 1960 doctoral dissertation [42] which are shown to achieve capacity [43]. These codes can be decoded using iterative decoding techniques such as the sum-product algorithm [44, 45, 46] which will be explained in Section 3.4. The codes are defined by a *parity-check matrix* $\mathbf{H}$, which is a sparse parity-check matrix with m rows and n columns and $m < n$, where n is the the block length and

m is the number of parity-checks. The set $\mathcal{C}$ represents the set of codewords

$$\mathcal{C} = \{\mathbf{x} : \mathbf{Hx} = \mathbf{0}\} \tag{3.12}$$

The space $\mathcal{C}$ represents all sets of codewords $\mathbf{x} \in \mathcal{C}$ that satisfy the condition $\mathbf{Hx} = \mathbf{0}$. A transformation on equation (3.12) yields equation (3.13) which generates the codeword $\mathbf{x}$ using a length-κ vector $\mathbf{u}$ and a *generator matrix* $\mathbf{G}$ which is an n by κ matrix. The generator matrix is used to map a length-κ ($\kappa = n - m$) information vector $\mathbf{u}$ to a length-n codeword vector $\mathbf{x}$.

$$\mathbf{x} = \mathbf{Gu} \tag{3.13}$$

Equation (3.14) below is an example of a parity-check matrix. Each column is associated with a codeword bit – column 1 with $\mathbf{x_1}$, column 2 with $\mathbf{x_2}$, etc. Each row represents a parity-check where a '1' signifies that this bit contributes to the parity-check. This implies equation (3.15)

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{3.14}$$

$$\begin{aligned} x_2 \oplus x_4 &= 0 \\ x_1 \oplus x_3 \oplus x_5 &= 0 \\ x_1 \oplus x_2 \oplus x_5 &= 0 \end{aligned} \tag{3.15}$$

3.4 Sum-Product Algorithm and Factor Graphs

The sum-product algorithm is an iterative *message passing algorithm* to calculate the marginal distributions. This algorithm is used to decode low-density parity-check codes (given in Section 3.3), but in various other areas iterative decoding techniques are an instance of the sum-product algorithm such as the forward-backward and Veterbi algorithms for hidden Markov models or the Kalman filter for signal processing [47].

The sum-product algorithm is solved using graphical models called factor graphs. One specific type of factor graph called the Forney factor graph [48] uses square nodes to denote function and edges or half-edges to denote the variables. Suppose e_i is an edge of a node g, then g is a function of e_i.

Suppose g is a function over variables $\{e^*, e_1, \ldots, e_n\}$. Let $\mu_{g \to e^*}$ be the 'message' sent from node g along edge e^*, then

$$\mu_{g \to e^*} \triangleq \sum_{e_1, \ldots, e_n} g(e^*, e_1, \ldots, e_n) \quad \mu_{e_1 \to g}(e_1) \ldots \mu_{e_n \to g}(e_n) \tag{3.16}$$

For example, Figure 3.2 is a graphical representation of equation (3.17) which finds the marginal distribution of x_3 over a function f that can be factored into functions f_1, f_2, f_3, f_4.

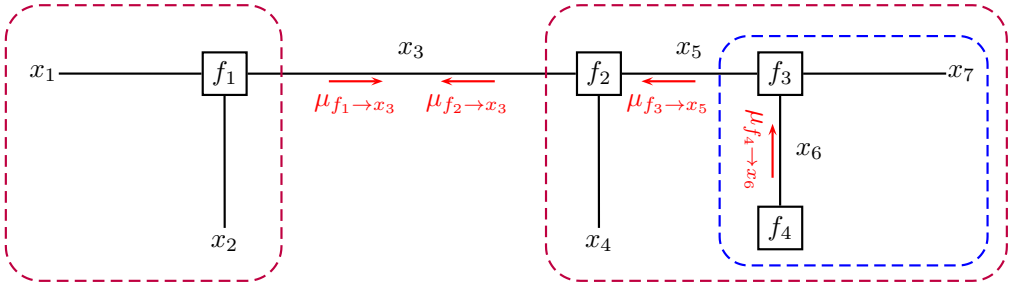


Figure 3.2: Forney factor graph representation.

$$
\begin{aligned}
f(x_3) &= \sum_{x_1,x_2,x_4,x_5,x_6,x_7} f(x_1,x_2,x_2,x_4,x_5,x_6,x_7) \\
&= \sum_{x_1,x_2,x_4,x_5} f_1(x_1,x_2,x_3) f_2(x_3,x_4,x_5) f_3(x_5,x_7,x_6) f_4(x_6) \\
&= \sum_{x_1,x_2} f_1(x_1,x_2,x_3) \sum_{x_4,x_5} f_2(x_3,x_4,x_5) \sum_{x_5,x_6} f_3(x_5,x_7,x_6) f_4(x_6)
\end{aligned} \tag{3.17}
$$

where each factor can be summarized as a closed box (μ) over the internal variables enclosed within the dashed boxes of Figure 3.2, which results in the following

$$
f(x_3) = \underbrace{\sum_{x_1,x_2} f_1(x_1,x_2,x_3)}_{\mu_{f_1\to x_3}} \underbrace{\sum_{x_4,x_5} f_2(x_3,x_4,x_5) \overbrace{\sum_{x_5,x_6} f_3(x_5,x_7,x_6) f_4(x_6)}^{\mu_{f_3\to x_6}}}_{\mu_{f_2\to x_3}} \tag{3.18}
$$

$$
f(x_3) = \mu_{f_1\to x_3} \quad \mu_{f_2\to x_3}
$$

Notice that equation (3.18) matches the equation in equation (3.17); therefore using the sum-product algorithm we can recursively solve for each edge of the graph.

3.5 Practical Codes for Timing Channels

Coleman and Kiyavash [49] designed a coding scheme capable of achieving the fundamental limits of the timing channel given in Section 3.1.2. This section will explain the mechanism of their encoding and decoding scheme, which will be used in Chapter 4.

3.5.1 Encoder Structure

The encoder controls the IPDs between successive packets. It needs to encode a message defined on $[0, Q-1]$, but the IPDs of a normal network flow follow certain distributions. On a continuous time scale, these delays are exponentially distributed, and on a discrete time scale, they are geometrically distributed. To map each bit of the encoded message to an IPD, the encoder utilizes a shaping technique that is capable of transforming a uniformly random variable into one with the desired distribution. Given this technique, the encoding now require generating a uniformly distributed random variable for each bit of the message. To generate the required uniformly distributed inputs, the encoder employs dithers that are proven to generate a uniformly distributed random variable. Moreover, the encoder also uses linear coset codes defined on a finite field that is shown to be capable of achieving capacity.

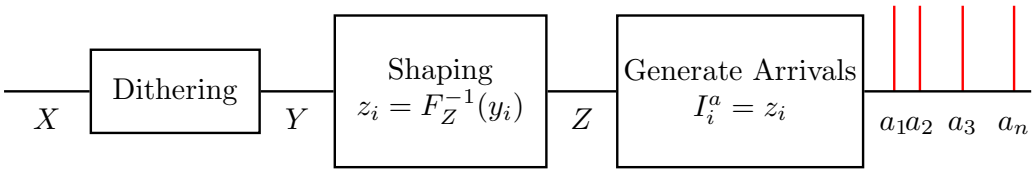


Figure 3.3: The stages in the encoding process that maps each x_i to an either exponential or geometric inter-arrival time given by z_i. The dithering stage ensures the distribution is uniform, and shaping ensures that the distribution matches the Poisson distribution of the arrival process.

Figure 3.3 gives an overview of the encoding process:

- The process X first undergoes a *dithering* process that ensures that the distribution is uniform resulting in Y.

- The result Y is transformed into the required exponential or geometric distribution through *shaping* forming Z.
- The IPDs of the input flow are given by z_i, i.e. $I_i^a = z_i$.

Low-Density Parity-Check Codes

We use a coset code construction [50] defined over a finite field with maximum likelihood decoding that is shown to achieve capacity ([44, 51]). These coset codes use the sparse graph code given by:

$$\mathcal{C} = \{\mathbf{x} : \mathbf{H}\mathbf{x} = \bar{\mathbf{s}}\} \tag{3.19}$$

where $\mathbf{H}$ is an m by n matrix and $\bar{\mathbf{s}}$ is called the *syndrome* of length m, and all operations are defined over the finite field $\mathbb{F}_Q$ (see Section 3.3).

Dithering

Each bit x_i is mapped to a uniform distribution by

$$y_i = \left[\frac{x_i}{Q} + \xi_i\right]_{mod\ 1} \tag{3.20}$$

where $\xi_1, \xi_2, \ldots, \xi_n$ are i.i.d uniformly distributed *dithers* on $[0, 1]$.

The y_i in equation (3.20) are uniformly distributed since

$$\begin{aligned} x_i &\in [0, Q-1] \\ \frac{x_i}{Q} &\in \left[\frac{0}{Q-1}, \frac{Q-1}{Q}\right] \end{aligned} \tag{3.21}$$

which is uniformly distributed for large values of Q. Dithering serves two purposes [52, 53, 54, 55]:

- It ensures that y_i is statistically independent of $\frac{x_i}{Q}$.
- By the Crypto lemma [55, Lemma 2], the equation (3.20) is uniformly distributed.

The second point is key to the next step that maps each uniformly distributed y_i to a value with the required distribution.

Shaping

A continuous flow should consist of i.i.d exponentially distributed IPDs, and in the discrete case, it should have a geometric distribution. In the earlier steps of encoding, uniformly distributed inputs were achieved. To get these inputs to have the desired distribution, a technique called *shaping* is used. This technique maps each uniformly distributed random variable to one with the desired distribution. Given a uniformly distributed variable Y on $[0, 1]$, a random variable Z is constructed with the desired cumulative distribution $F_Z(z)$ by (see Appendix C.1)

$$Z = F_Z^{-1}(Y) \tag{3.22}$$

The cumulative distribution F for an exponential random variable with rate λ is given by

$$F_Z(z) = 1 - e^{-\lambda z} \tag{3.23}$$

And given a uniformly distributed Y on $[0, 1]$,

$$Z = -\frac{\ln(1 - Y)}{\lambda} \tag{3.24}$$

Combining the dithering and shaping steps, a transformation matrix $\Psi(.)$

can be defined which maps from a uniformly distributed random variable $x_i \in \mathbb{F}_Q$ to an exponentially or geometrically distributed z_i:

$$\begin{aligned} Z_i &= \Psi(x_i) \\ &= F_Z^{-1}\left(\left[\frac{x_i}{Q} + \xi_i\right]_{mod\ 1}\right) \end{aligned} \tag{3.25}$$

which is a table with n rows and Q columns, having n functions $\Psi_i : \mathbb{F}_Q \to \mathbb{R}$. This lookup table (LUT) is shared between the encoder and decoder.

3.5.2 Decoding Structure

The decoder of [49] exploits the graphical structure of the probabilistic dynamics of a queuing system given in equation (3.8). Each arrival time a_i depends on the arrival of the previous packet a_{i-1} and the IPD associated with the Ψ transformation on x_i. Similarly, the service times depend on the current and previous departure times and the current arrival time (Section 3.1.3). The only parameters that the decoder has knowledge of are the departure times d^n, the sparse matrix $\mathbf{H}$, the syndrome $\bar{\mathbf{s}}$ and the transformation table Ψ_i. Due to the encoding structure, the decoder employs the standard low-complexity message-passing algorithms (the sum-product algorithm) on a Forney factor graph based on the graphical structure of the queuing channel and the encoding constraint given by equation (3.19) (see Figure 3.4) to produce low bit-error rates even at rates close to capacity.

In [56], to reduce decoding complexity from quadratic to linear, two techniques are employed: Little's law (Appendix C.2) and discretization of time in the continuous case where the times are binned into a discrete time scale with some small fixed resolution.

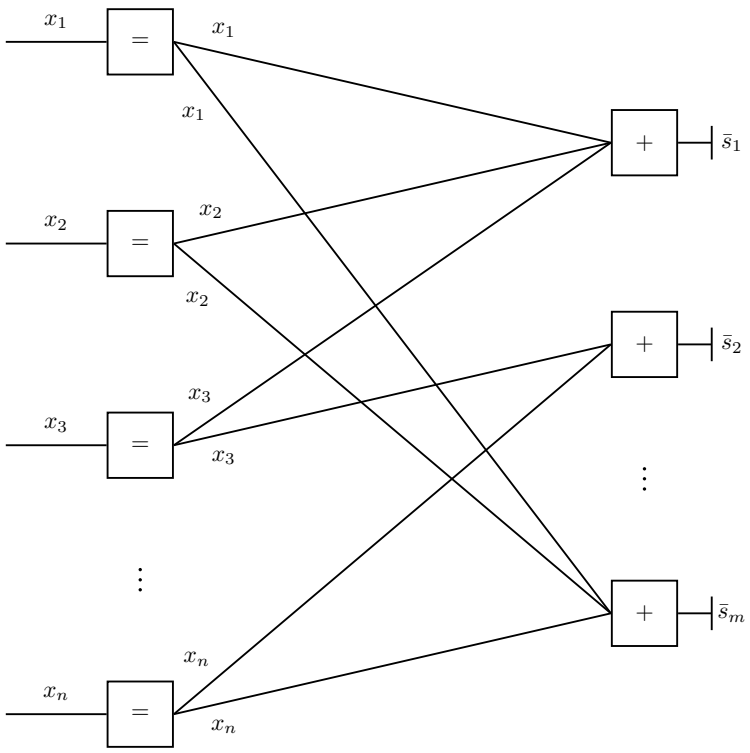


Figure 3.4: Forney factor graph of $\mathbf{Hx}=\bar{\mathbf{s}}$.

