

3-11-2011

Covert Channels Within IRC

Wayne C. Henry

Follow this and additional works at: <https://scholar.afit.edu/etd>

Part of the [Computer and Systems Architecture Commons](#), and the [Digital Communications and Networking Commons](#)

Recommended Citation

Henry, Wayne C., "Covert Channels Within IRC" (2011). *Theses and Dissertations*. 1394.
<https://scholar.afit.edu/etd/1394>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.

88th ABW PUBLIC AFFAIRS SECURITY AND POLICY REVIEW WORKSHEET

NOTE: Public release clearance is NOT required for material presented in a closed meeting and which will not be made available to the general public, on the Internet, in print or electronic media.

Internal Case Number

AFIT/ENG/11-04

1. SUBMITTER NAME janice.jones@afit.edu	OFFICE SYMBOL AFIT/ENG	PHONE 53636	2. DATE SUBMITTED
3. AUTHOR(S) NAME Wayne C. Henry	ORGANIZATION		PHONE 53636
4. DOCUMENT TITLE Covert Channels Within IRC	6. CONFERENCE/EVENT/PUBLICATION NAME N/A		7. DATE NEEDED N/A
5. DOCUMENT TYPE <input type="checkbox"/> ABSTRACT <input type="checkbox"/> TECH REPORT <input type="checkbox"/> JOURNAL ARTICLE <input type="checkbox"/> VIDEO <input type="checkbox"/> SPEECH <input type="checkbox"/> TECH PAPER <input type="checkbox"/> BRIEFING CHARTS <input type="checkbox"/> PHOTO <input checked="" type="checkbox"/> THESIS/DISSERTATION <input type="checkbox"/> OTHER _____	8. EVENT LOCATION N/A		9. EVENT DATE N/A

10. RELATED CASES PREVIOUSLY APPROVED	11. OTHER AGENCY COORDINATION REQUIRED <i>(List contact information)</i>	12. OTHER INFORMATION
---------------------------------------	---	-----------------------


13. NATIONAL SECURITY STATUTES/TECHNOLOGY ISSUES a. <input type="checkbox"/> YES <input checked="" type="checkbox"/> NO Are any aspects of this technology included in: U.S. Munitions List; ITAR 22, CFR Part 121; CCL; Security Classification Guide; DD Form 254 or a Technology Protection Plan? <i>(If YES, please explain in Block 16)</i> b. <input checked="" type="checkbox"/> YES <input type="checkbox"/> NO Does this information meet the criteria for Distribution Statement "A" - unclassified, unlimited distribution?	14. NATIONAL SECURITY STATUTES/TECHNOLOGY ISSUES a. <input type="checkbox"/> YES <input checked="" type="checkbox"/> NO If this material results from an international agreement is the USAF authorized to release program information? <i>(If NO, please identify release authority organization)</i> b. <input type="checkbox"/> YES <input checked="" type="checkbox"/> NO If this is a joint program, does your organization maintain primary management responsibility and authority to release all information? <i>(If NO, please provide name of lead organization/POC (i.e., DARPA, NAS, ARMY, etc.))</i> c. <input type="checkbox"/> YES <input checked="" type="checkbox"/> NO If this information is for a SBIR contractor and the program manager is responsible for public release, is a waiver letter on the file granting permission to release?
--	--

15. BUDGET CATEGORIES <i>(Funding is under Budget Category-Program Element)</i> <input type="checkbox"/> 6.1. <input type="checkbox"/> 6.3. <input checked="" type="checkbox"/> N/A <input type="checkbox"/> 6.2. <input type="checkbox"/> 6.4./HIGHER <input type="checkbox"/> OTHER _____

16. EXPLANATION

17. ORIGINATOR I certify the attached material is unclassified, technically accurate, contains no critical military technology, is not subject to export controls and is suitable for public release.	PRINT NAME Wayne Henry SIGNATURE HENRY.WAYNE.CHRISTOPHE.1177463	DATE (YYYYMMDD) 20110224
--	--	-----------------------------

18. TECHNICAL REVIEW AND CERTIFICATION I certify the information contained in the attached document is technically accurate; does not disclose classified, sensitive, or military critical technology, does not violate proprietary rights, copyright restrictions; and is not subject to export control regulations. I further certify that this information is suitable for public release.	PRINT NAME Barry E. Mullins SIGNATURE MULLINS.BARRY.E.1102753409	DATE (YYYYMMDD) 20110224
--	---	-----------------------------

19. SECURITY MANAGER REVIEW Signature certifies that the information has been reviewed and the information contains no Operational Security issues.	PRINT NAME Nathaniel J Davis IV, PhD SIGNATURE 	DATE (YYYYMMDD) 2011 Feb 25
--	--	--------------------------------

20. ADDITIONAL REVIEW I certify that this information is suitable for public release.	PRINT NAME SIGNATURE CLICK HERE TO SIGN	DATE (YYYYMMDD)
--	---	-----------------

21. PA USE ONLY			
<input type="checkbox"/> APPROVED <input type="checkbox"/> AS AMENDED <input type="checkbox"/> DISAPPROVED	DATE	PAO SIGNATURE	CASE NUMBER



COVERT CHANNELS WITHIN IRC

THESIS

Wayne C. Henry, Captain, USAF

AFIT/GCE/ENG/11-04

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, Department of Defense, or the United States Government. This material is declared a work of the U. S. Government and is not subject to copyright protection in the United States.

AFIT/GCE/ENG/11-04

COVERT CHANNELS WITHIN IRC

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Computer Engineering

Wayne C. Henry, BSCE

Captain, USAF

March 2011

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

COVERT CHANNELS WITHIN IRC

Wayne C. Henry, BSCE
Captain, USAF

Approved:

Dr. Barry E. Mullins (Chairman)

Date

Dr. Robert F. Mills (Member)

Date

Dr. Rusty O. Baldwin (Member)

Date

Abstract

The exploration of advanced information hiding techniques is important to understand and defend against illicit data extractions over networks. Many techniques have been developed to covertly transmit data over networks, each differing in their capabilities, methods, and levels of complexity.

This research introduces a new class of information hiding techniques for use over Internet Relay Chat (IRC), called the Variable Advanced Network IRC Stealth Handler (VANISH) system. Three methods for concealing information are developed under this framework to suit the needs of an attacker. These methods are referred to as the Throughput, Stealth, and Baseline scenarios. Each is designed for a specific purpose: to maximize channel capacity, minimize shape-based detectability, or provide a baseline for comparison using established techniques applied to IRC.

The effectiveness of these scenarios is empirically tested using public IRC servers in Chicago, Illinois and Amsterdam, Netherlands. The Throughput method exfiltrates covert data at nearly 800 bits per second (bps) compared to 18 bps with the Baseline method and 0.13 bps for the Stealth method. The Stealth method uses Reed-Solomon forward error correction to reduce bit errors from 3.1% to nearly 0% with minimal additional overhead. The Stealth method also successfully evades shape-based detection tests but is vulnerable to regularity-based tests.

Acknowledgements

I would like to thank my thesis adviser, Dr. Barry Mullins, for his guidance and support through this thesis, allowing me freedom to pursue my ideas while steering me on track. I would also like to thank Dr. Rusty Baldwin and Dr. Robert Mills for their support and assistance.

I also thank my fellow classmates, whose listening ears and helpful support helped keep my sanity in check.

Finally, I would be remiss if I did not thank my loving wife and children, who gave me daily encouragement, understanding, and support throughout this process. This effort could not have been possible without them.

Wayne C. Henry

Table of Contents

Abstract	iv
Acknowledgements	v
Table of Contents	vi
List of Figures	ix
List of Tables	xi
I. Introduction	1
1.1 Motivation	1
1.2 Internet Relay Chat for Covert Communications	2
1.3 Steganography and Covert Channels	3
1.4 Research Goals	4
1.5 Assumptions and Limitations	5
1.6 Thesis Overview	6
II. Literature Review and Related Research	7
2.1 Internet Relay Chat	7
2.2 Steganography	11
2.2.1 Terminology	13
2.2.2 Encrypted Steganographic Systems	15
2.2.3 Text-Based Steganography	15
2.2.4 Whitespace Steganography	18
2.3 Covert Timing Channels	20
2.3.1 Examples of Covert Timing Channels	21
2.3.1.1 IP Covert Timing Channel	22
2.3.1.2 Model Based Covert Timing Channel	22
2.3.1.3 Jitterbug Covert Timing Channel	23