3.6 CSMA/CA in IEEE 802.11

The MAC layer of the IEEE Standard 802.11 for Wireless Local Access Networks (WLANs) uses CSMA/CA for collision avoidance [57].

The main idea behind the CSMA/CA collision avoidance strategy is that the channel is always sensed before any transmission. If the channel is sensed busy, the sender waits until the channel is idle and then goes through a random back-off period before retrying. The random back-off period ensures fairness among contending transmissions. More precisely, the transmission occurs only if, during a fixed period of time equal to a distributed interface state (DIFS), the channel is sensed idle. If the sensed channel is busy during or immediately after the DIFS, the station continues monitoring the

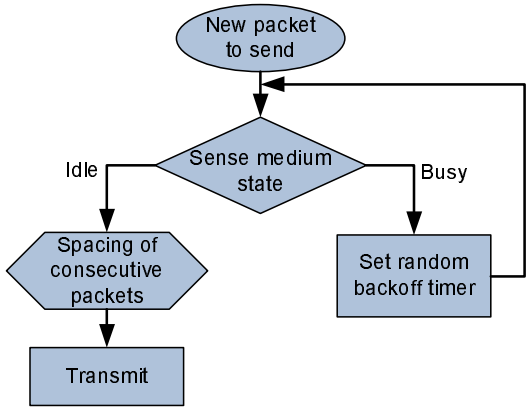


Figure 3.5: A flow chart depicting the flow of the CSMA/CA protocol that avoids collision in packet transmission using exponential back-off time. The sender only sends when the channel is sensed to be idle, otherwise it waits a random exponential time, ensuring fairness between various packet transmissions.

channel until the channel is sensed idle for a DIFS period. At this time, the CSMA/CA generates a random back-off interval before transmitting to avoid the possible collisions by minimizing the probability of collision with packets transmitted by other radios. Even if the medium is sensed idle during the DIFS period, a radio waits a random back-off time between consecutive transmission of two packets to avoid channel capture. The flowchart in Figure 3.5 shows the basic steps of the CSMA/CA at the transmitting radio.

Moreover for each packet transmission, an exponential back-off method is used, and to signal a successful packet transmission, an ACK is transmitted by the destination radio. This ACK is automatically transmitted at the end of the packet, after a short inter-frame space (SIFS). Since the combination of the SIFS and channel delay is shorter than DIFS, the channel cannot be detected idle for a DIFS until the end of an ACK.

CHAPTER 4

A TIMING CHANNEL SPYWARE FOR CSMA/CA PROTOCOL

4.1 Introduction

Rapid technological advances in wireless communication and the growth in the number and diversity of applications and services raise the demand for data-integrity and communication security. The complexity of the emerging applications and platforms introduces newer vulnerabilities that need to be carefully identified, analyzed, and addressed.

The emerging 802.xx standard protocols must be continually enriched with security features to meet the new and more critical application demands. For instance, the IEEE 802.22 is a cognitive radio standard being developed to bring broadband access to less populated rural areas by using vacant TV channels. To address the complex adaptive nature required by the technology, a cognitive radio device is typically implemented by a general purpose computer processor that also runs radio application software. As a result, this standard is susceptible to spyware and malicious software or hardware.

In this chapter, we will present a new transparent and robust covert communication scheme that can reliably leak the chip data to outside entities by exploiting the timing channel resulting from the inter-arrival times of the legitimate transmitted packets. Our scheme can be embedded within any medium access control (MAC) protocol that avoids collision in packet retransmissions by using an exponential back-off rule. For instance, the IEEE 802.11

Wireless Local Area Network (WLAN) standard's main MAC layer protocol, Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), is an example of a widely used protocol that employs such a rule.

The effects of the CSMA/CA's back-off rule perturbs packet timings by virtue of queuing and hence introduce a non-standard noisy channel that interferes with timing-based communication purposes. Although the maximum rate of information exchange of such channels (i.e. capacity) was characterized in [10], only recently have practical encoding/decoding methods to instantiate the theoretical limit with low (i.e. linear) complexity been introduced [49]. However, to date there have been no practical *covert* information encoding methods with low encoder/decoder complexity to instantiate the findings of [10]. Here we establish the first reliable covert timing channel that is also robust to the effects of CSMA/CA's back-off rule.

We identify the following performance metrics and evaluate them on our architecture:

1. Transparency of the spyware circuitry: This criterion pertains to the efficiency of implementation at the encoder to prevent the detection of the spyware circuitry.

2. Robustness: This criterion tests the robustness of the communication scheme to heterogeneous CSMA/CA effects which act as noise.

3. Transparency of the covert channel: This criterion measures the difficulty of the covert channel detection.

We evaluate criterion 1 completely theoretically. Criterion 2 is evaluated by simulating a wireless CSMA/CA architecture and testing the robustness of the decoder in different heterogeneous wireless conditions. Criterion 3 is con-

firmed experimentally using state-of-the-art covert timing channel detection methods.

The main contributions of this chapter are as follows:

- We present the first design of spyware integrated within the communication circuitry of wireless CSMA/CA that exploits the timing channel resulting from the inter-arrival times of packets to leak data from the chip.

- We introduce the first *covert* channel encoding that practically attains rates near the fundamental limits utilizing some methodologies developed in [49].

- Our spyware employs a low-complexity error-correcting code framework that is robust to timing perturbations resulting from the CSMA/CA's collision avoidance back-off strategy. This back-off strategy introduces a non-standard queuing noisy channel that interferes with timing-based communication purposes.

- We show the difficulty of the spyware detection, both at the hardware level and at the packet timing level, from both theoretical and practical perspectives.

- To substantiate our theoretical findings and to evaluate the overhead, the spyware is implemented and tested on the widely used WARP wireless radio platform [58].

The subject is organized as follows. The related literature was previously surveyed in Sections 2.1 and 2.4, and the preliminaries were described in Sections 3.1, 3.6 and 4.3. Section 4.3 introduces the theoretical results for covert channel embedding. In Section 4.4, we devise the new spyware embedding

algorithm. Detection of the spyware in circuitry and at the communication channel is presented in Section 4.5. Comprehensive experimental results for implementation, simulations, and detection are demonstrated in Section 4.6. We conclude in Section 4.7.

4.2 Mechanism and Coding Scheme

In this section, we will we summarize the methods for triggering the spyware and the mechanism and reason for using the coding scheme explained in Section 4.3.

4.2.1 Triggering the Spyware

The spyware is activated upon the arrival of a trigger, which may come from either from internal or external sources. The spyware does not need to be active all the time. We call the active duration of a spyware the *spying interval*. An internal spyware trigger comes from within the hardware. The two most obvious choices for an internal incitement are the states of the registers in the design and the clock. The states of the register can be utilized in a number of ways – for example, by arriving at a certain internal state of the communication controller, or upon reaching a certain counter state. Since we implement the trigger at the hardware level and integrate it within the design, it is impossible to detect these types of signals by studying the radio output.

The external triggers have to find a way for the outside source to reach the spyware. The most common method is via the communication channel. Note that other hardware interfaces to the outer world, such as sensors or power lines, can be used as extrinsic stimuli. The covert communication channel

can also be used for signaling the trigger. For example, the radio can be intentionally kept busy for certain time intervals. An important observation is that the external trigger does not have to be sent via the covert channel. For example, the receiver can be manipulated such that a certain combination of the input signals would not create an interrupt to the layers above. At the same time, the carrier sense signal can be raised high by the spyware, disallowing the legitimate messages to route via the communication channel. Unless the secret trigger combination is known and the radio is under physical tests observing its activity, detecting this type of external trigger is extremely difficult, if not impossible.

4.3 Covert Communication with Timings

Historically, timing channels are synonymous with covert channels [1, 4, 5, 6, 7, 8, 14, 27, 9]. Figure 4.1 illustrates a covert timing channel between an infected transmitter and a legitimate receiver. The *warden* (a.k.a eavesdropper) sees the exchange of packets but fails to realize the covert communication in the packet timings. On the other hand, the spyware receiver, which has side information pertaining to the parameters of the encoding scheme (see Section 3.5.2), can decode the message conveyed through the timings. As mentioned in Section 4.2, the CSMA/CA protocol results in random delays in the packet inter-arrival times. We will model the impact of the CSMA/CA as a first-come first-served (FCFS) queuing system. In particular, a special case of an FCFS queuing timing channel is the exponential server timing channel (ESTC)(Section 3.1.1), where the service times are explained in Section 3.1.3. We are using the ESTC as a model for the queuing channel for the CSMA/CA protocol since the ESTC has the smallest capacity among

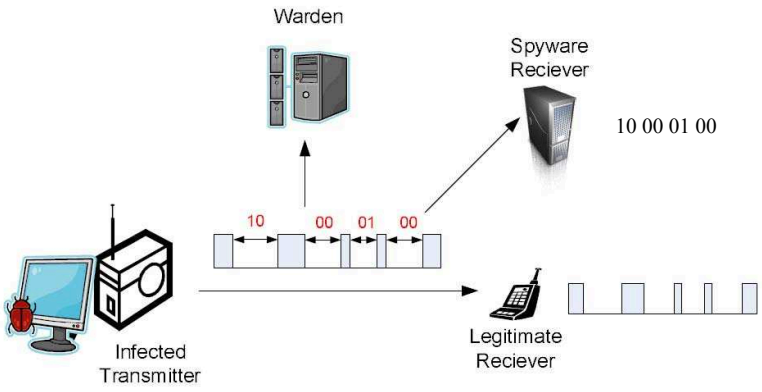


Figure 4.1: The warden monitors the communication between an infected transmitter and a legitimate receiver. Although the transmitter leaks spied information within packets timings, the warden is unable to decode the messages modulated by the packet timings.

all FCFS queuing channels with the same average service rate [59]; hence, a covert communication scheme that can survive the ESTC is most likely robust to other types of queuing noise. The maximum rates for an ESTC are given in Section 3.1.2; however, it was not clear until recently [49] how to develop practical codes that can withstand the noise from the queueing channels and approach the fundamental limits given by equation (3.1). In this chapter, we will use a similar code construction of the coding scheme, explained in Section 3.5, for the purpose of covert communication over a timing channel introduced due to queuing effects of CSMA/CA, as illustrated in Figure 4.2. The coding scheme involves an algebraic code construction that allows for a simple encoder, thus lending itself nicely to the design of in-circuit spyware. In our encoding structure, the encoder module uses the generator-representation of the code given by

$$\mathbf{x} = \mathbf{G}\mathbf{u} + \mathbf{p} \tag{4.1}$$

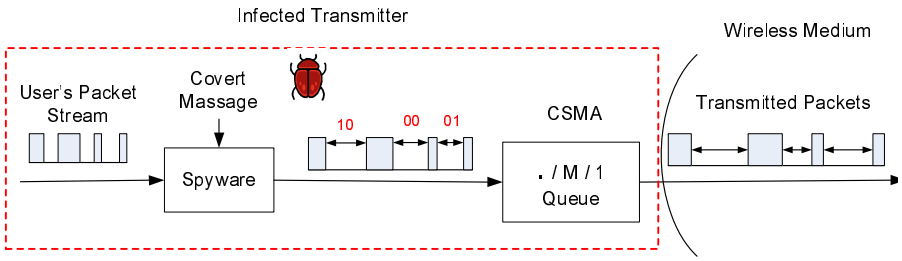


Figure 4.2: An infected transmitted transmits spied information in the form of bits which are encoded into the arrival times of the user packet stream using a spyware device. The packets are sent over the wireless medium which employs CSMA/CA which will be modeled as a $\cdot/M/1$ queue.

where a length-κ ($\kappa = n - m$) information vector is mapped to a length-n codeword vector $\mathbf{x}$ by use of an n by κ generator matrix G and a length-n vector $\mathbf{p}$. The decoder module resides in software at the spyware receiver, and upon receiving the packets, it decodes the transmitted message using the inter-packet times of the departure process (as depicted in Figure 3.1).

4.4 Implementation

The CSMA/CA which constitutes the communications MAC layer may be implemented in software or hardware. Either way, the MAC needs to interact with the physical layer (PHY) and the hardware components that provide the necessary information for the correct operation of the protocol. The PHY layer can be modified without affecting the upper layer MAC or notifying the changes to any of the higher layers, or even providing false information to them. Figure 4.3 shows different ways of interaction between the MAC and the PHY layer. The MAC queries the PHY for the status of the medium and timing information. As can be seen on the figure, the PHY can be modified to mislead the MAC by sending false timing information to it, or even notify the MAC that the medium is busy when it is idle. Perhaps the most important

means for spying the information at this layer is that the front transmission and reception of the packets must be done through the PHY.