2.3.1.4	Liquid Covert Timing Channel	24
2.3.2	Defense Against Covert Channels	24
2.3.2.1	Kolmogorov-Smirnov Test	25
2.3.2.2	Regularity Test	26
2.4	RC4 Encryption	27
2.5	Summary	29
III.	Methodology	30
3.1	Problem Definition.....	30
3.1.1	Goals and Hypothesis	31
3.1.2	Approach.....	32
3.1.2.1	Experiment #1: Determine IRC Hidden Characters.....	34
3.1.2.2	Experiment #2: Maximize IRC Capacity Limits.....	35
3.1.2.3	Experiment #3: Identify Legitimate IRC Traffic Distribution	35
3.2	System Boundaries.....	36
3.3	System Services	37
3.4	Workload.....	37
3.5	Performance Metrics	38
3.6	Parameters.....	39
3.6.1	Workload Parameters.....	39
3.6.2	System Parameters	40
3.7	Factors.....	42
3.8	Evaluation Technique	43
3.9	Experimental Design.....	44
3.10	Summary.....	45
IV.	Software Architecture and Design	46
4.1	Baseline Encoding Algorithm.....	46
4.1.1	Experiment #1: Determine IRC Hidden Characters	46
4.1.2	Baseline Encoding Algorithm Details	49

4.2	Throughput Encoding Algorithm.....	53
4.2.1	Experiment #2: Maximize IRC Throughput.....	53
4.2.2	Maximizing Capacity per Packet.....	54
4.2.3	Maximizing Capacity per Second.....	57
4.2.4	Results and Analysis of Server Throttling Experiments.....	59
4.3	Stealth Encoding Algorithm.....	62
4.3.1	Experiment #3: Identify Legitimate IRC Traffic Distribution.....	62
4.3.2	Stealth Encoding Algorithm Details.....	64
4.3.3	Reed-Solomon Forward Error Correction.....	67
4.4	Summary.....	68
V.	Results and Analysis.....	70
5.1	Results and Analysis of IPD Characteristics.....	70
5.2	Throughput and BER Results.....	75
5.3	Detection Results.....	79
5.4	Summary.....	87
VI.	Conclusions.....	88
6.1	Recommendations for Future Research.....	88
6.2	Research Summary.....	89
6.3	Significance of Research.....	91
	Bibliography.....	93

List of Figures

Figure	Page
1. IRC Client to Client Communication	8
2. Users On IRC During a Two Week Period in May, 2010 [Gel10].....	9
3. Data-hiding Problem Space [BGM+96]	13
4. A Keyed Steganography System	14
5. Example of character coding [BLM+99].....	18
6. Whitespace encoded text [BGM+96]	19
7. Data hidden through whitespace with justification [BGM+96].....	19
8. RC4 Encryption Algorithm.....	28
9. Framework to design VANISH sender	32
10. Framework to design VANISH receiver.....	33
11. Experiments and Tests for Research Methodology	34
12. The VANISH System	36
13. Experimental Configuration.....	44
14. ASCII Table Reference.....	47
15. Hidden Character Output, Test Characters 0x20h, 0x03h, and 0x02h	49
16. Snow Steganographic Encoding	50
17. IRC Character Test - Tab 0x09h.....	50
18. Baseline Encoding Wireshark Capture	52
19. Baseline Encoding IRC Channel Traffic	52
20. Throughput Transmitted Packet (left), Received Packet (right).....	56

Figure	Page
21. Throughput Encoded Channel Traffic	56
22. Timing Variance for the 0.5 Second Trials.....	60
23. Timing Variance for the 1.0 Second Trials.....	61
24. PDF of Observed IRC Traffic.....	62
25. Q-Q Plot of Log Observed Traffic versus Normal Distribution.....	63
26. PDF of Observed IRC Traffic versus Best Fit Curve (red)	64
27. Sample Gaussian Distribution.....	66
28. Legitimate Traffic versus Stealth Results.....	75
29. p-Value Weighting Scale [RaS02].....	80
30. Legitimate Traffic Variance.....	81
31. Stealth Probability Density (red) versus Legitimate Traffic Density (black line) - Log Transformed, WAN-US (left) and WAN-EU (right)	84
32. Stealth WAN-US Traffic Regularity	86
33. Baseline WAN-US Traffic Regularity.....	87

List of Tables

Table	Page
1. Common Commands Used by IRC Applications	8
2. Test Network Scenarios	42
3. Factor Levels.....	42
4. Hidden Character Categories	48
5. Bit Sequence 3-to-1 Whitespace Multiplexer	51
6. Throughput Encoding Scheme.....	55
7. Timing Reliability for Three Time Interval Configurations	59
8. Legitimate Traffic Statistical Summary.....	71
9. WAN-US Inter-Packet Delay Statistical Summary	71
10. WAN-EU Inter-Packet Delay Statistical Summary	72
11. Throughput and BER Results - WAN-US	76
12. Throughput and BER Results - WAN-EU.....	76
13. Legitimate Traffic Detection Results.....	81
14. Detection Results - WAN-US	82
15. Detection Results - WAN-EU.....	82

COVERT CHANNELS WITHIN IRC

I. Introduction

1.1 Motivation

The emergence and growth of the Internet has created a global society dependent on information systems. Businesses leverage information systems to market products and aid efficiency, increasing their bottom line. Governments and academia rely on the Internet for information collection and collaboration. With the growing reliance on information and information technology, maintaining the confidentiality of sensitive data stored on computers is paramount to the success and failure of many projects. To this end, the exploration of information hiding techniques, including steganography and covert channels, are important to understand and defend against illicit data exfiltration over networks.

There have been several high-profile cases involving steganography and covert channel use in the media. After the September 11, 2001 attacks on the World Trade Center and Pentagon, news spread that the Al-Qaeda agents involved in the attacks used steganography to covertly communicate while evading detection by federal agents [CaK01]. A recent case made headlines when U.S. agents examined the computers of 11 captured Russian spies and discovered several steganographic tools used to communicate sensitive information from the U.S. back to Russia [Eat10].

It is clear that adversaries use steganography and other covert communication methods to bypass detection mechanisms. Therefore, investigating new hidden channel techniques are important to understanding and defending against future attacks.

1.2 Internet Relay Chat for Covert Communications

Nearly any type of network traffic can transmit covert information including text, images, audio, video, and even unused fields of network packet headers [Cac05]. However, despite its use dating back to 1988, researchers have not focused on IRC as a means for covert communication. Yet, there are several reasons why IRC is a prime choice for hidden channels:

- IRC boasts a large and diverse group of users including business, academia, and the military [Lea09] [ZLC08] [Eov06].
- IRC messages are small in size and great in number which makes it difficult to examine each message thoroughly [Ada08].
- Botnet masters will likely begin using steganography to disguise their command and control communications over IRC in 2011 [Lew10].
- IRC servers act as a man-in-the-middle between the transmitter of the covert data and their recipient. There is no network trace of the recipient's IP on the attacked network, only on server records. This issue is further compounded by the proliferation of unregulated IRC servers that do not disclose user identities. These same servers provide a haven for botnet masters against legal repercussions.

1.3 Steganography and Covert Channels

Within the literature on information hiding, the terms “steganography” and “covert channel” are often used interchangeably. Steganography is the process through which a message is hidden within a static medium and can only be seen by a readily prepared party [KaP00]. Because there are typically large amounts of redundancy, or unused space, in files or network mediums, it is ideal for concealing large amounts of data. Covert channels on the other hand, provide a way to surreptitiously leak information from an entity in a higher-security level to an entity in a lower level [KaP00]. The difficulty of detecting or eliminating such channels makes them a desirable choice for adversaries that value stealth over throughput.

There are two types of covert channels: covert storage channels and covert timing channels. A covert storage channel manipulates the contents of a storage location (e.g., disk, memory, packet headers, etc.) to transfer information. A covert timing channel (CTC) manipulates the timing or ordering of events (e.g., disk accesses, memory accesses, packet arrivals, etc.) to transfer information.

The two primary design goals of information hiding techniques are high capacity and detection resistance. However, the pursuit of one of these goals often comes at the sacrifice of the other [BGM⁺96]. Previous research has shown that entropy-based techniques are effective at detecting the presence of CTCs through statistical analysis. Specifically, the shape [BGC05] [GWW⁺08] and regularity [CBS04] [SMB06] of network traffic with covert signals should be effective discriminating factors when compared to legitimate network samples.

1.4 *Research Goals*

This thesis develops new information hiding techniques over IRC and evaluates their performance using public IRC servers to emulate real scenarios. The system herein uses steganographic or CTC encoding methods to exfiltrate data with high capacity or high detection resistance, depending on the needs of the user.

The goals of this research are to:

- Develop a baseline steganographic system over IRC based on Snow [Kwa06], an existing steganographic tool for whitespace encoding in text documents. This serves as the baseline for comparison.
- Construct a steganographic system maximizing throughput capacity per packet (CPP) and capacity per second (CPS). Capacity is the amount of covert information in the secret file in bits. Therefore, CPP is a function of the size of the secret file and amount of covert information embedded per packet, while CPS is dependent on the size of the secret file and the rate at which the covert packets are transmitted to the IRC server.
- Build a covert timing channel system which minimizes detectability by “hiding” in legitimate IRC traffic patterns.
- Examine and analyze the encoding techniques for reliability, throughput, and detectability.

The three covert channel techniques are designed under a common modular framework called the Variable Advanced Network IRC Stealth Handler (VANISH) system. The three covert channels under test, maximizing throughput, minimizing

detectability, and the baseline, are not intended to exhaust all possible information hiding capabilities over IRC. Rather, they are meant to show a range of capabilities by exploiting the IRC protocol in different ways, while maximizing either throughput or stealth.

1.5 Assumptions and Limitations

There are several assumptions made that allow the covert communications to occur while also limiting the scope of this research effort. First, these techniques require access to the victim's computer to execute the covert channel software. There are a variety of ways to gain access to target computers including malicious emails, USB devices, websites, and insider threats; however, it is assumed the malicious code is already running on the target machine. Second, it is assumed that the computer network of the infected host does not block or filter IRC traffic. Blocking all IRC traffic is the best way to prevent these threats. However, blocking IRC traffic may decrease productivity of the organization, depending on their use of IRC. In any case, these techniques could be applied to other types of text-based network traffic with little effort. Finally, the covert techniques developed do not provide a secure method for exchanging encryption keys and, following best practice, keys are assumed to be securely exchanged out-of-band and prior to the covert transmission. While encryption is not necessary for the covert methods to operate, it adds an extra layer of security to the system in the event that the covert traffic is discovered while imposing no additional network overhead.

1.6 Thesis Overview

The remainder of this document is structured as follows. Chapter 2 covers background and related work in IRC, steganography, covert timing channels, and their detection schemes. Chapter 3 outlines the methodology used to design, setup, and conduct the experiments to test the effectiveness of the VANISH system. Chapter 4 discusses the VANISH framework, trade studies, and design decisions. Chapter 5 presents the results and validates the effectiveness of the techniques through experimentation. Finally, Chapter 6 presents conclusions and discusses directions for future work.

II. Literature Review and Related Research

This chapter presents an overview of background information and related research on Internet Relay Chat (IRC), steganography techniques, covert timing channels, and the encryption scheme used in this research. Section 2.1 provides historical background on IRC as well as its uses today. Section 2.2 discusses the background in steganography and presents the current research in the field of text-based and whitespace steganography. Section 2.3 is a literature review of the latest research in the field of covert timing channels and their detection mechanisms. The RC4 encryption scheme used in this research is presented in Section 2.4. Finally, the chapter is summarized in Section 2.5.

2.1 *Internet Relay Chat*

Internet Relay Chat (IRC) was created in August 1988 by Jarkko Oikarinen as a means for communication based on the Bulletin Board System [Oik05]. In May 1993, Request for Comments (RFC) 1459 established a formal IRC protocol using Transmission Control Protocol (TCP) for reliable end-to-end chat services and optional Transport Layer Security (TLS) for encryption [OiR93]. The default port for IRC is 6667, however many IRC networks will also accept clients on nearby port numbers such as 6661-6679. In the IRC architecture, users download an IRC client, install it, and then connect to centralized servers. Users then join a “channel” which divides users into groups based on discussion topics and allows them to communicate to the group or to an

individual via a private message. Table 1 describes the most common commands used in IRC applications.

Table 1: Common Commands Used by IRC Applications

Command	Function
/help	Get help using IRC
/server	Connect to a server
/join	Join a channel
/leave	Leave a channel
/list	List all the channels on the server
/msg	Send a message to a user or channel
/nick	Change user's nickname
/quit	Terminate the IRC session

Figure 1 shows how the IRC architecture enables communication between peers and the server. As shown in the figure, to send a message to a channel the following occurs:

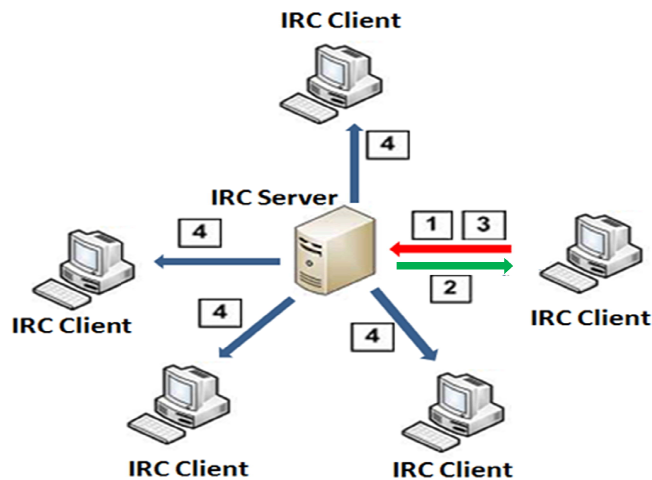


Figure 1: IRC Client to Client Communication

1. The client connects to an IRC server and selects a channel to join (red arrow).
2. Client and server exchange information establishing the user's nickname and the channel information (green arrow).
3. Client sends a message to be displayed in the channel. This communication is sent to the server in a "Request" packet (red arrow).
4. Server forwards the packet to all other clients within the same channel as a "Response" packet. This communication is not sent back to the original sender's client by the server (blue arrows).

Due to IRC's ease of use and ability to instantly communicate with a large number of people, its use has steadily increased since its introduction. In 1991, IRC was used to report on the Soviet coup d'état attempt throughout the government-mandated media blackout [IRC91]. During the Gulf War, IRC gained popularity as an instant communications medium for late-breaking news updates before news broadcast coverage [IRC94]. Figure 2 shows the number of IRC users and channels active during a two week period from the top 10 major IRC networks participating in the survey. The total number of users fluctuates between 600,000 to 830,000 people.

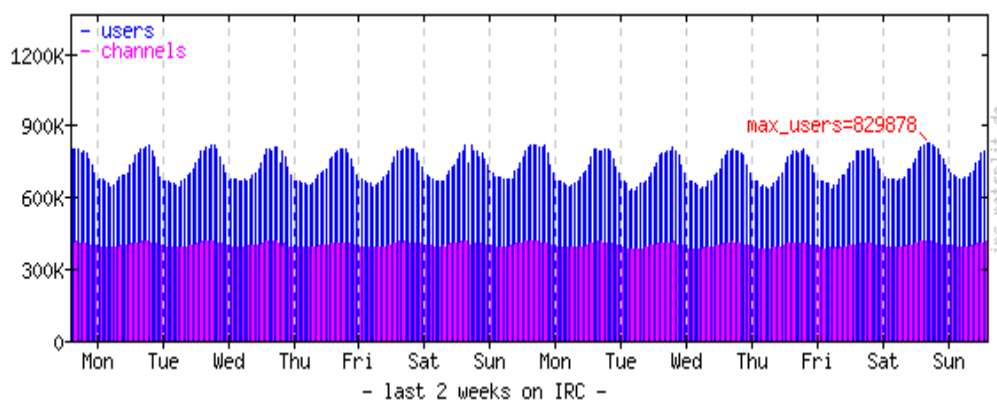


Figure 2: Users On IRC During a Two Week Period in May, 2010 [Gel10]

In the US military, applications such as e-mail and chat have proven to be warfighting enablers by enhancing mission planning with a rapid communication channel [Eov06]. Text-based chat is used extensively by all military branches and throughout the Department of Defense. It is used for unit-level tactical coordination as well as broad-scale strategic planning and joint operations. Increasingly, IRC-like applications are becoming a preferred tool for communication between disparate platforms or with coalition partners.

Eovito [Eov06] notes that the use of chat among joint forces has evolved in an ad-hoc fashion in an effort to fill gaps in existing command and control (C2) systems, but has become an essential communications tool favored over more traditional methods. IRC and similar instant chat applications have several advantages over other traditional C2 systems:

1. ***Bandwidth.*** The bandwidth requirements for text-based chat are far less than for other data systems. This is important in bandwidth-constrained tactical environments.
2. ***Speed.*** Chat is faster than other systems due to rapid transmission time of text and also due to the more rapid turnaround compared to other methods such as phone calls or radio. Chat provides for simultaneous transcription and dissemination in a one-to-many fashion.
3. ***Ease of use.*** Most chat clients have a very small learning curve compared to other C2 systems, thus requiring less training.
4. ***Availability.*** Users typically experience a higher degree of availability with chat servers compared with other C2 systems. According to [Eov06], users “reported

that chat was the only form of communication in many cases, where units were too far for voice, and the available transmission systems lacked the bandwidth for larger C2 systems.” Additionally, many Command, Control, Communications, Computer, and Intelligence (C4I) plans call for chat to be one of the first systems available when deployed, making it useful as a coordination tool for bringing other C2 systems online.

5. *Efficiency*. Tactical users often find that “chat allows them to send more data with less time and effort” [Eov06]. Also, it is easy to monitor chat while working with other onscreen tools, maps, etc. Since chat provides a running transcript, users spend less time having to repeat information that was previously disseminated. Additionally, because users may participate in multiple chat rooms, it is easier to target a designated audience.