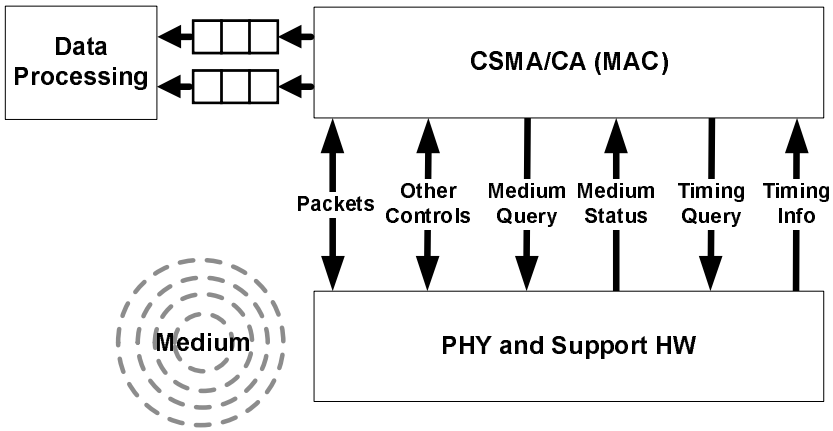


Figure 4.3: Interaction between the CSMA/CA which constitutes the MAC layer, and the physical layer.

In our implementation of the spyware, we modify the PHY so that the timings of the sent packets are altered in order to covertly send the stolen data to the outside world. To achieve this goal, we implement the encoder explained in Section 4.3 using the generator matrix given in equation (4.1). The block diagram of the encoder circuit is shown in Figure 4.4.

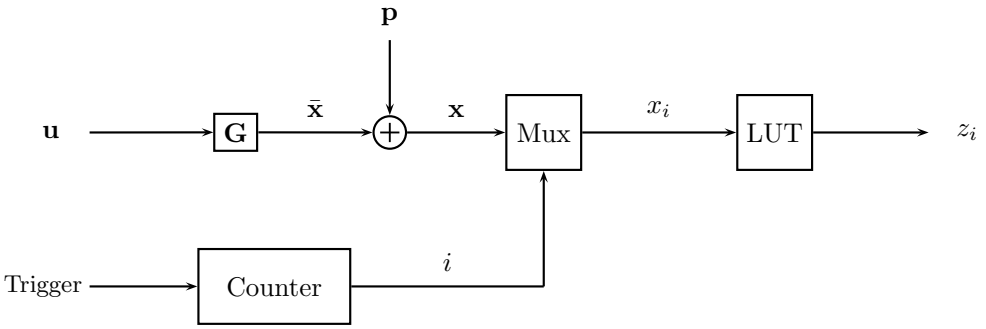


Figure 4.4: A block diagram showing the encoding circuit of the spyware.

Both the generator matrix $\mathbf{G}$ and the length-n vector $\mathbf{p}$ are hard-coded in the PHY layer. $\mathbf{G}$ is generated for a specific finite field $\mathbb{F}_Q$. Algorithm 1 shows the steps for constructing the circuit representing multiplications by $\mathbf{G}$. First,

a template table Ψ is constructed for $\mathbb{F}_Q$, which has Q entries representing small circuits that perform addition and multiplication of constants from 0 to $Q-1$ with a variable in $\mathbb{F}_Q$. The circuits in the template table are simple XOR gates, constants, or wires. The circuit is generated by the loop in Lines 5 to 7 in the algorithm.

Algorithm 1 Algorithm for Encoding with the Generator matrix **G**

Require: Q: the field size to be used, **G**: $n \times \kappa$ matrix over $\mathbb{F}_Q$, Ψ: $Q \times Q$ matrix over $\mathbb{F}_Q$.

1: Input: **u**, the input vector to the circuit of size κ
2: Output: $\bar{\mathbf{x}}$, the output vector of the circuit of size n
3: **for** $i = 1$ to n **do**
4: $\quad \bar{X}(i)=0$
5: $\quad$ **for** $j = 1$ to κ **do**
6: $\quad\quad \bar{X}(i)=\bar{X}(i) \oplus \Psi(G(i,j), u(j))$
7: $\quad$ **end for**
8: **end for**

The final circuit is just a network of XOR gates. The output of this circuit is then added to the constant vector **p**. This step is also hard-coded in the circuit by inverting some of the bits of $\bar{\mathbf{x}}$ to generate **x**. Each element in **x** represents an index for a specific inter-arrival time value stored in the LUT. The elements of **x** are traversed using a counter, which is reset when the trigger is raised. Every time a packet is successfully transmitted, the counter is incremented for generating the next inter-arrival time.

4.5 Spyware Detection

The spyware can be distinguished either by realizing the presence of the spyware circuitry on the chip or by detecting the covert timing channel. We next show that neither of these two measures are feasible.

4.5.1 Detection of Spyware Circuitry

It should be noted that the above circuitry of Section 4.4 is mainly combinational and can be interwoven and concealed within the sea of gates in the PHY and the communication signal processing circuitry. Only a few latches are used for the counter and the LUT, which can also be effectively embedded within the regular combinational logic in the circuit. The overhead presented by the encoder is a function of the size of $\mathbf{G}$. However, while a smaller $\mathbf{G}$ provides a smaller overhead, there is a tradeoff between the size of $\mathbf{G}$ and the accuracy of decoding the encoded message. An example implementation of the encoder and the tradeoff between the overhead and accuracy of the decoder is shown in Table 4.1 of Section 4.6.1.

Our implementation method renders the removal attack almost impossible: it is well known that reverse-engineering the functionality of the circuit by accessing the gate-level information and even the layout information is not practically possible [34]. This is because the synthesis step of the design which transforms the functional specifications into the gate-level consists of several optimizations that do not maintain the functional relationships. Note that circuit verification cannot reveal the spyware either. The circuit is verified for having the same functionality as the protocol specification by means of checking the integrity of the protocol states. Verification does not generally detect (verify) the unknown added states. Furthermore, the verification criteria are often set by the designer, who is also the spyware implementer in our case and can effectively hide the spyware from the other verifiers.

Note that the currently available Trojan detection methods are unable to detect our circuit-level manipulations and spyware. All of the Trojan detec-

tion methods are based on the assumption that the original circuit's hardware description (the netlist or GDSII files that the designer sends to the foundry) are available at the detector [33]. The adversarial model is that the foundry is changing the circuitry by adding components. The designer gets the ICs back and can test to see if the IC is implementing the original design. In our model, the designer is modifying the circuit herself and ultimately decides the number and arrangement of gates. Furthermore, given the complexity of the state-of-the-art designs, reverse-engineering is not practical.

Let us assume a scenario in which another entity designs a circuit with exactly the same functionality. Even this circuit would not provide a good comparison point, because of the degree of freedom in selecting the design components. In other words, there is no minimum number of gates or particular arrangement of them that can be computed for a given specification, so there would be little relevance between the new entity's design and the original circuitry without the spyware. Thus, this attack is not at all valid on our spyware.

4.5.2 Detection of Covert Communication

Earlier measures against the timing channel have focused on disruption techniques to eliminate the covert channel. Notably, our wireless timing channel is susceptible to injection of false packet events by an external entity. For instance, a second node that transmits cover traffic claiming the identity of the node containing the spyware. However, such measures come at the cost of the loss of the legitimate user's channel capacity and the resources of the attacker. Another countermeasure focuses on identifying the covert channel. If the modifications made to the timings of the flow are visibly different to an

observer, as in our case, once the presence of spyware deduced covert channels is detected, the observer will eliminate the covert channel by eradicating the spyware.

We check the detection resistance of our scheme against the current state-of-the-art covert timing channel detection tests – the regularity test, the Kolmogorov-Smirnov test and the entropy detection test, explained in Section 3.2. In Section 4.6, we will show that none of the above tests can reliably detect the presence of covert communication under certain conditions.

4.6 Experimental Results

The hardware-based covert channel spyware was implemented using the Wireless Open-Access Research Platform (WARP) [58], which is a scalable, extensible programmable wireless platform that enables advanced wireless network prototyping. The encoder is implemented in hardware, and its overhead when integrated within the WARP platform is reported for different encoder sizes in Section 4.6.1. The decoder is implemented in software using Matlab, since we do not study the spying receiver's architecture. We also perform network level simulations, where the communication among the different nodes introduces interfering network traffic. The goal is to evaluate the accuracy of decoding in the presence of random interference as well as to study whether the covert flow can be detected by traffic analysis. Our studies require two types of experimental setups: hardware setup and software simulation setup. The details of the hardware setup are as follows:

- The Vertix2P7 FPGA on the WARP platform is used for the encoder implementation. The spyware encoder is implemented in Verilog and integrated within the physical layer blocks on the FPGA.

- CSMA/CA MAC layer protocol is implemented on the PowerPC hard-core processor embedded in the WARP FPGA.
- The physical layer blocks with the exception of the analog front-end are all implemented on the WARP FPGA.
- A field size $Q = 4$ and $GF(4)$ operations are used for encoding.
- Three different sizes of the encoding matrix ($\mathbf{G}$) are implemented: 20×8, 40×16, and 80×32.

The details of the simulation setup are as follows:

- Aside from the spyware encoder and its intended receiver, there are K other nodes (possibly interfering) in the wireless network.
- The transmitting nodes send out the packets according to a Bernoulli process, with burstiness probability B_p.
- The spyware information is encoded in the timings of the packets, as discussed in Section 3.5.1, for a fixed encoder rate of $\frac{\kappa}{n} \log_2 Q = 0.4$ bits/packet, and $\lambda = 0.225$ packets/sec.
- The collision management scheme in the CSMA/CA protocol introduces queuing of the packets – thus acting as a queuing timing channel, as discussed in Section 4.2.
- The spyware decoder senses the transmission times of the wireless radio, which act as the departure process of the queue, and uses the iterative probabilistic decoder discussed in Section 3.5.2.

As we discussed in Section 4.1, the performance of our spyware is evaluated with regards to three criteria:

- Transparency of the spyware circuitry.
- Robustness of the communication scheme to heterogeneous CSMA/CA effects.
- Transparency of the covert communication scheme.

4.6.1 Transparency of the Spyware Circuitry

In this subsection, we study the transparency of the spyware circuitry. This is done by evaluating the cost of integrating the hardware encoder on the FPGA. The field size (Q) is set to 4 for the implementation. A template table implementing both addition and multiplication in $GF(4)$ is constructed and a C program is written to generate the encoder corresponding to any matrix $\mathbf{G}$ of arbitrary size in Verilog. The encoder is then inserted within the WARP platform physical blocks on the FPGA.

Table 4.1: The overhead introduced by the encoder implementation on a WARP node for three encoding matrix sizes.

Original		Small		Medium		Large	
		#	Overhead	#	Overhead	#	Overhead
Slices	24,518	24,531	0.05%	24,643	0.50%	24,993	1.94%
Gates	18,741,445	18,741,598	0.00%	18,742,966	0.01%	18,747,184	0.03%

Table 4.1 shows the overhead introduced by the encoder implementation on a WARP node. The first column in Table 4.1 is labeled *original* and presents the original area of the wireless node components on the WARP. The area is given in terms of the number of slices used on the FPGA and the estimated number of gates in the design. The next two columns show the area of the wireless node on the FPGA when we integrate an encoder with small matrix $\mathbf{G}$ of size 20×8. The percentage overhead in area in terms of FPGA slices is

0.05%, while the overhead in terms of the extra number of gates is negligible. The next four columns show the area and the percentage overheads when using a matrix $\mathbf{G}$ with size 40×16, and 80×32, respectively. Naturally, the overhead increases as the size of the matrix increases. Due to the very small area overhead, the integration of the encoder has no impact on the overall power of the WARP platform. Also, the encoder does not affect the overall delay of the digital circuitry implementing the radio because it does not affect the critical path of the digital design. Again we emphasize that the overhead is only measurable by the original designer, and the original unaltered files are never shared with any entity, including the foundry.

4.6.2 Robustness of the Communication Scheme to Heterogeneous CSMA/CA Effects

In this subsection, we study the robustness of the spyware communication in terms of the symbol error rate (SER) at the decoder. The SER is affected by the queuing effects introduced by the CSMA/CA due to varying network conditions. We generate multiple levels of interfering traffics on the network by simulating different number of nodes that communicate in the range of both the sender and receiver. It should be noted that the only purpose of the network traffic is to introduce interference that can affect the times of arriving packets carrying the spyware information. Table 4.2 shows the average SER at the decoder. These average SER values are shown for various numbers of interfering nodes, namely values of $K \in \{5, 10, 15\}$, as well as burstiness probability $B_p \in \{0.4, 0.6\}$ for three encoding matrix sizes $(\kappa, n) \in \{(8, 20); (16, 40); (32, 80)\}$ corresponding to small, medium, and large on the table, respectively.

Although the SER values are only on the order of 10^{-2}, we bring attention

to the specific context of this application: a short code-length, externally triggered, and hardware implementation. Since this approach uses linear coset codes (which have extremely small undetected error probability [60]), an error event is essentially equivalent to the condition $\mathbf{H}\hat{\mathbf{x}} \neq \bar{\mathbf{s}}$. Thus, the spyware receiver can simply externally retrigger the transmission at a random time in the future. As can be expected, Table 4.2 shows that as the length of the code (size of the encoding matrix) increases, the average symbol error rate at the decoder improves.

Figure 4.5 illustrates boxplots of the SER across multiple trials. The lower edge of the rectangular plot corresponds to $e_{.25}$, the 25th percentile; the upper edge pertains to $e_{.75}$, the 75th percentile. The 50th percentile, or median, appears as the horizontal line between the lower and upper bounds. The upper 'tail' of the box corresponds to the vertical line extending beyond the 75th percentile, which is of length $1.5(e_{.75} - e_{.25})$. All outliers extending beyond the 'tail' are explicitly drawn with the '+' symbol. We see from Figure 4.5, the statistical structure of the SER is highly concentrated near 0, in such a way that cannot be evidenced by merely the mean SER. Specifically, the 25th percentile of the errors is 0. In terms of externally triggered retransmissions, this means that simple "majority-rules" decoding schemes will work particularly well with this approach. More specifically, while the average SER is of the order of 10^{-2}, around 25% of the time, the decoder receives symbols with zero error and thus a majority decoding rule after 2 or 3 transmissions will result in perfect decoding.

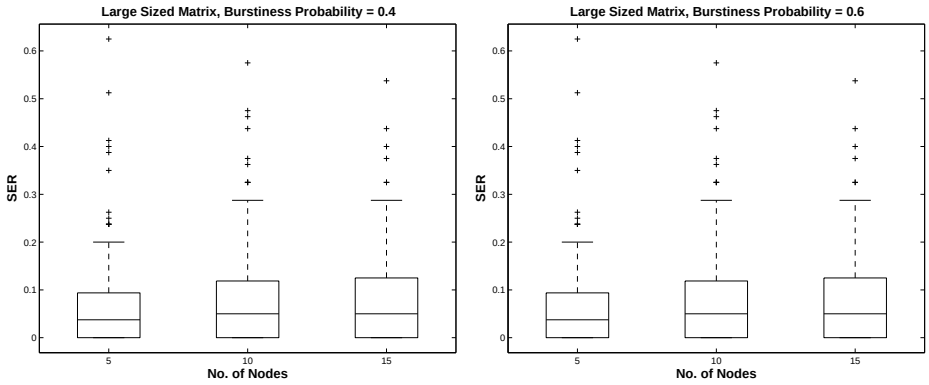


Figure 4.5: Boxplots of the symbol errors across multiple trials, using the large encoding matrix with burstiness probabilities $B_p \in \{0.4, 0.6\}$ with 200 runs.

Table 4.2: Average SER of the spyware decoder vs. number of congesting nodes. The study is done for different probabilities of sending packets over the wireless medium, and the matrix sizes.

Number of nodes	Burstiness prob. = 0.4			Burstiness prob. = 0.6		
	Small	Medium	Large	Small	Medium	Large
5	0.16	0.08	0.08	0.13	0.09	0.05
10	0.14	0.10	0.09	0.14	0.10	0.07
15	0.14	0.09	0.08	0.13	0.09	0.08

4.6.3 Transparency of the Covert Communication Scheme

In this subsection, we study the detectability of the covert information hidden in the network traffic by generating regular traffic and comparing the traffic with spyware covert channels for various network settings. We use the different traffic flows to generate inter-arrival times that are used as inputs to three covert channel detection tests: the entropy detection, Komolgorov-Smirnov and regularity tests of Section 3.2.

Table 4.3: Regularity test, using a window of size 100 packets, fails to detect the presence of covert communication.

	Small		Medium		Large	
Packets	Covert	Non-covert	Covert	Non-covert	Covert	Non-covert
1000	0.36	0.3	0.36	0.3	0.35	0.3
2000	0.4	0.37	0.38	0.3	0.37	0.3

Table 4.3 gives the results for the regularity test with non-overlapping windows of 100 packets over a collection of 1000 and 2000 packets. We randomly injected an external trigger activating covert transmission of $n = 20$ (small), $n = 40$ (medium), or $n = 80$ (large) consecutive packets. As mentioned earlier in Section 3.2.1, the regularity test of Cabuk et al. [14] examines whether the traffic is more regular than expected with legitimate traffic. Specifically, this test checks whether the variance of the inter-arrival times remains small. As seen in Table 4.3, the regularity test completely fails to detect the covert communication.

The results for the corrected conditional entropy test (CCE) and the entropy test (EN) are given in Table 4.4 and Table 4.5, respectively, over a collection of 1000 and 2000 packets. Again we randomly injected an external trigger activating covert transmission of $n = 20$ (small), $n = 40$ (medium),

Table 4.4: Detection (P_D) and false alarm (P_{FA}) probabilities for using the corrected conditional entropy (CCE) on the packet inter-arrival times.

Number of packets	P_D			P_{FA}		
	Small	Medium	Large	Small	Medium	Large
1000	0.043	0.050	0.148	0.011	0.008	0.009
2000	0.058	0.082	0.178	0.011	0.008	0.010

Table 4.5: Detection (P_D) and false alarm (P_{FA}) probabilities for using the entropy test (EN) on the packet inter-arrival times.

Number of packets	P_D			P_{FA}		
	Small	Medium	Large	Small	Medium	Large
1000	0.046	0.056	0.066	0.010	0.090	0.090
2000	0.061	0.064	0.090	0.009	0.011	0.011

or $n = 80$ (large) consecutive packets. The false alarm event occurs when all packets transmitted do not have a trigger but are erroneously deemed to have been containing spyware covert timing information. The rightmost column illustrates the false alarm probability (P_{FA}) for the 1000 and 2000 packet transmission scenarios. Meanwhile, the detection event pertains to the scenario where an external trigger is randomly injected into the packet stream and it is successfully detected using the hypothesis testing procedure. Detection probabilities (P_D) are given for small, medium, and large matrix sizes. The threshold was decided at the 99% confidence interval. We can see that these tests either fail to detect the covert transmission or have very low detection rates.

Table 4.6 gives results for the Komolgorov-Smirnov test where, over a collection of 250, 500, 1000 and 2000 packets, we randomly injected an external trigger activating covert transmission of $n = 20$ (small), $n = 40$ (medium), or $n = 80$ (large) consecutive packets. In all cases, the threshold for the

hypothesis testing procedure was the deviation of the Komolgorov-Smirnov statistic, given by equation (3.10), beyond the 99% confidence interval. Over 1000 and 2000 packets, this approach is successfully undetectable: P_D and P_{FA} are close to one another in virtually all scenarios. We note that for larger matrix sizes, the deviation between P_D and P_{FA} increases as does the probability of detection, thus illustrating a tradeoff between error tolerance robustness and detection probability. Reducing the interval to 250 and 500 makes the transmission more visible, although the detection rates remain low, except when $n = 80$ over 250 packets. In this case most of the covert transmissions were detected.

Table 4.6: Detection (P_D) and false alarm (P_{FA}) probabilities for using a Komolgorov-Smirnov test on the packet inter-arrival times.

Number of packets	P_D			P_{FA}		
	Small	Medium	Large	Small	Medium	Large
250	0.0357	0.2820	0.9022	0.0105	0.0117	0.0133
500	0.0164	0.0280	0.3288	0.0115	0.0103	0.0106
1000	0.0114	0.0195	0.0258	0.0108	0.0107	0.0107
2000	0.0106	0.0122	0.0206	0.0105	0.0114	0.0107

4.7 Conclusion and Future Work

We have shown a new methodology for the design and implementation of spyware communication circuits that exploits the CSMA/CA MAC protocol properties to covertly leak the chip data to an intended spyware receiver. Our proposed scheme is robust to the collision avoidance techniques in the CSMA/CA protocol of the MAC layer. We interweave the encoder in the states and transitions of the wireless transmitter pre-synthesis, such that distinction and uncoupling of the spyware from the radio's physical layer is

impossible. The spying act is initiated upon an occasional external or internal trigger. The proof-of-concept implementation is demonstrated on the widely used Wireless Open-Access Research Platform (WARP). Three metrics were used for evaluation of the spyware performance: 1) efficiency of the encoder implementation, which was theoretically demonstrated; 2) robustness of the communication method to CSMA/CA effects, which was done by simulating the wireless architecture across a range of parameters; and 3) difficulty of covert channel detection that was shown to be resilient against the known covert timing channels detection methodologies. While there exists a trade-off between accuracy, overhead and detectability, our experimental results show that our implemented spyware simultaneously meets the above desired metrics.

Some possible future directions include detection of the "trigger" time, or the time at which covert communication begins. The detection can potentially be done by the use of "change-point detection" approaches [61, 62] that attempt to find the location in time where inter-arrival times drawn from a distribution F_0 switch to a distribution F_1.

However, if the circuit-level design criterion permits more sophisticated shaping schemes, even the change-point detection will fail. For instance, the encoder's shaping technique and dithering technique (equation (3.25)) of Section 4.3 can be used to design provably undetectable covert communication schemes. At a high level, the idea is that the dithering scheme discussed in Section 4.3 has two important properties: It follows theoretically that any statistical test that models normal traffic as i.i.d. with inter-packet timings of distribution F_Z will provably fail. More specifically, the inverse CDF, $F_Z^{-1}(\cdot)$, can be simply shaped to any desirable legitimate traffic model, as long as the inter-packet times are i.i.d.

CHAPTER 5

WATERMARKING: TRACING TRAFFIC FLOWS THROUGH THE NETWORK

5.1 Introduction

Detecting correlated network flows, a.k.a flow linking, is a crucial technique in traffic analysis, specially in protecting against cyber-attacks, which have continued to increase and cause financial damages [63]. By linking flows, an attacker can defeat anonymous system such as Tor[1]. Moreover, flow linking can help expose a *stepping stone* attacker who uses intermediate hosts to attack a network system.

Many network intruders attack indirectly by relaying their traffic through a series of intermediate hosts, known as *stepping stones*, to mask the source of the attack. Often they encrypt their traffic by using secure protocols such as SSH and telnet. An intruder, at a system, will set up a connection to an unsecured host, and then launch attacks on a third system. The examination of the attacking traffic would consider the unsecured intermediate host to be the source, preserving the anonymity of the intruder. Since the contents of packets are often encrypted or can easily be modified, studies have looked into the characteristics of the traffic (packet timings, packet sizes, etc.) between suspected connected flows, since there would be similar characteristics in flows belonging to the same chain, however it may be routed through the network. These features of the traffic can help correlate flows to help trace

[1]http://www.torproject.org

the true source of the network traffic.

Recent work has shown that similarities in communication patterns such as packet sizes and timings may be used for flow linking [16, 17, 18, 19]. Two types of traffic analysis techniques, *passive* and *active*, are commonly used for flow linking.

Passive analysis [15, 16, 20, 21] makes use of the original characteristics in a packet flow, such as correlating periods of activity or idleness [16] or finding a correlation measure based on the inter-packet delay (IPD) of the flows [21]. These schemes rely only on the original characteristics in the observed flows, which can be easily weakened by timing perturbations during network transmissions. Therefore, successful detection using these schemes, often require analysis of a large number of packets.

Active analysis schemes are able to perform reliable detection with shorter flows by injecting patterns such as watermarks into the flows [17, 18, 19]. These schemes are also known as one-bit watermarks, since the watermark contains no hidden messages and the decoder is only interested in detecting the presence or absence of the watermark. The robustness and low complexity of these watermarking have led to a growing research interest.

Flow watermarking approaches are mainly classified into two classes; *interval*-based and *IPD*-based.

In interval-based schemes [18], the flow is first divided into fixed lengths of time intervals. Then timing patterns of all packets within an interval are reshaped to encode the watermark. Since the watermark pattern is embedded within multiple packets, interval-based schemes are robust to packet losses. However, shifting packets in groups causes visible 'artifacts' that in turn can reveal the embedded watermark. Kiyavash et al. [3] showed that interval-based watermarking schemes are vulnerable to the *multi-flow attack*,

whereupon observing a few flows, the attacker can detect the watermark as abnormally large number of empty time periods, that were created during the embedding process.

Fortunately, the alternative solution, IPD-based flow watermarks resists this attack. In IPD-based schemes [17, 22, 19], watermark bits are embedded into the inter-arrival times of packets in a flow-dependent manner. Thus, it is hard for the attacker to find noticeable artifacts even with access to many watermarked flows. However, many schemes do not consider packet losses that are possible in a network. The drawback of per-packet embedding is that it requires synchronizations of packets for watermark detection, and therefore packet losses could cause severe decoding errors.

This chapter proposes a novel IPD-based flow watermarking scheme that can withstand packet losses. We embed the watermark within the IPDs using quantization index modulation (QIM) [53], which is invisible even under the multi-flow attack. To withstand packet losses that may lead to both deletion and substitution errors, we develop a hidden Markov model (HMM) for our channel which has dependent deletion and substitution error states. At the detector, we use a maximum likelihood decoding algorithm, paired with a forward-backward algorithm, for deriving the posterior probabilities. Through simulations, we show that our scheme performs well in presence of deletion and substitution errors, while not introducing any visible artifacts. It is noteworthy that the substitution errors are introduced not only by network jitters but also by packet deletions within the network that desynchronize the watermark, because packet losses merge consecutive IPDs.

5.2 System Model

In this section, we describe the components of the proposed scheme. Figure 5.1 depicts our embedding and extraction procedures, starting with an input flow with IPDs $\mathbf{I}$.