2.2 *Steganography*

Within the domain of information hiding, there are two primary methods for transmitting secret information: encryption and steganography. Encryption is the practice of obscuring secret data so that it is unintelligible, whereas steganography hides the existence of the exchanged information within a seemingly harmless message [Cac05]. More precisely, steganography is a method for supplementing encryption to prevent the existence of data from being detected [Con03].

Steganography is one of the oldest techniques for data hiding. A renowned Greek historian, Herodotus (485 – 525 BC), describes a story during the war between the Persian Empire and the Greek city-states where messengers would shave their heads,

write a message on their scalp, and then wait for the hair to regrow [Sel62]. In this way, the messenger could travel freely to their destination and transmit the covertly hidden message.

The classic model for hidden communication was first proposed by Simmons as the prisoners' problem [Sim84]. In his model, Alice and Bob have committed a crime and are kept in separate cells of a prison. They are only allowed to communicate with each other on paper via a warden named Wendy, with the restriction that they will not encrypt their messages. If the warden detects any suspicious communications in their messages then they will both be put into solitary confinement. To plan an escape, Alice and Bob need to communicate their secrets within the inspected messages in a way that avoids suspicion, such as through steganography.

Steganography techniques fall under the problem domain where the goal is to hide as much information as possible, bandwidth, while remaining immune to discovery. This goal is often referred to as robustness [KaP00]. Figure 3 is a conceptual view of the data-hiding problem. The tradeoff between bandwidth and robustness infers that as more data is hidden, the resulting scheme to do so will be a less secure or robust. On the other hand, the less data there is to hide, the more secure or robust the scheme will be. Due to this tradeoff, current steganography techniques define their systems based on how detectable they are to various methods as well as their throughput capabilities.

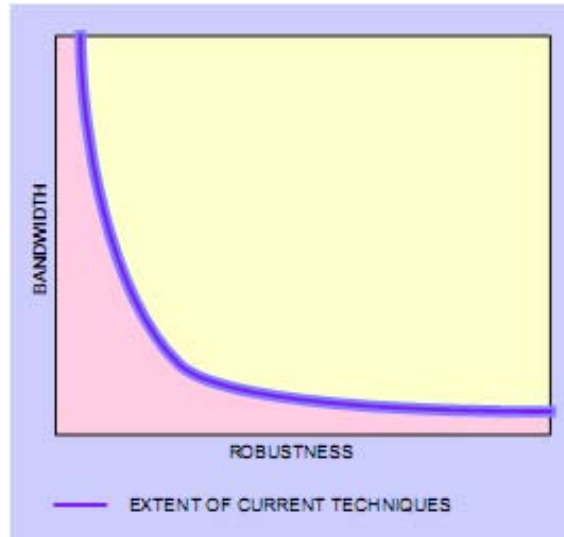


Figure 3: Data-hiding Problem Space [BGM+96]

2.2.1 Terminology

Steganography has a specific lexicon to describe different aspects of the system's components and functions. Outlined below are the terms used in this thesis when describing the steganographic system. Figure 4 shows a typical steganography system used to transmit and decode a secret message.

- **Cover.** The object being used to hide the secret information; this is often in the form of pictures [FPK07], audio files [Cve04], videos [NFN⁺04], or text documents [Cha97]. Discovery of the cover alone should not arouse suspicion.
- **Secret.** The file or message which is covertly hidden within the cover to evade detection. The secret is recoverable by the receiver.
- **Embedding Process.** Method used to conceal a secret within the cover. This process often includes compression as well as an encryption algorithm. While compression and encryption are not required to embed a secret, in recent years

this has become the de facto standard for nearly all steganographic systems [RaS10].

- **Stego Object.** The resulting data from the embedding process. This object contains the embedded secret information which should not be easily discoverable.
- **Extraction Process.** Method used to recover the secret from the stego object. The cover is usually discarded in this process.
- **Stego Key.** The secret key used to encrypt/decrypt the secret information in the embedding process. This key is usually shared between sender and receiver out-of-band from the stego object method of transmission.

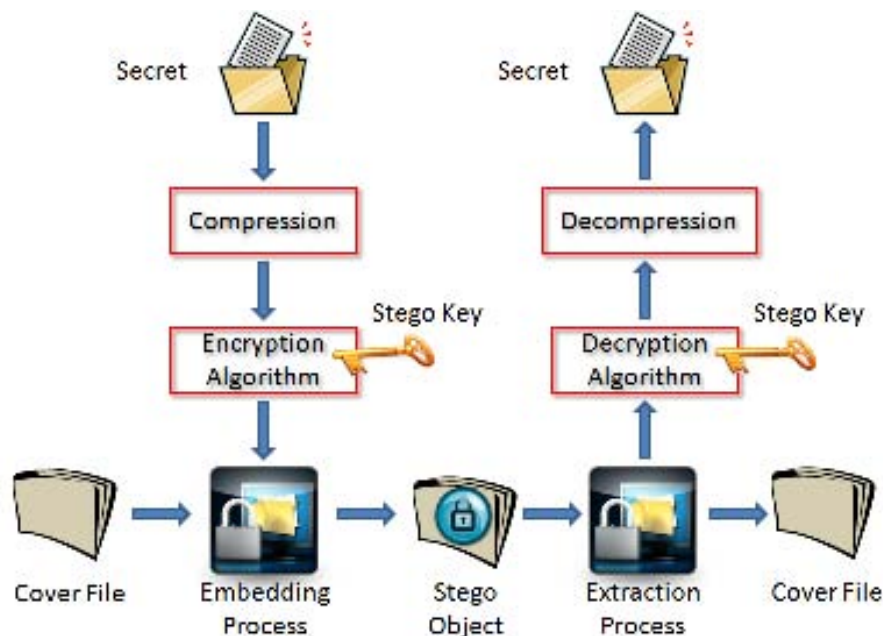


Figure 4: A Keyed Steganography System

2.2.2 Encrypted Steganographic Systems

Steganographic systems can be divided into three categories based on the encryption scheme used in the embedding process [KCC⁺07]. Pure Steganography is the weakest and least robust approach because it does not use any encryption; instead it relies on the assumption that parties other than the sender and intended receiver are unaware of the exchange of secret information.

Public Key Steganography (PKS) uses an asymmetric key algorithm to establish public and private keys to encrypt the secret information. In this scheme, the sender encrypts the secret with their public key which can only be decrypted with the corresponding private key from the intended receiver. This is the most robust approach but has the disadvantage of having the highest computational overhead as well as key management complexity. Properly exchanging keys is one of the major challenges for any keyed encryption technique [KaP00].

In Secret Key Steganography (SKS), also called symmetric key steganography, both the sender and receiver share or have agreed on a common set of stego keys prior to sending the stego object. The advantages of this approach includes the simplicity of key management as well as the difficulty for an adversary to conduct a brute force attack which would require excessive amount of computational power, time, and determination.

2.2.3 Text-Based Steganography

Within the last decade, steganography techniques have largely focused on hiding information within images because they provide significant pixel redundancy and are frequently transmitted over the Internet. However, images are not ideal when attempting

to exfiltrate large amounts of data for two reasons. For a technique that focuses on concealing as much information as possible, the image file grows proportionally to the amount of information being stored. Exceptionally large images are more likely to be noticed by network administrators. Additionally, the increase in hidden information potentially distorts the image itself, providing a clue to its hidden contents. On the other hand, if the approach focuses on robustness (i.e., keeping the size of the image intact) then only a few bits of information can be encoded per image.

Text data is still the largest bulk of digital data used and exchanged daily, spurred by the rise of information dissemination media such as email, blogs, and text messaging [KaP00]. The preponderance of this text media creates an attractive venue for covert communication channels and has emerged as a solution for transmitting information efficiently and securely.

Hiding data in text is an exercise in modifying the cover text so that the changes go unnoticed to readers. Various text steganography techniques were used during World War I and II to conceal messages from the enemy in case of interception. In World War I, the German Embassy in Washington, D.C. sent the following telegram messages to its Berlin headquarters [Sta05]:

"PRESIDENT'S EMBARGO RULING SHOULD HAVE IMMEDIATE NOTICE. GRAVE SITUATION AFFECTING INTERNATIONAL LAW. STATEMENT FORESHADOWS RUIN OF MANY NEUTRALS. YELLOW JOURNALS UNIFYING NATIONAL EXCITEMENT IMMENSELY.

APPARENTLY NEUTRAL'S PROTEST IS THOROUGHLY DISCOUNTED AND IGNORED. ISMAN HARD HIT. BLOCKADE ISSUE AFFECTS PRETEXT FOR EMBARGO ON BYPRODUCTS, EJECTING SUETS AND VEGETABLE OILS."

By concatenating the first character of every word in the first message and the second character of every word in the second message, the following concealed secret is retrieved [Sta05]:

'PERSHING SAILS FROM NY JUNE 1'

During World War II, invisible ink was used to write information on paper so that the paper appeared to be blank to the average person. To make the message viewable, liquids such as milk, vinegar, fruit juices or urine were used because, when heated, they darken and become visible to the human eye.

Text steganography techniques can be classified into two categories: linguistic or technical [Con03] [RaS10]. Linguistic, or Natural Language (NL), steganography manipulates the cover-text's lexical, syntactic, or semantic properties while preserving the meaning as much as possible to embed secret information, such as through synonym substitution [TTA06] [ShS08] [BoI04]. Technical-based methods use physical text formatting as a way to hide information, such as through the insertion of extra spaces [Kwa06], deliberate misspellings [TTA07] [Sha08], or resizing of the fonts throughout the text [RaS10].

Brassil et al. describe several text-based steganography methods which enable positive identification of copyright infringed documents [BLM⁺95] [BLM⁺99]. These techniques can also be used to transmit hidden messages. Line-shift coding is a way of altering a document by vertically shifting the locations of text lines to uniquely encode the document. Word-shift coding alters a document by horizontally shifting the locations of words within text lines.

Character coding, or feature specific coding, is a technique that is applied only on the bitmap image of the document and could be examined for chosen character features. Those features are altered or not altered depending on the codeword. For example, in Figure 5 the letter “r” in the word “Internet” is shifted down by 1/600 inch. The second line reveals the displacement more clearly by reprinting the word in a larger font size. A document marked in such an indiscernible way could be used to identify the copyright owner. If a document copy is suspected to have been illicitly disseminated, that copy could be decoded and the copyright owner identified.

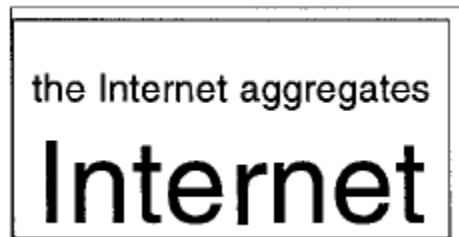


Figure 5: Example of character coding [BLM+99]

2.2.4 *Whitespace Steganography*

Whitespace steganography manipulates or inserts spaces (ASCII character 32), tabs (ASCII 9) and line feeds (ASCII 10). Because these characters are invisible to the casual observer, it is often advantageous to utilize them for covert communication. In general, whitespace encoding methods can be useful as long as the text remains in an ASCII format.

Bender, Gruhl, Morimoto, and Lu discuss a number of steganographic techniques for hiding data in cover text [BGM⁺96]. Figure 6 shows one of their techniques which places one extra whitespace after a line to conceal a secret ‘0’ or two extra whitespaces to

conceal a secret ‘1’ bit. This approach has the advantage that extra whitespaces at the end of lines typically go unnoticed to readers. While robust to interception, this approach has the disadvantage of not being ideal for transmitting large amounts of secret information because of the low encoding throughput, transmitting only one bit per line.



Figure 6: Whitespace encoded text [BGM+96]

Another technique discussed by [BGM⁺96] uses whitespaces throughout a justified cover text to encode a bit pattern, as shown in Figure 7. Data is encoded by controlling where extra spaces are placed. One space between words is interpreted as a ‘0’, and two spaces are interpreted as a ‘1’. This method results in higher throughput than the previous approach, able to encode several bits per line, however because this approach is limited to justified text documents, its usage is somewhat more limited.

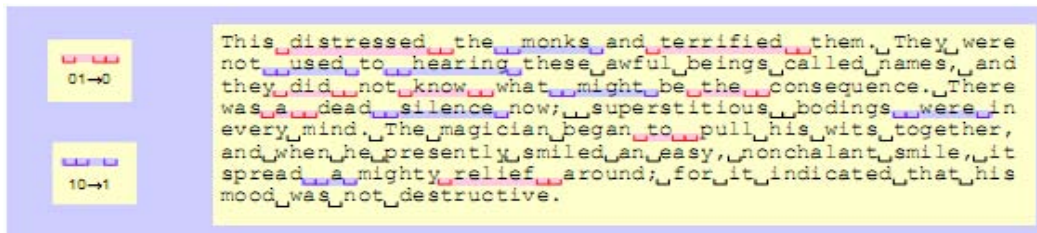


Figure 7: Data hidden through whitespace with justification [BGM+96]

WhiteSteg is a tool which uses a hybrid approach for whitespace encoding using both the spaces between words and between paragraphs [PAD08]. WhiteSteg dynamically generates cover-text using nursery rhymes according to the length of the secret message, up to 512 bytes. The chorus of rhymes is center-aligned because extra spacing is easier to hide in this configuration. This approach uses the better of two

different whitespace padding techniques but it cannot encode messages larger than 512 bytes.

Snow is an open-source whitespace steganographic program created by Kwan that is used to conceal messages or files in ASCII text by appending whitespaces and tabs to the ends of lines [Kwa06]. Since trailing spaces and tabs occasionally occur naturally in text, their existence should not raise suspicions to alert observers who stumble across them. Additionally, Snow embeds multiple bits per line depending on the cover traffic, which enables higher throughput than similar whitespace encoding techniques. The details of this algorithm are described in Chapter 4.

2.3 *Covert Timing Channels*

A covert channel is a communication medium, unintended by the system designer, that an attacker can use to transmit hidden messages from an entity in a higher security zone to an entity in a lower security zone [Kem83].

The difficulty of detecting or eliminating timing channels makes them a desirable choice for adversaries that value stealth over throughput. In particular, attackers may attempt to leak data over a network, through manipulating legitimate network streams to move sensitive information without detection.

Timing channels herein are assumed to be within a network environment. From this point forward, any reference to a covert timing channel (CTC), unless otherwise stated, will be channels that operate within a network. In general, CTCs modulate the time period between two consecutive packets in a network stream to encode a symbol. The time between consecutive packets is referred to as the inter-packet delay (IPD).

Since the IPD is the only part of the stream that is modified, CTCs are effective regardless of the actual packet payload, even if it is encrypted. However, this also means that at least two packets must be sent for each symbol. As such, the capacity of a CTC is significantly lower than standard communication protocols such as File Transfer Protocol (FTP).

The node that encodes the message into the target network stream is referred to as the sender, and the node that decodes the message is the receiver. For the transmission of the message to be successful the receiver must be positioned so that it is able to observe the IPDs of the stream at some point during its transit through the network. The receiver must also have a means to identify which stream contains the covert channel. It is important to note that the receiver is not necessarily the final destination of the stream; it could be positioned between the sender and an intended recipient.

Since the CTC is may be moving data from a higher security zone to a lower security zone, it is logical to assume that only simplex communication is available due to firewalls or the inability of the receiver to modify the stream. This provides the worst case scenario as the receiver cannot communicate directly with the sender. As a result, synchronization and error correction become more critical since the receiver cannot inform the sender of its current status.

2.3.1 Examples of Covert Timing Channels

Covert timing channels are either passive or active [GiW07]. Passive CTCs only modify the IPDs of existing network streams to encode the message, i.e., they do not generate any additional traffic. Conversely, active channels generate new traffic with

IPDs that match the symbols of the message. Intuitively, passive channels are harder to detect since they use legitimate streams and can thus evade intrusion detection systems and monitoring; however, they are also dependent on a process that the attacker may not control. This means they sacrifice capacity in exchange for increased detection resistance. Examples of both active and passive CTCs are given below.

2.3.1.1 IP Covert Timing Channel

The IP Covert Timing Channel, an active CTC developed by Cabuk et al. [CBS09], maps an arbitrary number of symbols to specific IPD values. This interval is known by both the sender and receiver. The specific choice of timing interval must balance channel capacity with the frequency of bit errors. If the interval is too small, network jitter could cause bits to be flipped, corrupting the message. Conversely, if the interval is too large, the capacity of the channel may be too small to be considered practical.

In the simple case of a binary scheme, a distinct IPD value is assigned to represent 0 and 1 bits, denoted as IPD0 and IPD1 respectively. In order to transmit a 1 bit, the sender transmits a packet such that the inter-packet delay between the current and previous packets will be equal to IPD1. A similar process is used to encode a 0 bit. Since only two different values are used to encode the bits, the timing IP CTC is easy to detect.

2.3.1.2 Model Based Covert Timing Channel

Gianvecchio et al. [GWW⁺08] proposed a framework for an active CTC that mimics the statistical properties of legitimate network streams, referred to as the model-

based CTC (MBCTC). The framework creates a model of legitimate traffic, which in turn helps determine the properties of the CTC.

To construct the CTC, the system first analyzes the target traffic type and finds the best fit for the IPD distribution. The secret is split into symbols that map to IPDs based on the inverse distribution function of the best fitting distribution. Finally, packets are sent using the calculated IPDs. Decoding is performed using the cumulative distribution function. The distribution can be changed over time to reflect any changes in the target traffic.

2.3.1.3 Jitterbug Covert Timing Channel

Jitterbug is a passive CTC that uses a hardware device [SMB06]. This device exploits a network-based timing channel to transmit a hidden message and is designed to transmit passwords and secret information over interactive network applications (e.g., SSH, X-windows). JitterBug is a keylogger and resides between the keyboard and the CPU. In interactive network applications, every keystroke generates a packet and by modifying the keystroke timing carefully, JitterBug encodes the message.