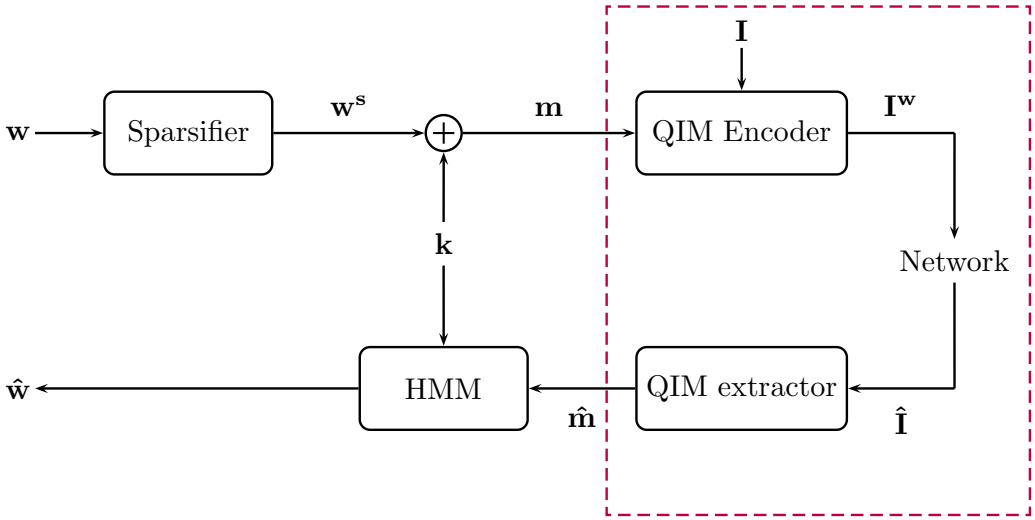


Figure 5.1: Overview of our watermarking scheme.

Our encoding scheme consists of a *sparsifier* and *QIM embedder*, which modifies the IPDs of the original flow to embed the watermark. As the watermarked flow passes through the network, the IPDs are modified by network jitters, and some packets are dropped. The decoding scheme consists of two steps: the *QIM extractor* and *HMM decoder*, which analyze the received flow and extract an estimate of the original watermark. The decoding scheme decides if the original watermark is present in the received flow and if the estimated watermark is similar to the original by the means of a threshold.

The following notations are used throughout the thesis.

- $\mathbf{w}$: The original watermark $\mathbf{w}$ is a binary sequence of length N.

- $\mathbf{w^s}$: A sparsified version of $\mathbf{w}$ extended to length ηN for an appropriate integer η.
- $\hat{\mathbf{w}}$: The estimate of $\mathbf{w}$ extracted at the HMM decoder.
- $\mathbf{k}$: a length M pseudo-random binary sequence (key) available both at the embedder and HMM decoder.
- $\mathbf{m}$: The sequence of length M that is embedded into the flow's IPDs by the QIM embedder. Henceforth, it shall be referred to as the *marker*.
- $\hat{\mathbf{m}}$: The estimate of $\mathbf{m}$ at the QIM extractor.
- $\mathbf{I}$: The IPD sequence in the original flow.
- $\mathbf{I^w}$: The IPD sequence after the QIM encoder embeds $\mathbf{m}$ within $\mathbf{I}$.
- $\hat{\mathbf{I}}$: The IPD sequence received after transversing the network.

5.2.1 Sparsifier

In the first step of embedding, the binary watermark $\mathbf{w}$ is sparsified, by mapping each bit of $\mathbf{w}$ to a longer binary sequence of length η according to a deterministic sparsification table. The resulting sequence $\mathbf{w^s}$, is xored with a key $\mathbf{k}$ resulting in $\mathbf{m}$ which is embedded into the flow $\mathbf{I}$ using QIM.

The sequence $\mathbf{k}$ serves as a 'helper' for watermark synchronization [64]. The intuition is that changes in the 'pattern' of $\mathbf{k}$ provide information about deletions that have occurred. For instance, consider the case when $\mathbf{w^s}$ is all zeros: if $\mathbf{k}$ is 0111001001, then $\mathbf{m}$ is also 0111001001. If 01100101 was received, it is easy to conclude that a 1 in the second run and a 0 in the fifth run were deleted.

In practice, any 1's in $\mathbf{w^s}$ will create a bit flip in $\mathbf{k}$. Furthermore, the network could introduce more substitution errors. Therefore to retain the patterns of $\mathbf{k}$ necessary for synchronization, we need to ensure that $\mathbf{w^s}$ is sparse. It is noted that the choice of sparsification factor trades on the rate of the watermark and the detection performance. However, in most flow linking applications, rate is not of concern and a large sparsification factor may be picked.

The sparsification density f is another parameter of this scheme also available during decoding. The density f is defined as the density of 1's in the sequence $\mathbf{m}$, given by

$$f = \sum_{i=1}^{\eta N} \frac{m_i}{\eta N} \quad \text{where } m_i = w_i^s \oplus k_i \tag{5.1}$$

5.2.2 QIM Embedder and Extractor

In the next step, we modify the IPDs in the original flow using QIM watermarking. We pick a quantization step-size Δ, which is the distance between two consecutive '0' quantizers. The IPD I_i is changed to

$$I_i^{\mathbf{w}} = \begin{cases} c\Delta & \text{if } m_i = 0 \\ (c + 0.5)\Delta & \text{if } m_i = 1 \end{cases} \tag{5.2}$$

As packets can only be delayed, we choose c to be the smallest integer such that the change in $I_i^{\mathbf{w}}$ would delay the ith packet.

The flows with IPDs $I_i^{\mathbf{w}}$, enters the network where it is subjected to network jitters and packet losses. It is received at the decoding side as $\hat{\mathbf{I}}$ and the following QIM decoding function is used to recover the embedded bits $\hat{\mathbf{m}}$.

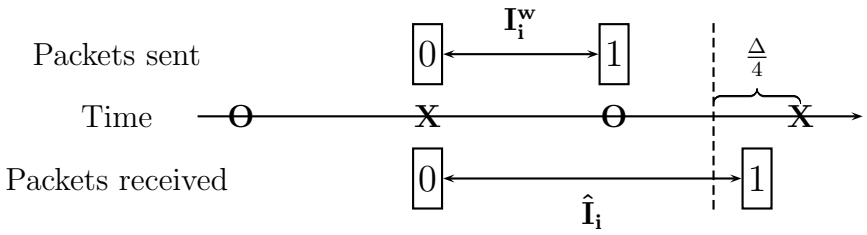


Figure 5.2: An example for substitution errors caused by IPD jitters. '**x**'s are '0' quantizers and '**o**'s are '1' quantizers. The bit embedded on I_i is 1, but the decoded bit from $\hat{I}_i$ is 0.

$$\hat{m}_i = \begin{cases} \mod(\lfloor \frac{2\hat{I}_i}{\Delta} \rfloor, 2) & \text{if } \frac{2\hat{I}_i}{\Delta} - \lfloor \frac{2\hat{I}_i}{\Delta} \rfloor \leq 0.5 \\ \mod(\lceil \frac{2\hat{I}_i}{\Delta} \rceil, 2) & \text{if } \frac{2\hat{I}_i}{\Delta} - \lfloor \frac{2\hat{I}_i}{\Delta} \rfloor > 0.5 \end{cases} \tag{5.3}$$

In Figure 5.2, the IPD $I_i^{\mathbf{w}}$ encodes a bit 1. If the network jitter increases the IPD by less than $\Delta/4$, the IPD $\hat{I}$ will still be close to the '1' quantizer (represented by '**o**'). But if the network jitter is greater than $\Delta/4$ but less than $3\Delta/4$, then $\hat{I}$ will be closed to the '0' quantizer (represented by '**x**'), causing a decoding error. Therefore for some positive integer c,

$$\hat{m}_i \neq m_i \quad \text{if} \quad \frac{(4c-3)\Delta}{4} < \mod(\hat{I} - I_i^{\mathbf{w}}) \leq \frac{(4c-1)\Delta}{4} \tag{5.4}$$

5.2.3 HMM Decoder

At the HMM decoder, we first develop a hidden Markov model of the channel. Based on this model, the posterior probabilities $P(\hat{\mathbf{m}}|w_j)$ are calculated. Watermark bits w_j are subsequently decoded as $\hat{w}_j$ using maximum likelihood (ML) decoding.

Note that in Figure 5.1 the QIM embedder, the network, and the QIM extractor could be regarded as a communication channel (within the dashed

box) with two types of errors: *substitutions* and *deletions*. The substitution error refers to a bit flip due either to network jitters or deletions that result in the merger of IPDs. It has been shown that the network jitter may be approximated as independently identically distributed Laplace random variables with zero mean and a standard deviation of σ [19] . Since during QIM decoding we map each IPD to its closest quantizer, equation (5.4) provides the range at which a substitution error will occur. Since there is a lower probability of larger jitter values, for simplification we consider that any jitter over $\Delta/4$ would possibly result in a substitution error (see Figure 5.2). In general, the probability of a substitution error caused by jitters is estimated as[2]

$$P_s \approx 2 - (1 + sgn(\frac{\Delta}{4}) \cdot (1 - e^{\frac{-|\Delta|}{2\sqrt{2}\sigma}})). \tag{5.5}$$

Transition Probabilities

The deletion error refers to a bit lost due to a packet drop. Davey and Mackay [64] proposed a probabilistic decoding scheme to handle independent deletion, insertion and substitution errors in a communication channel. Our channel differs from the model in [64] as a single packet drop results in the merger of two consecutive IPDs. For instance, in Figure 5.3, the deletion of Packet 1 merges the bits m_1 and m_2 into $m_1 \oplus m_2$. This causes a deletion of m_1 and possibly a substitution error of m_2 in the received stream. Therefore, we develop a new channel model to handle dependent substitution and deletion errors. Without loss of generality, we consider the packet deletion probability P_d to be identical for all packets, and assume that Packet 0 is

[2] $sgn(\cdot)$ denotes the sign function.

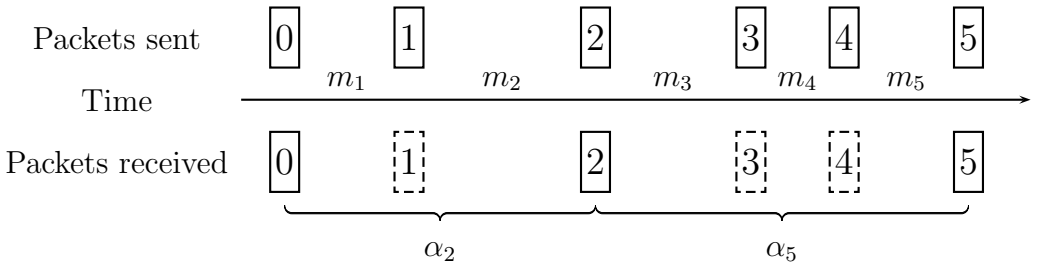


Figure 5.3: Merging of IPDs when packets are dropped.

always synchronized[3].

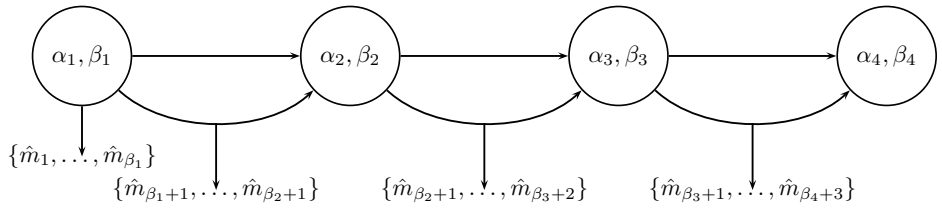


Figure 5.4: The HMM of the watermark-over-network channel.

Figure 5.4 depicts the HMM model of our channel. The hidden states are defined as (α_i, β_i), where $i = 1, 2, \cdots M$. The notation β_i is the *drift* of packet i in the received flow. If k packets were dropped before packet i, then β_i equals $-k$. For example, in Figure 5.3, Packet 2 has drift of -1 due to the loss of Packet 1, and Packet 5 has drift of -3 because of the loss of three previous packets. The symbol α_i denotes the *accumulated* bit when sending Packet i. Again in Figure 5.3, before transmitting Packet 2, the bit in the current IPD is a merger of m_1 and m_2, that is $\alpha_2 = m_1 \oplus m_2$. Similarly, $\alpha_5 = m_3 \oplus m_4 \oplus m_5$. In general, α_i equals to $\sum_{j=r+1}^{i} m_j$, where r is the index of the last successfully received packet before Packet i. The observed states of Figure 5.4 are the received bits $\hat{\mathbf{m}}$. Note that the watermark extractor

[3]A scenario when the first packet is lost can easily be dealt with by repeating the watermark in the network flow.

receives $\max\{\beta_i + i - 1, 0\}$ bits in total before Packet i is sent.

Posterior probabilities of this HMM model are required for the ML decoder that extracts the watermark estimate $\hat{\mathbf{w}}$. For this derivation, the transition probability distribution, $P(\hat{m}_{i-1+\beta_{i-1}}^{i-1+\beta_i}, \alpha_i, \beta_i | \alpha_{i-1}, \beta_{i-1})$, is required. These transition probabilities can be calculated in terms of density f, deletion probability P_d and the substitution probability P_s of equation (5.5) (we need to consider the possibility of occurrence of a substitution event when the current packet is successfully received).

We use an example to illustrate how this quantity is computed. In Figure 5.3, when sending Packet 3, the hidden state is $(\beta_3 = -1, \alpha_3 = m_3)$. If Packet 3 is lost (with probability P_d), no new bit is transmitted, i.e. $\hat{m}_{\beta_3+3}^{\beta_4+3}$ is the empty sequence, $\emptyset$. The next state is $(\beta_4 = \beta_3 - 1, \alpha_4 = \alpha_3 \oplus k_4 \oplus w_4^s)$ (since $m_4 = k_4 \oplus w_4^{\mathbf{s}}$). When the sparsified sequence density f is small, w_4^s may be modeled as a Bernoulli random variable with parameter f. Therefore, $P(\emptyset, \alpha_3 \oplus k_4 \oplus 1, \beta_3 - 1 | \alpha_3, d_3)$ is given by fP_d, and $P(\emptyset, \alpha_3 \oplus k_4 \oplus 0, \beta_3 - 1 | \alpha_3, d_3)$ equals $(1-f)P_d$.

Referring to Figure 5.3, when sending Packet 2, the hidden state is $(\beta_2 = -1, \alpha_2 = m_2 \oplus m_1)$, due to the loss of Packet 1. Since Packet 2 is successfully transmitted, the next state becomes $(\beta_3 = -1, \alpha_3 = m_3)$ and one new bit is transmitted the sequence $\hat{m}_1$. Assume that the new bit received had a bit flip i.e. its value $\hat{m}_1 = \alpha_2 \oplus 1$, then $P(\hat{m}_1, k_3 \oplus 1, \beta_3 | \alpha_2, \beta_2)$ is given by $f * (1 - P_d) * P_s$, and $P(\hat{m}_1, k_3 \oplus 0, \beta_3 | \alpha_2, \beta_2)$ equals $(1-f) * (1 - P_d) * P_s$.

Therefore, the transition probabilities can be computed as follows

$$P(\hat{m}_{i-1+\beta_{i-1}}^{i-1+\beta_i}, \alpha_i, \beta_i | \alpha_{i-1}, \beta_{i-1}) =$$

$$\begin{cases} fP_d, & \text{if } \beta_i = \beta_{i-1} - 1 \text{ and } \alpha_i = \alpha_{i-1} \oplus k_i \oplus 1; \\ (1-f)P_d, & \text{if } \beta_i = \beta_{i-1} - 1 \text{ and } \alpha_i = \alpha_{i-1} \oplus k_i; \\ fP_s(1-P_d), & \text{if } \beta_i = \beta_{i-1}, \alpha_i = k_i \oplus 1 \text{ and } \hat{m}_{i-1+\beta_{i-1}} = \alpha_{i-1} \oplus 1; \\ (1-f)P_s(1-P_d), & \text{if } \beta_i = \beta_{i-1}, \alpha_i = k_i \text{ and } \hat{m}_{i-1+\beta_{i-1}} = \alpha_{i-1} \oplus 1; \\ f(1-P_s)(1-P_d), & \text{if } \beta_i = \beta_{i-1}, \alpha_i = k_i \oplus 1 \text{ and } \hat{m}_{i-1+\beta_{i-1}} = \alpha_{i-1}; \\ (1-f)(1-P_s)(1-P_d), & \text{if } \beta_i = \beta_{i-1}; \alpha_i = k_i; \hat{m}_{i-1+\beta_{i-1}} = \alpha_{i-1} \end{cases} \tag{5.6}$$

where $\beta_i = \beta_{i-1} - 1$ indicates the loss of a packet, and $\hat{m}_{i-1+\beta_{i-1}} = \alpha_{i-1} \oplus 1$ indicates a bit flip.

Forward-Backward Algorithm and Maximum Likelihood (ML) Decoding

Once the complete HMM model is in place, the standard forward-backward algorithm may be used to calculate the posterior probabilities $P(\hat{\mathbf{m}}|\hat{w}_j)$ for $j = 1, 2, \cdots, N$ (see Appendix B).

The *forward* probability is the joint probability of receiving a sequence $\hat{m}_1^{i-1+\beta_i}$ before transmitting the accumulated bit a_i. It is given by

$$\begin{aligned} F_i(\alpha_i, \beta_i) &= P(\hat{m}_1^{i-1+\beta_i}, \alpha_i, \beta_i) \\ &= \sum_{\substack{\alpha_{i-1}, \\ \beta_{i-1}}} F_{i-1}(\alpha_{i-1}, \beta_{i-1}) P(\hat{m}_{i-1+\beta_{i-1}}^{i-1+\beta_i}, \alpha_i, \beta_i | \alpha_{i-1}, \beta_{i-1}) \end{aligned} \tag{5.7}$$

Similarly, the *backward* probability $B_i(a_i, \beta_i)$ is defined and recursively computed as

$$
\begin{aligned}
B_i(\alpha_i, \beta_i) &= P(\hat{m}_{i+\beta_i}^{end} | \alpha_i, \beta_i) \\
&= \sum_{\substack{\alpha_{i+1}, \\ \beta_{i+1}}} P(\hat{m}_{i+\beta_i}^{i+\beta_{i+1}}, \alpha_{i+1}, \beta_{i+1} | \alpha_i, \beta_i) B_{i+1}(\alpha_{i+1}, \beta_{i+1})
\end{aligned} \tag{5.8}
$$

These forward and backward quantities are used by the ML decoder to extract the watermark estimate $\hat{\mathbf{w}}$. The posterior probability of the jth watermark bit is expanded as

$$
P(\hat{\mathbf{m}} | \hat{w}_j) = \sum_{\substack{\alpha_{j_-}, \alpha_{j_+}, \\ \beta_{j_-}, \beta_{j_+}}} F_{j_-}(\alpha_{j_-}, \beta_{j_-}) F'_{j_+}(\alpha_{j_+}, \beta_{j_+}) B_{j_+}(\alpha_{j_+}, \beta_{j_+}) \tag{5.9}
$$

where $j_- = n \cdot j$ and $j_+ = n \cdot (j+1)$ and

$$
F'_k(t_k, \beta_k) = P(\hat{m}_{j_- + \beta_{j_-}}^{k+d_k-1}, \alpha_k, \beta_k | \alpha_{j_-}, \beta_{j_-}, \hat{w}_j) \tag{5.10}
$$

and it is recursively computed as

$$
\begin{aligned}
&F'_k(\alpha_k, \beta_k) = \\
&\sum_{\substack{\alpha_{k-1}, \\ \beta_{k-1}}} F'_{k-1}(\alpha_{k-1}, \beta_{k-1}) P(\hat{m}_{k-1+\beta_{k-1}}^{k+\beta_k-1}, \alpha_k, \beta_k | \alpha_{k-1}, \beta_{k-1}, \hat{w}_j)
\end{aligned} \tag{5.11}
$$

The transition probability in equation (5.11) can be similarly derived as

in equation (5.6).

$$P(\hat{m}_{i-1+\beta_{i-1}}^{i-1+\beta_i}, \alpha_i, \beta_i | \alpha_{i-1}, \beta_{i-1}, \hat{w}_j) =$$

$$\begin{cases} P_s(1-P_d) & \text{if } \beta_i = \beta_{i-1}; \hat{m}_{i-1+\beta_{i-1}} = \alpha_{i-1} \oplus 1; \alpha_i = w_i \oplus k_i \\ (1-P_s)(1-P_d) & \text{if } \beta_i = \beta_{i-1}; \hat{m}_{i-1+\beta_{i-1}} = \alpha_{i-1}; \alpha_i = w_i \oplus k_i \\ P_d & \text{if } \beta_i = \beta_{i-1} - 1; \alpha_i = w_i \oplus k_i \oplus \alpha_{i-1} \end{cases} .$$

where $\beta_i = \beta_{i-1} - 1$ shows a packet loss which leads to merging of the IPDs ($\alpha_i = w_i \oplus k_i \oplus \alpha_{i-1}$), and $\hat{m}_{i-1+\beta_{i-1}} = \alpha_{i-1} \oplus 1$ shows a substitution error.

Once the posterior probability in equation (5.10) is solved, $\hat{w}_j$ can be decided using simple maximum likelihood decoding, and the same decoding procedure repeats for all watermark bits. Finally, the distance between $\hat{\mathbf{w}}$ and the original watermark $\mathbf{w}$ is compared to a threshold to decide whether the watermark is present.

5.3 Evaluation

To test the robustness of our scheme against network jitters and packet losses, we have analyzed their effects on the true and false positive rates. The true positives indicate the ratio of test cases where our scheme correctly detected the presence of a watermark. The false positives, on the other hand, indicate the ratio of test cases where our scheme detected the watermark in legitimate non-watermarked flows. We evaluated our watermarking scheme on network traffic generated from an independent Poisson process of rate $\lambda = 3.3$ packets per second (pps), and length 2000 packets.

The following tests were studied to analyze the effects of varying the parameters set by the watermarking scheme – the watermark length and quan-

tization step-size – and the robustness of our scheme against network jitters and packet deletions.

Each of the above scenarios was tested with varying parameters that will be listed along with the test result. The parameters that are kept constant through all the simulations are

- Sparsification factor $\eta = 10$.
- Network jitters, modeled as a Laplacian with zero mean and a standard deviation σ, which in most cases is set to 10 ms [19].
- Number of packets in all the input flow is 2000.
- The threshold was decided at a 99% confidence interval.

5.3.1 Effects of Watermark Length and Quantization Step-Size

This test studies the effects of varying the parameters that our scheme has control over – the watermark length N and the quantization step-size Δ. We compared the results of 5000 marked flows to 5000 unmarked flows, with the parameters set as follows:

Parameter	Value
N	$\{10, 30, 50\}$
P_d	0.01
σ(ms)	10
Δ(ms)	$\{20, 60, 100\}$
Test Size	5000

The results of these experiments are given in Table 5.1 and 5.2. These experiments show that increasing the watermark length and quantization step-size can improve the decoder's detection rate. It was difficult to find a

pattern in the false positives, given that they were sensitive to the threshold. When the quantization step-size is the lowest ($\Delta = 20\,\text{ms}$), the true positive instead decreases with increasing watermark length, because at that watermark length, there is no difference between flows with or without the watermark, as seen in Figure 5.5(g), (d), and (a).

Table 5.1: False positive rates of tests with varying parameters for a threshold set to have less than 1% false positives.

N \ Δ (ms)	100	60	20
50	0.0076	0.0078	0.0066
30	0.0078	0.0076	0.0094
10	0.0006	0.0090	0.0078

Table 5.2: True positives rates of tests with varying parameters for a threshold set to have less than 1% false positives.

N \ Δ (ms)	100	60	20
50	1	0.9990	0.0272
30	0.9970	0.9790	0.0310
10	0.6224	0.6050	0.0310

Figure 5.5(a)– (i) gives a better analysis of the effects of varying the watermark length and step-size. The dotted histogram shows the correlation values for the false alarm cases, i.e. the flows that did not have a watermark. The solid histogram indicates the correlation values for the 5000 watermarked flows. Increasing the step-size decreases the average of the correlation for the false alarms while increasing the average correlation for the true cases, thus allowing for a clearer distinction between the two cases. Increasing the watermark length helps decrease the average correlations and variance of the false positives, although the average of the true cases remains about constant.

For the case of $N = 50$ and $\Delta = 100\,\text{ms}$ our scheme was successfully able to recover nearly all the watermarks from the 5000 marked network flows while maintaining low false positives. Therefore, for all the tests for robustness against network jitters and packet losses, we will use $N = 50$ and $\Delta = 100\,\text{ms}$ as the parameters of our scheme.

5.3.2 Robustness against Network Jitters

The results in Table 5.3 show the effects of the network jitters on the detection of the watermark by varying the standard deviation of the jitters and setting constant all other parameters as follows:

Parameter	Value
$\sigma_N(ms)$	$\{10, 20, 30, 40\}$
P_d	0.01
N	50
Δ ms	100
Test Size	1000

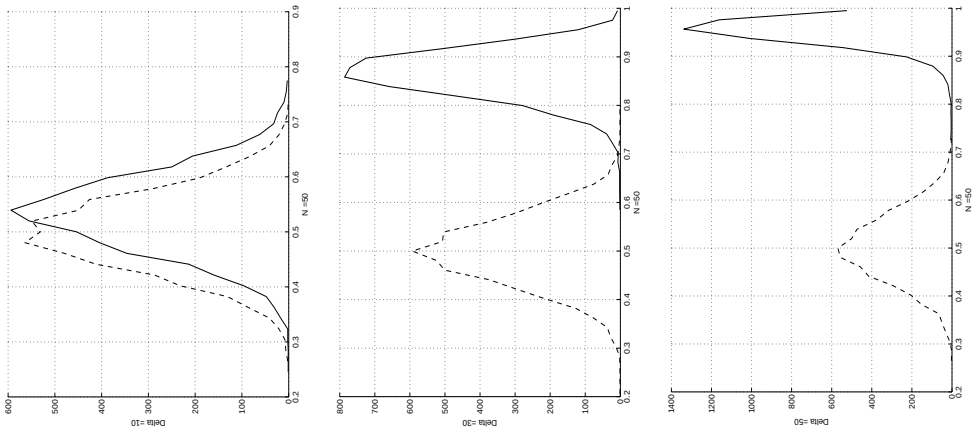
The results show that our watermarking scheme is robust up to a jitter deviation of 20 ms by achieving detection rates of over 98%. For higher jitter standard deviation, the detection reduces as expected.

Table 5.3: True positive (TP) watermark detection rates for a Laplacian modeled network jitter with varying standard deviation (S.D.).