A timing window, w , determines the additional delay required to encode a message bit. A one bit is encoded by increasing inter-packet delay to a value modulo w milliseconds, a zero bit is encoded by increasing inter-packet delay by modulo $\lfloor \frac{w}{2} \rfloor$ milliseconds. The timing window w should be large enough to avoid errors induced by network jitter. Experimentally, Jitterbug achieves reliable communication with window sizes ranging from 2 to 20 milliseconds, depending on network delays.

2.3.1.4 Liquid Covert Timing Channel

Walls and Wright proposed a passive CTC that uses a portion of the compromised stream to smooth out the distortion that can be recognized by shape detection tests, called Liquid [WrW09]. Liquid uses half of the IPDs to encode the hidden message and the other half of the IPDs to increase detection resistance via entropy tests. In the entropy test, each IPD is mapped to one of M symbols, which are histograms of IPDs of legitimate traffic. Liquid keeps track of the mapped symbols during message encoding and generates other symbols based on prior history. Thus, the probability of a symbol being transmitted is nearly equal, increasing the entropy value.

2.3.2 Defense Against Covert Channels

Techniques to defend against covert timing channels can be categorized as either prevention or detection techniques. Prevention either eliminates the possibility of a channel or reduces the capacity to such a degree that using the channel is impractical. Detection techniques, on the other hand, attempt to identify active covert channels.

Kemmerer proposed the use of a Shared Resource Matrix to identify resources and entities that could be used as covert channels [Kem83] [Kem02]. This matrix is used at design time to assist system architects when creating multilevel secure systems. Adding noise to system clocks can reduce the capacity of a timing channel [Hu91]. A more aggressive prevention technique called the Pump is placed between two processes so that communications are intercepted and sent through a randomization scheme to perturb timing information [KaM93] [KMC05].

Active Warden removes storage channels from objects that have strict format definitions which allow the content of the objects to be objectively verified [FFP⁺03]. A practical example of such an object would be the packet header definition of a network protocol. Similar to Pump, Active Warden intercepts each of these objects as they are being transmitted. Before being forwarded to their destination, the Warden applies specific rules that may alter packet data to make the object's content more consistent. For example, a rule may specify that an unused header field must contain all zeros. When applying this rule, the Warden would zero all of the bits in that field.

Burke et al. [BGC05] investigated a simple statistical method for detecting covert timing channels when the legitimate network stream IPD's roughly fit a normal distribution. A histogram of IPDs composed of equal size bins is created and the shape of the histogram is compared to a desired network stream. The presence of a bimodal or multimodal distribution would suggest the existence of a covert timing channel. Using this method, the probability that a stream contains a covert channel is

$$P_{CovertChannel} = 1 - \left(\frac{C_{\mu}}{C_{max}} \right) \quad (1)$$

where C_{μ} is the count of the bin with the mean delay, and C_{max} is the count of the largest bin.

2.3.2.1 Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov (KS) test has also been used to determine if the distribution of a sample set of IPDs matched that of a legitimate set of IPDs [GiW07]. A difference in the distributions suggests the presence of a CTC in the sample. The KS test

is a non-parametric test that determines if two samples are from the same distribution. One major advantage of this test is that it does not rely on any assumptions in regards to the actual distribution of the samples. The KS test statistic measures the maximum distance between two empirical distribution functions (or a sample and a distribution)

$$KS\ TEST = \max|S_1(x) - S_2(x)| \quad (2)$$

where S_1 and S_2 are the empirical distribution functions of two samples. Since the KS test directly compares the empirical distribution functions, the samples do not need to be the same size. The KS test is useful for covert channel methods designed to avoid shape-based detection.

2.3.2.2 Regularity Test

A regularity test detects CTCs based on the variance of the IPDs is relatively constant [CBS04]. For most network traffic the variance of the IPDs changes over time, whereas with covert timing channels, if the encoding scheme does not change over time, the variance of the IPDs remains relatively constant. A sample is separated into sets of w inter-packet delays. Then for each w the standard deviation σ_i of the set is computed. The regularity is the standard deviation of the pair wise differences between each σ_i and σ_j for all sets $i < j$ or

$$regularity = STDEV\left(\frac{|\sigma_i - \sigma_j|}{\sigma_i}\right), i < j, \forall i, j \quad (3)$$

where σ_i is the standard deviation of the i^{th} window. A large difference from an established norm for legitimate traffic suggests the use of a CTC since similar traffic is expected to cluster around similar timing intervals. Covert timing channels using static

IPDs, such as $IPD_0 = 0$ and $IPD_1 = 1$, is easily detected with this method because the IPD variances will remain relatively constant over time, resulting in a low regularity value. Regularity is a widely accepted metric for detecting covert channels [CBS09] [GiW07] [GWW⁺08] [WrW09], and differs significantly from the KS test.

2.4 RC4 Encryption

Encryption in VANISH further obfuscates the hidden text so that even if the covert communication is discovered and analyzed, it is difficult to decipher. RC4 is chosen due to its simplicity, speed, and cryptographic strength.

RC4 was designed by Ron Rivest of RSA Security in 1987 [RSA10]. It uses a symmetric key stream-cipher algorithm which requires a secure exchange of a shared key which the algorithm does not provide. The key exchange is typically assumed to be securely swapped out-of-band from the IRC communication channel. A stream cipher is one of the simplest methods of encrypting data; each byte is sequentially encrypted using one byte of the key. Stream ciphers are ideal for this application because they produce variable length ciphertext which can be matched to any size plaintext message. Since its inception, RC4 has become one of the most widely used encryption standards. In fact, it is used to encrypt Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA) optionally, Transport Layer Security (TLS), BitTorrent protocol, Oracle SQL, Remote Desktop Protocol, and PDF documents.

The RC4 algorithm, shown in Figure 8, works in three phases: initialization, Key Scheduling Algorithm (KSA) and Pseudo-Random Generation Algorithm (PRGA) [Enc07]. During the initialization phase, an array containing 256 elements is generated

with unique byte codes in each element. The initialized array is run through the KSA which uses the secret key to scramble each element of the array. The 256-element array is then fed to the PRGA to further scramble the array and generate the keystream. An Exclusive OR (XOR) operation between the keystream and the plaintext, byte by byte, is performed to produce the ciphertext.

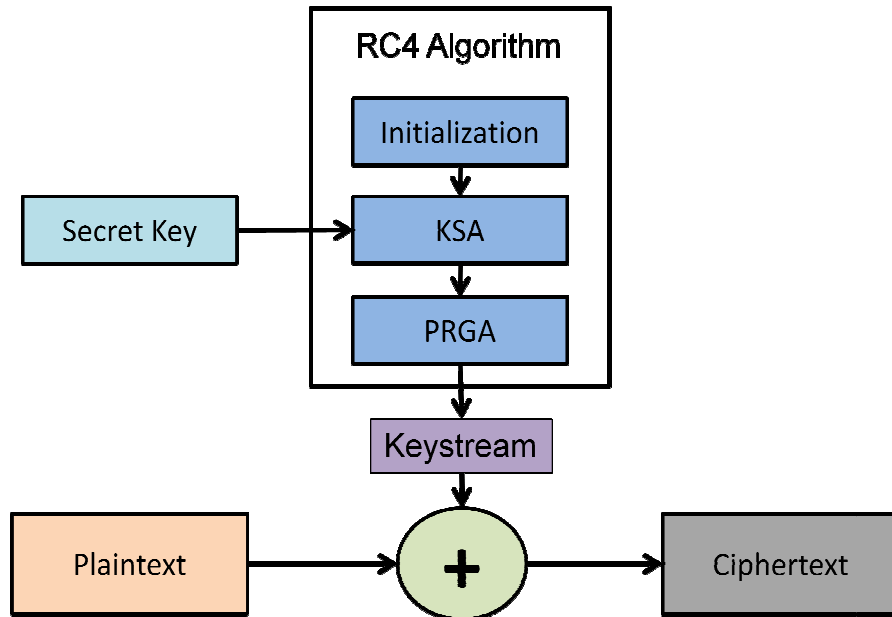


Figure 8: RC4 Encryption Algorithm

Decryption in RC4 is a straightforward process. Since the keystream generated by the algorithm is only dependent on the secret key, the receiver only has to know what key was used along with the ciphertext to recover the plaintext. An XOR operation is performed on the ciphertext with the generated keystream byte by byte to produce the original plaintext message. The logic for the decipher is

$$(A \text{ xor } B) \text{ xor } B = A \quad (4)$$

where A is the original secret message and B is the keystream produced by the RC4 algorithm.