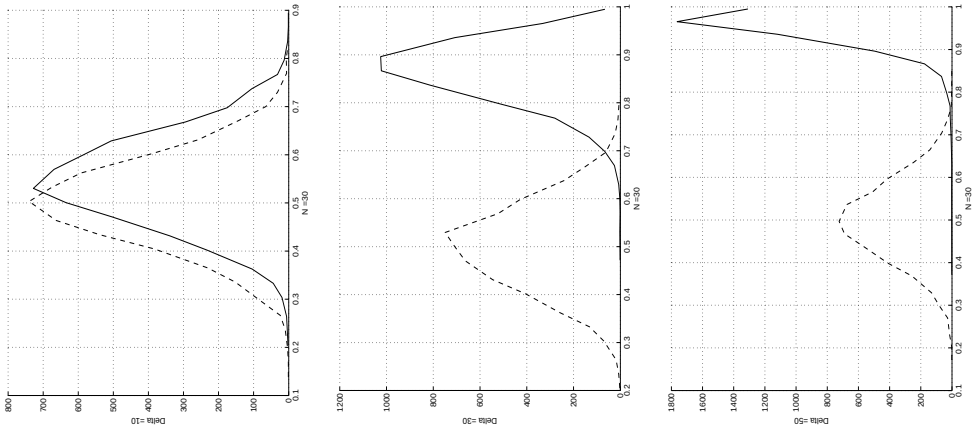
Standard Deviation (ms)	10	20	30	40
TP	1.000	0.989	0.770	0.232

5.3.3 Robustness against Packet Losses

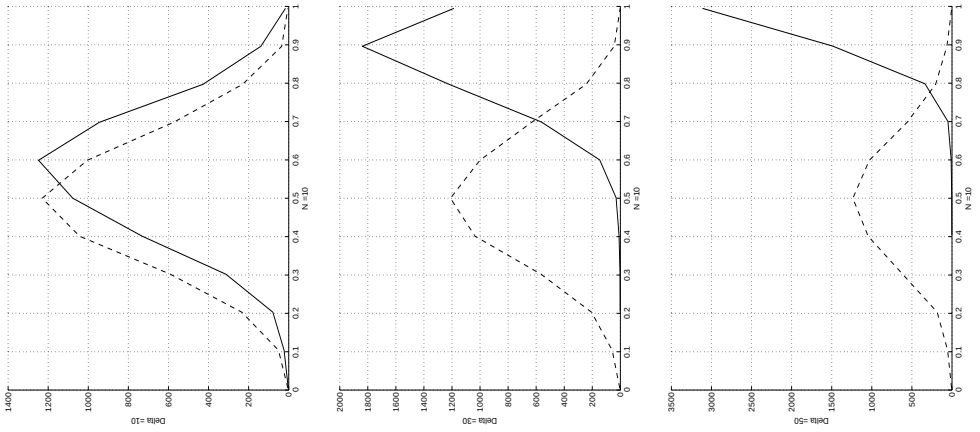
One of the novelties of this scheme is that it also considers the possibility of packet drops. To understand the effects of deletions on our detection



(a) $N = 50, \Delta = 20\,\text{ms}$ (b) $N = 50, \Delta = 60\,\text{ms}$ (c) $N = 50, \Delta = 100\,\text{ms}$



(d) $N = 30, \Delta = 20\,\text{ms}$ (e) $N = 30, \Delta = 60\,\text{ms}$ (f) $N = 30, \Delta = 100\,\text{ms}$



(g) $N = 10, \Delta = 20\,\text{ms}$ (h) $N = 10, \Delta = 60\,\text{ms}$ (i) $N = 10, \Delta = 100\,\text{ms}$

Figure 5.5: A histogram of the correlations between the extracted watermark $\hat{\mathbf{w}}$ and $\mathbf{w}$ of 5000 marked flows (solid curve) and 5000 unmarked flows (dotted curve).

probabilities, we evaluated our scheme when there were packet drops with probability of deletion $P_d = \{0.01, 0.02, 0.03, 0.1\}$ (Table 5.4) with the other parameters remaining constant and set as follows:

Parameter	Value
N	50
$\sigma_N(\,ms)$	10
Δ ms	100
Test Size	1000

The results show the effectiveness our scheme even against packet drops. In Table 5.4, we see that the detector achieves rather high true positive rates, even when up to 10% of packets were deleted while maintaining false positives under 1%. Further tests show that the true positive rate would drop to 59% when the packet deletion ratio is at 20%, which is a rare occurrence in a network system. Hence, unlike other IPD-based designs which suffer from desynchronization, our scheme is robust against both network jitters and packet losses.

Table 5.4: True positive (TP) watermark detection rates for various deletion ratios P_d. All false positive (FP) rates are restricted under 1%.

P_d	1%	2%	3%	10%
TP	1.000	1.000	1.000	0.995

5.4 Watermark Visibility

It is important that when a watermark is placed, the resultant flow should be as similar to the original flow as possible. If the watermarked flow differs from the original, then an attacker will know that the flows are being watermarked and attempt to remove the watermark or discard the flows.

In this section, to examine the visibility of our scheme; we perform two experiments using well-known techniques capable of detecting the presence of a watermark: the Kolmogorov-Smirnov test and multi-flow attack. We test the visibility using simulated flows with rates $\lambda = 3.3$ pps.

5.4.1 Kolmogorov-Smirnov Test

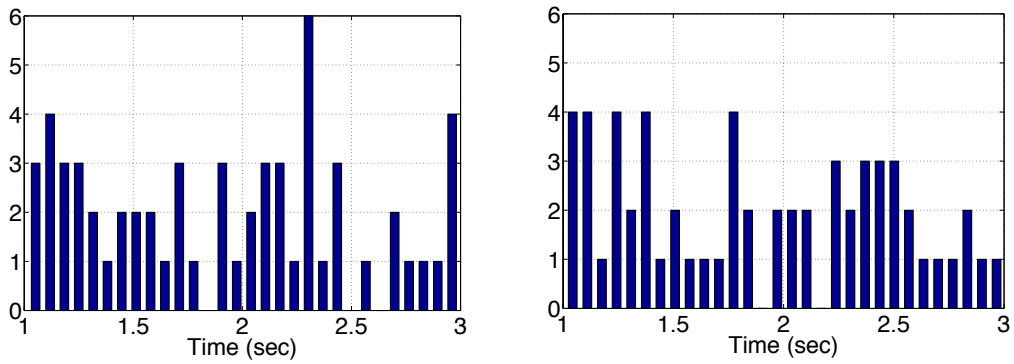
The Kolmogorov-Smirnov test was explained in Section 3.2. We performed the K-S test on simulated networks flows, comparing 1000 watermarked flows against 1000 unwatermarked flows. The results in Table 5.5 give the d_K statistic for varying watermark length $N = \{30, 40, 50\}$ and $\Delta = \{60, 80, 100\}$ ms. These results show that our watermark stays invisible within 99% confidence intervals corresponding to K-S distances below 0.036, a reference threshold suggested in [38].

Table 5.5: Average K-S distances for varying watermark lengths and step-sizes.

N \ Δ (ms)	100	80	60
30	0.0177	0.0138	0.0101
40	0.0233	0.0181	0.0133
50	0.0284	0.0223	0.0160

5.4.2 Multi-Flow Attack

We tested the visibility of our scheme against the multi-flow attack to check if our quantization step-size causes visible empty intervals. The multi-flow attack was shown to be weak against watermarks located at multiple positions [3]. For this test though, we used the same marker **m** for all the 10



(a) Unwatermarked flows

(b) Watermarked flows

Figure 5.6: Histogram of empty intervals in an aggregate of 10 flows

aggregated flows, and the marker was located in the same position (a disadvantageous setup for our scheme). The embedding parameters used were $N = 50$, $n = 10$ and $\Delta = 100\,\text{ms}$.

Testing against simulated traffic at the rate of $\lambda = 3.3$ pps, the histogram of empty intervals in the aggregated flow is depicted in Figure 5.6. Compared to the unmarked flows, no clear abnormal number of empty interval patterns are observed in the marked flows. The exact statistics of empty intervals for the two cases are given in Table 5.6, which gives the average of the number of empty intervals in the 500 packets that contain the watermark for 10 sets of aggregated flows.

Our scheme remains invisible to the multi-flow attack since, in spite of modifying the IPDs, these modifications are flow dependent, i.e. the modified IPD is always dependent on the existing IPDs. Therefore, different flows will use different IPDs to embed the same watermark.

Table 5.6: Average number of empty intervals over the first 500 packets for watermarked and unwatermarked flows.

	Marked	Unmarked
Mean	24.07	25.96
Standard Deviation	5.246	5.187

5.5 Conclusion

An invisible flow watermarking scheme is presented for network forensic application. This scheme is identified as:

- Robust to network jitters
- Robust to network packet drops
- Undetectable to schemes such as the K-S test and multi-flow attack.

Experimental results verified the first two points, that the embedded watermark can be retrieved with high probability in presence of both network jitters and high rate of packet drop. Moreover, we verified the transparency of the scheme against the K-S test and the multi-flow attack.

This scheme can later be extended to include the possibility of insertions that are also present in the network. We will also, in addition to simulated traffic, use real time SSH traffic to show that our watermarking scheme is effective at tracing network flows.

CHAPTER 6

CONCLUSION

With the rapid growth of technology, more data is being stored on the network and more systems are connecting to it. There is an increasing awareness about the importance of data security and integrity. Covert communication channels enable the transmission of data without anyone's knowledge except the intended receiver. The timing channel has been studied as a means to covertly encrypt data within the IPDs of the network flow by modifying the timings of successive packets.

In this thesis, we proposed two novel designs that, by the means of the timing channel, embed some information within the network flows; we were able to achieve the following:

- Relay spied data using a spyware which when triggered, injects the data within the IPDs of the network flow by using a coding scheme that is robust to the noise introduced by the CSMA/CA protocol. This coding scheme is based on the low-density parity-check codes coupled with shaping and dithering, followed by a decoding strategy on the graphical structure formed by the queuing channel and the encoding scheme. We demonstrated the data is decodable in spite of the back-off rule of the CSMA/CA protocol. We have verified the transparency of the spyware circuitry and the injected data against covert channel methodologies.

- Link network flows belonging to the same chain by injecting a watermark within the IPDs of the network flow. The watermark can withstand both deletion and substitution errors caused by the network jitters and packet drops by using the maximum likelihood and forward-backward algorithm on the hidden Markov model of the channel with dependent deletion and substitution errors. We demonstrated that in spite of the jitters and packet drops, we were able to reliably detect the watermark, which remained undetectable to well-known detection techniques that attempt to identify and remove the watermarked flows.

APPENDIX A

INFORMATION THEORY CONCEPTS

This chapter presents some of the information theory concepts and formulas needed for this thesis as given by [65].

Entropy is a logarithmic function of the probability of a random variable. It denotes the average uncertainty and the average number of bits required to encode a random variable

The entropy of a random variable X with probability mass function $p(x)$ is given by (using base 2)

$$H(X) = -\sum_{x} p(x) \log_2 p(x) \tag{A.1}$$

The entropy is always positive or zero, and is maximized by a uniform distribution.

For a pair of random variables (X, Y) with joint probability distribution $p(x, y)$, the *joint entropy* $H(X, Y)$ is given by

$$H(X, Y) = -\sum_{x} \sum_{y} p(x, y) \log_2 p(x, y) \tag{A.2}$$

by the chain rule we can get the *conditional entropy*,

$$H(X|Y) = H(X, Y) - H(X) \tag{A.3}$$

The *entropy rate* of a stochastic process X_i is defined by

$$H(\mathcal{X}) = \lim_{n\to\infty} \frac{1}{n} H(X_1, X_2, X_3, \ldots, X_n) \tag{A.4}$$

Mutual Information is the amount of information one random variable contains about the other. For random variables X and Y with probability mass funciton $p(x)$ and $p(y)$, respectively, the mutual information is given by

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X) \\ &= H(X) + H(Y) - H(X,Y) \end{aligned} \tag{A.5}$$

Capacity of a channel is the maximum rate at which information can be sent over the channel for it to be decoded with near zero probability of error. It is the maximum over the mutual information between the input and the output.

For a channel with input X and output Y, the capacity is given by:

$$C = \max_{p(x)} I(X;Y) \tag{A.6}$$

APPENDIX B

HIDDEN MARKOV MODEL AND FORWARD-BACKWARD ALGORITHM

A hidden Markov model (HMM) is a Markov model where the output of each state is visible, but the underlining state itself is not. For example, tossing a fair and unfair coin, where the we do not know which coin is being tossed, but we see the series of 'heads' and 'tails'.

Consider an HMM with states s_i where $i \in [1, M]$ with an output at each state i being o_i. By the properties of a Markov process,

$$P(s_{i+1}|s_1, \ldots, s_i) = P(s_{i+1}|s_i)$$

Two important definitions used on the HMM are the *forward* probability ($F_i(k)$) and the *backward* probability ($B_i(k)$) which are given by

$$\begin{aligned} F_i(k) &= P(o_1, \ldots, o_i|s_i = k) \\ &= \sum_{j=1}^{M} F_{i-1}(j) P(o_i, s_i = k|s_{i-1} = j) \\ B_i(k) &= P(o_{i+1}, \ldots, o_M|s_i = k) \\ &= \sum_{j=1}^{M} B_{i+1}(j) P(o_{i+1}, s_{i+1} = j|s_i = k) \end{aligned}$$

Using the above definitions, we can calculate the following

$$P(s_i = k, s_{i+1} = j|o_1, \ldots, o_M) = F_i(k) P(o_{i+1}, s_{i+1} = j|s_i = k) B_{i+1}(j)$$

APPENDIX C

DERIVATIONS

C.1 Shaping

Shaping or the inverse transformation technique allows us to generate a random variable with the desired distribution from a uniformly distributed random variable by taking the required distribution inverse cumulative distribution on the uniform random variable. If two random variables have the same cumulative distribution, then their probability distributions are also the same.