2.5 *Summary*

This chapter presents background information on IRC and its networking protocol. Current usage trends among the top 10 IRC servers are discussed as well as its use and benefits to the military. Steganography is discussed along with current research in the area of text-based steganography. Next, covert timing channels are explored detailing several of the latest research efforts in this field. The details of two covert timing channel detection mechanisms are explained, the Kolmogorov-Smirnov and regularity tests. Finally, the chapter concludes with a background on the RC4 encryption scheme.

III. Methodology

This chapter presents the methodology to evaluate the performance of the VANISH system. Section 3.1 addresses the problem definition, goals, hypotheses, and approach. The System Boundaries and System Under Test (SUT) are defined in Section 3.2. Section 3.3 discusses the system services including their outcome, followed by a detailed description of the workload in Section 3.4. The performance metrics are presented in Section 3.5, system parameters in Section 3.6, and the factors involved are discussed in Section 3.7. A detailed explanation of the evaluation technique follows in Section 3.8. Section 3.9 addresses the experimental design for this research. Finally, the chapter is summarized in Section 3.10.

3.1 Problem Definition

The problem considered herein supposes an experienced enemy has gained control of a machine with intent to exfiltrate its files. Network probing determines that IRC traffic is enabled on the machine's firewall, so the adversary uses this vulnerability to covertly transmit the machine's files. Through social engineering or other means, the adversary learns the IRC server/channel which the machine usually connects to. The problem then is to engineer the traffic to appear legitimate to channel observers. Depending on the adversary's needs, the file extraction goal could be to transmit the files as quickly as possible or to use a more stealthy approach to minimize suspicion and detection.

3.1.1 Goals and Hypothesis

The objective of this research is to develop and evaluate new information hiding techniques. Specifically, the proposed system uses either steganographic or CTC encoding methods to exfiltrate data with either high capacity or high detection resistance, depending on the user's requirements. Two covert data exfiltration techniques over IRC are designed and analyzed against a common baseline.

The goals of this research are to:

- Construct an IRC steganographic system using the software algorithm from Snow [Kwa06], a high-throughput steganographic tool for whitespace encoding. This system is hereafter referred to as the Baseline scenario.
- Develop and optimize a steganographic system which maximizes the throughput capacity per message while remaining sufficiently undetectable in the channel, hereafter referred to as the Throughput scenario.
- Develop and optimize a covert timing channel system which maximizes detection resistance by modeling and blending in with legitimate network traffic by manipulating inter-packet delays, hereafter referred to as the Stealth scenario.

The three encoding techniques under test do not exhaust all possible information hiding capabilities over IRC. Rather, they show a range of capabilities by exploiting the IRC protocol in different ways while maximizing either throughput or stealth against a common baseline.

It is hypothesized that the Throughput scenario can approach the theoretical limits for transferring covert information over IRC by transmitting as much secret data as possible per message, as well as transmitting the messages as fast as possible within

server limitations. It is also hypothesized that the Stealth encoding algorithm can evade shape and regularity detection measures to within a 95% confidence interval by emulating legitimate traffic distributions.

3.1.2 Approach

The VANISH system is the framework for testing the Baseline, Throughput, and Stealth encoding techniques. As shown in Figure 9, the framework is a pipeline structure with three modules: encryption, encoding, and transmission. Encryption of the secret file adds a second layer of security to the covert messages so even if the covert traffic was detected and captured, it is difficult for the attacker to determine its contents without knowing the key. After encryption, the file passes to one of three modules which covertly encode the secret: Baseline, Throughput, or Stealth. The encoded file is sent to the transmitter module which connects to an IRC server and transmitting the encoded secret.

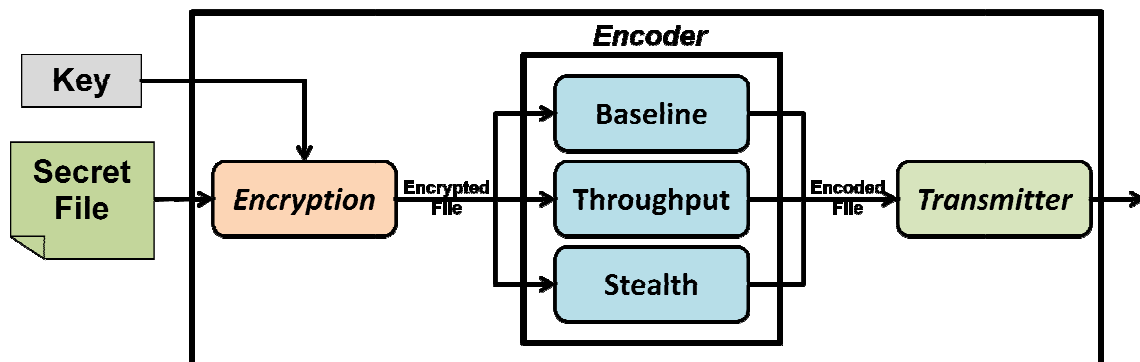


Figure 9: Framework to design VANISH sender

Once the IRC server receives the message, it is transmitted to all clients in the channel where the receiver awaits. The decoding process is performed on the receiving client with three modules: receiver, decoder, and decryption, as shown in Figure 10. The

receiver module connects to the IRC server and forwards all encoded messages from the sender client to the selectable decoder module. The sender and receiver are assumed to have agreed on an encoding technique prior to the transmission. The encoded messages are sent through the decoding process to produce the encrypted file which is decrypted using the same key to recover the secret file.

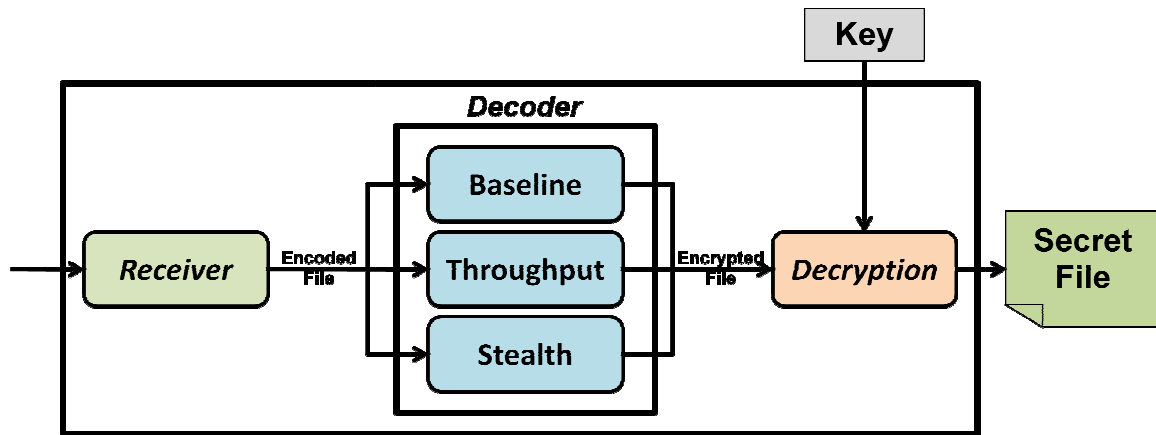


Figure 10: Framework to design VANISH receiver

VANISH is developed in the C as a user-space process which provides a platform independent solution benefitting from a strong heritage of simplicity and speed. The C language also allows for relatively simple file and socket API manipulation. The encoder module uses random number generation and random variable models from the GNU Scientific Library (GSL) [GNU10].

In this design, there is a unidirectional communication model for the covert data transfer. Note that only the covert communication is assumed to be unidirectional; IRC itself is bidirectional as all TCP/IP packets are acknowledged. A unidirectional covert channel means the receiver cannot communicate directly with the sender which increases the difficulty in achieving an error-free transfer. In particular, the receiver cannot

acknowledge the correct receipt of covert packets, rate limit the sender, or indicate when to resynchronize. These challenges, as well as those imposed by the IRC server (such as traffic throttling and enforced maximum packet size), create a complex network of design variables for the covert communication system.

Figure 11 shows the experimental methodology and tests which are elaborated in the following sections. For the purposes of evaluating throughput and detectability, the cover traffic is not crucial to the experiments. In practice however, the cover should be germane to the channel so it does not raise suspicion.