Claim 1. *A uniformly distributed random variable U on $[0, 1]$ can be mapped to a random variable X with the desired distribution having an invertible non-decreasing cumulative distribution $F_X(x)$ by*

$$X = F_X^{-1}(U) \tag{C.1}$$

Proof.

$$\begin{aligned} P(X <= x) &= P(F_X^{-1}(U) <= x) \quad \text{(given in equation (C.1))} \\ &= P(U <= F_X(x)) \quad (F \text{ is an increasing function}) \\ &= F_X(x) \end{aligned}$$

□

C.2 Little's Law

Little's law states that the average number of customers in a queuing system is equal to the average arrival rate of customers to that system, times the mean time of customers in that system [66].

Claim 2. *According to Little's law, the average waiting time (W) for a queuing channel with rate μ and average arrival rate λ where $\lambda < \mu$ is given by*

$$E[W] = \frac{1}{(\mu - \lambda)} \tag{C.2}$$

From Little's law we can say that:

$$E[N] = \lambda E[W] \tag{C.3}$$

where $E[N]$, λ and $E[W]$ represent the average number of customers, average arrival rate and average time spent (waiting) in the system, respectively. Let N be the number of customers in the server, and set π_i be the probability of i customers. Then for $\lambda < \mu$, the steady state solution is

$$\begin{aligned} \lambda \pi_{i-1} &= \mu \pi_i \\ \text{where } \pi_i &\triangleq P(N = i) \end{aligned} \tag{C.4}$$

Let us define

$$\rho = \frac{\lambda}{\mu}$$

Then we can rewrite equation (C.4) as

$$\pi_i = \rho\pi_{i-1}$$

$$\pi_i = \rho^i\pi_0$$

$$\pi_i = \rho^i(1-\rho)$$

Therefore,

$$\begin{aligned} E[N] &= \sum_{i=0}^{i=\infty} i\pi_i \\ &= \sum_{i=0}^{i=\infty} i(1-\rho)\rho^i \\ &= (1-\rho)\sum_{i=0}^{i=\infty} i\rho^i \\ &= (1-\rho)\frac{\rho}{(1-\rho)^2} \\ &= \frac{\rho}{(1-\rho)} \end{aligned}$$

Substituting the above in equation (C.3) we have

$$\begin{aligned} E[N] &= \lambda E[W] \\ E[W] &= \frac{1}{\lambda}E[N] \\ &= \frac{1}{\lambda}\frac{\rho}{(1-\rho)} \\ &= \frac{1}{\lambda}\frac{\frac{\lambda}{\mu}}{(1-\frac{\lambda}{\mu})} \\ E[W] &= \frac{1}{(\mu-\lambda)} \end{aligned}$$

REFERENCES

[1] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, pp. 613–615, October 1973. [Online]. Available: http://doi.acm.org/10.1145/362375.362389

[2] R. A. Kemmerer, "Shared resource matrix methodology: An approach to identifying storage and timing channels," *ACM Trans. Comput. Syst.*, vol. 1, pp. 256–277, August 1983. [Online]. Available: http://doi.acm.org/10.1145/357369.357374

[3] N. Kiyavash, A. Houmansadr, and N. Borisov, "Multi-flow attacks against network flow watermarking schemes," in *USENIX Security Symposium*, 2008, pp. 307–320.

[4] "Trusted computer system evaluation," U.S. Department of Defense, Tech. Rep. DOD 5200.28-STD, 1985.

[5] A. Wagner and V. Anantharam, "Information theory of covert timing channels," in *NATO/ASI Workshop on Network Security and Intrusion Detection*, 2005.

[6] M. H. Kang, I. S. Moskowitz, D. C. Lee, M. Ieee, and M. Ieee, "A network pump," *IEEE Transactions on Software Engineering*, vol. 22, pp. 329–338, 1996.

[7] V. Anantharam and S. Verdu, "Reflections on the 1998 information theory society paper award: Bits through queues," *IEEE Information Theory Society Newsletter*, pp. 100–102, December 1999.

[8] S. Servetto and M. Vetterli, "Communication using phantom covert channels in the internet," in *IEEE International Symposium on Information Theory*, 2004, p. 229.

[9] X. Luo, E. W. Chan, and R. K. C. Chang, "Covert communications in TCP burstiness," Hong Kong Polytechnic University, preprint, 2007.

[10] V. Anantharam and S. Verdu, "Bits through queues," *IEEE Transactions on Information Theory*, vol. 42, no. 1, pp. 4 –18, January 1996.

[11] A. Bedekar and M. Azizoglu, "The information-theoretic capacity of discrete-time queues," *IEEE Transactions on Information Theory*, vol. 44, no. 2, pp. 446 –461, mar 1998.

[12] R. Sundaresan and S. Verdu, "Sequential decoding for the exponential server timing channel," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 705 –709, March 2000.

[13] M. A. Padlipsky, D. W. Snow, and P. A. Karger, "Limitations of end-to-end encryption in secure computer networks," Hanscom Air Force Base, MA, August 1978.

[14] S. Cabuk, C. Brodley, and C. Shields, "IP covert timing channels: Design and detection," in *ACM conference on Computer and Communications Security (CCS)*, 2004, pp. 178–187.

[15] S. Staniford-Chen and L. T. Heberlein, "Holding intruders accountable on the internet," in *IEEE Symposium on Security and Privacy*, pp. 39–49.

[16] Y. Zhang and V. Paxson, "Detecting stepping stones," in *USENIX Security Symposium*, 2000, pp. 171–184.

[17] X. Wang and D. S. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," in *ACM Conference on Computer and Communications Security*, 2003, pp. 20–29.

[18] Y. J. Pyun, Y. H. Park, X. Wang, D. S. Reeves, and P. Ning, "Tracing traffic through intermediate hosts that repacketize flows," in *Infocom*, 2007, pp. 634–642.

[19] A. Houmansadr, N. Kiyavash, and N. Borisov, "Rainbow: A robust and invisible non-blind watermark for network flows," in *Network and Distributed System Security Symposium*, 2009.

[20] K. Yoda and H. Etoh, "Finding a connection chain for tracing intruders," in *Proceedings of the 6th European Symposium on Research in Computer Security*. London, UK: Springer-Verlag, 2000, pp. 191–205.

[21] X. Wang, D. S. Reeves, and S. F. Wu, "Inter-packet delay based correlation for tracing encrypted connections through stepping stones," in *Proceedings of the 7th European Symposium on Research in Computer Security*, ser. ESORICS '02. London, UK, UK: Springer-Verlag, 2002, pp. 244–263.

[22] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer VoIP calls on the internet," in *Proceedings of the 12th ACM conference on Computer and communications security*, ser. CCS '05. New York, NY: ACM, 2005. [Online]. Available: http://doi.acm.org/10.1145/1102120.1102133 pp. 81–91.

[23] X. Wang, S. Chen, and S. Jajodia, "Network flow watermarking attack on low-latency anonymous communication systems," in *IEEE Symposium on Security and Privacy*, 2007.

[24] M. Kang and I. Moskowitz, "A pump for rapid, reliable, secure communication," in *ACM conference on Computer and communications security (CCS)*, 1993, pp. 119–129.

[25] J. Giles and B. Hajek, "An information-theoretic and game-theoretic study of timing channels," *IEEE Transactions on Information Theory*, vol. 48, no. 9, pp. 2455–2477, 2002.

[26] M. Kang, I. Moskowitz, and D. Lee, "A network version of the pump," in *Security and Privacy, 1995. Proceedings., 1995 IEEE Symposium on*, may 1995, pp. 144–154.

[27] V. Berk, A. Giani, and G. Cybenko, "Detection of covert channel encoding in network packet delays," Dartmouth College, Dept of Computer Science Tech. Rep. TR2005536, 2005.

[28] S. Gianvecchio and H. Wang, "Detecting covert timing channels: an entropy-based approach," in *Proceedings of the 14th ACM conference on Computer and communications security*, ser. CCS '07. New York, NY: ACM, 2007. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315284 pp. 307–316.

[29] P. Peng, P. Ning, and D. Reeves, "On the secrecy of timing-based active watermarking trace-back techniques," in *Security and Privacy, 2006 IEEE Symposium on*, may 2006, pp. 15 pp. –349.

[30] A. Kolmogorov, "Sulla determinazione empirica di una legge di distribuzione," *Giornale dell' Istituto Italiano degli Attuari*, vol. 4, pp. 83–91, 1933.

[31] N. Smirnov, "Sur les écarts de la courbe de distribution empirique," *Recueil Mathématique (Matematiceskii Sbornik), N.S.*, vol. 6, pp. 3–26, 1939.

[32] "Defense Science Board (DSB) study on high performance microchip supply," 2005, http://www.acq.osd.mil/dsb/reports/2005-02-HPMS_Report_Final.pdf.

[33] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using IC fingerprinting," in *IEEE Symposium on Security and Privacy*, 2007, pp. 296–310.

[34] Y. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *USENIX Security*, 2007, pp. 291–306.

[35] S. Adee, "The hunt for the kill switch," May 2008, http://www.spectrum.ieee.org/may08/6171.

[36] S. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou., "Designing and implementing malicious hardware," in *USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[37] B. Prabhakar and R. Gallager, "Entropy and the timing capacity of discrete queues," *IEEE Transactions on Information Theory*, vol. 49, no. 2, pp. 357 –370, February 2003.

[38] F. J. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.

[39] S. Cabuk, "Network covert channels: Design, analysis, detection, and elimination," Tech. Rep., 2006.

[40] G. Shah, A. Molina, and M. Blaze, "Keyboards and covert channels," in *Proceedings of the 15th conference on USENIX Security Symposium - Volume 15*, ser. USENIX-SS'06. Berkeley, CA: USENIX Association, 2006. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267336.1267341

[41] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "DSSS-based flow marking technique for invisible traceback," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*, ser. SP '07. Washington, DC: IEEE Computer Society, 2007. [Online]. Available: http://dx.doi.org/10.1109/SP.2007.14 pp. 18–32.

[42] R. G. Gallager, "Low density parity check codes," Ph.D. dissertation, Massachusetts Institute of Technology, 1960.

[43] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions Information Theory*, vol. 47, pp. 619–637, 2001.

[44] R. Gallager, "Low-density parity-check codes," *Information Theory, IRE Transactions on*, vol. 8, no. 1, pp. 21 –28, January 1962.

[45] D. J. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, 1996.

[46] D. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Transactions on Information Theory*, vol. 45, no. 2, pp. 399 –431, March 1999.

[47] F. Kschischang, B. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498 –519, February 2001.

[48] G. Forney, "Codes on graphs: Normal realizations," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 520–548, 2001.

[49] T. Coleman and N. Kiyavash, "Sparse graph codes and practical decoding algorithms for communicating over packet timings in networks," *Conference on Information Sciences and Systems (2008)*, pp. 447–452, 2008.

[50] R. G. Gallager, *Information Theory and Reliable Communication*. New York, NY: John Wiley & Sons, Inc., 1968.

[51] A. Bennatan and D. Burshtein, "Design and analysis of nonbinary LDPC codes for arbitrary discrete-memoryless channels," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 549–583, 2006.

[52] J. Ziv, "On universal quantization," *IEEE Transactions on Information Theory*, vol. 31, no. 3, pp. 344–347, 1985.

[53] B. Chen and G. W. Wornell, "Quantization index modulation: A class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, vol. 47, pp. 1423–1443, 2001.

[54] U. Erez and R. Zamir, "Achieving 1/2 log (1+SNR) on the AWGN channel with lattice encoding and decoding," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2293 – 2314, oct. 2004.

[55] G. D. F. Jr., "On the role of MMSE estimation in approaching the information-theoretic limits of linear Gaussian channels: Shannon meets Wiener," *CoRR*, vol. arXiv:cs.IT/0409053, 2004.

[56] N. Kiyavash, T. Coleman, and M. Rodrigues, "Novel shaping and complexity-reduction techniques for approaching capacity over queuing timing channels," in *Communications, 2009. ICC '09. IEEE International Conference on*, June 2009, pp. 1 –5.

[57] "IEEE 802.11 LAN/MAN Wireless LANS Standard," 2007, http://standards.ieee.org/getieee802/802.11.html.

[58] "WARP: Wireless Open-Access Research Platform," 2008, http://warp.rice.edu/trac/.

[59] R. Sundaresan and S. Verdu, "Robust decoding for timing channels," *IEEE Trans. on Information Theory*, vol. 46, no. 2, pp. 405–419, 2000.

[60] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York, NY, USA: Cambridge University Press, 2008.

[61] A. Tartakovsky and V. Veeravalli, "Asymptotic analysis of Bayesian quickest change detection procedures," *IEEE International Symposium on Information Theory*, 2002.

[62] A. Tartakovsky, B. Rozovskii, R. Blažek, and H. Kim, "Detection of intrusions in information systems by sequential change-point methods," *Statistical Methodology*, vol. 3, no. 3, pp. 252–293, 2006.

[63] W. Ding and S.-H. Huang, "Detecting intruders using a long connection chain to connect to a host," in *Advanced Information Networking and Applications (AINA), 2011 IEEE International Conference on*, march 2011, pp. 121 –128.

[64] M. C. Davey and D. J. C. Mackay, "Reliable communication over channels with insertions, deletions, and substitutions," *IEEE Transactions on Information Theory*, vol. 47, pp. 687 – 698, 2001.

[65] T. M. Cover and J. Thomas, *Elements of Information Theory*, 2nd ed. New York: Wiley, 1991.

[66] L. Kleinrock, *Theory, Volume 1, Queueing Systems*. Wiley-Interscience, NY, 1975.