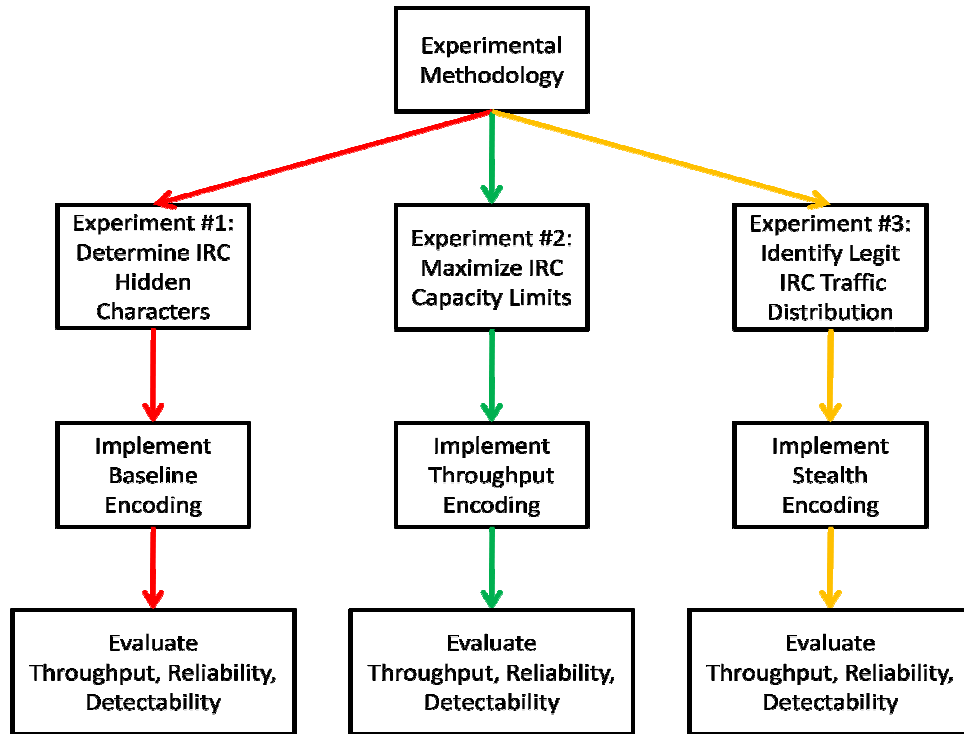


Figure 11: Experiments and Tests for Research Methodology

3.1.2.1 Experiment #1: Determine IRC Hidden Characters

The first experiment uses a modified whitespace encoding algorithm from the steganographic tool Snow [Kwa06] to transmit hidden messages over an IRC channel. Snow uses whitespaces and tab characters to hide messages at the ends of lines in text

documents. However, in a preliminary study, tab characters are found to be viewable in IRC clients; therefore, an analysis of all ASCII characters is conducted to find a suitable replacement hidden character. To demonstrate wide-spread applicability, the two most popular IRC client applications for Windows and Linux OS, mIRC and Xchat, are tested to identify shared non-viewable characters. The results of this experiment and details of the Baseline algorithm are discussed in Chapter 4.

3.1.2.2 Experiment #2: Maximize IRC Capacity Limits

For the Throughput scenario, a fictional adversary intends to exfiltrate the greatest amount of information, *capacity*, over IRC with little regard to detectability. Maximizing capacity requires sending the most information per packet as well as per time. An analysis of the server throttling limitations is conducted to determine the maximum speed messages can be sent to the channel. Shared non-viewable ASCII characters found from Experiment #1 are used to encode covert data. The results of this experiment and details of the Throughput algorithm are discussed in detail in Chapter 4.

3.1.2.3 Experiment #3: Identify Legitimate IRC Traffic Distribution

The Stealth scenario exploits the underlying network protocols of IRC to covertly transmit data based on the timing of messages sent and received. This approach sacrifices data rate to improve detection resistance by manipulating traffic timing patterns so they are statistically indistinguishable from normal traffic. To determine the distribution of legitimate IRC traffic, chat logs of observed IRC channels are captured and examined for best fit analysis. The distribution of the legitimate network traffic is

modeled by the Stealth encoding algorithm as it transmits covert data. The details of this experiment and the Stealth algorithm are discussed in Chapter 4.

3.2 System Boundaries

The System Under Test (SUT) is the VANISH System. A block diagram is shown in Figure 12. It consists of the following components: the encoding technique, encryption software, the computer which the experiments are run on, and the attached Ethernet network. This research uses an Ethernet network, but the techniques are equally applicable to other networks, such as 802.11. The Component Under Test (CUT) is the encoding technique which hides the covert message. The CUT is designed and optimized for each technique to compare system responses.

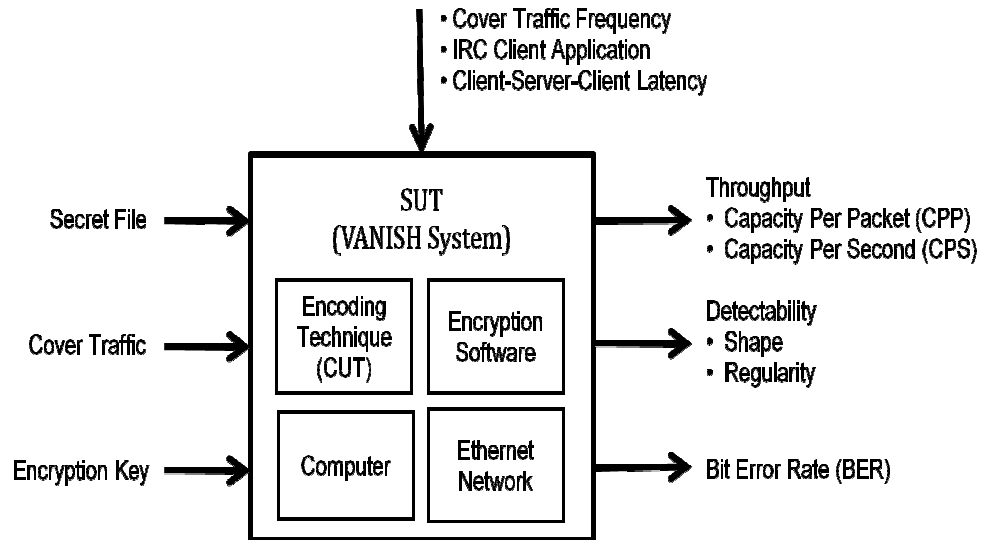


Figure 12: The VANISH System

Workload parameters include the secret file to encode, the IRC cover traffic, and the encryption key. The workload parameters are discussed in more detail in Section 3.4. The system parameters consist of the cover traffic frequency in the channel, the IRC

client application in use, and the latency between the sender and receiver clients via the IRC server. These parameters are discussed in more detail in Section 3.6. The desired system responses are characterized by the throughput in capacity per packet (CPP) and capacity per second (CPS), detectability using the shape and regularity tests, and bit error rate (BER) as a percentage of errors versus the total number of covert messages transmitted. The performance metric details are discussed in Section 3.5.

3.3 System Services

This system provides a covert communication service using steganographic or covert timing channel techniques over the IRC protocol. The service is successful when a secret is encrypted, encoded and transmitted to a designated IRC channel such that it is undetectable by the client application software. The intended receiver must correctly capture, decode, and decrypt the secret to extract the original message. The service fails when the secret cannot be properly encoded, the encoded message is viewable in the channel, or when the intended recipient cannot recover the secret message.

3.4 Workload

The workload of the SUT consists of three parts: the secret file to transmit, the cover traffic, and the encryption key. The secret file workload is defined by its file size and file type. The amount of cover traffic needed to disguise the data transfer is directly proportional to the secret's file size. The secret file type, on the other hand, has a lesser effect on performance because the encoder treats all file contents as bits. The content of the cover traffic is also very important to the success of the system. Since the motivation for this research is to identify methods for *covertly* communicating over public IRC

channels, the cover traffic must not be unusual or arouse suspicion for the given channel, i.e., it should be consistent with existing traffic. For example, users in a channel created for physics discussions may find it unusual if messages regarding home cooking were sent. The encryption key exchange between sender and receiver is assumed to take place securely and out-of-band from the IRC communication channel.

3.5 Performance Metrics

Performance metrics establish the impact of the experimental scenarios. Information hiding techniques generally focus on concealing as much information as discretely and accurately as possible. Therefore, three types of performance metrics are collected for this SUT to determine detectability, throughput, and bit error rate (BER).

Two metrics for throughput are collected for each encoding method: CPP and CPS. Capacity is the amount of covert information in the secret file in bits. Therefore, CPP is a function of the size of the secret file and amount of covert information embedded per packet, while CPS is dependent on the size of the secret file and the rate at which the covert packets are transmitted to the IRC server. Reliability of the VANISH system is measured by its bit error rate: the number of bit errors from the covert file transfer as measured by the receiver after forward error correction.

For the encoding methods described in this thesis to be effective, their covert traffic must be non-viewable to observers in the public IRC channel. However, deep-packet network inspection or other analysis techniques may successfully identify the covert channel, as demonstrated in Chapter 4. The detection metrics employed in this

thesis are commonly used to identify covert timing channels. These include the Kolmogorov-Smirnov test for shape and the regularity test for variance.

The Kolmogorov-Smirnov test is run 100 times for each experimental test against randomly generated distributions which use the best-fit parameters from legitimate traffic samples. Each test produces a p-value as a measure of the significance of the null hypothesis that both distributions are different. The higher the p-value, the better the generated covert traffic fits the distribution of the observed traffic. The mean IPD, standard deviation, standard error of the mean, and confidence interval of each test are reported.

In previous research [CBS04] [GiW07], low regularity was used as an indicator of a CTC. Therefore, the regularity of the legitimate traffic sample herein is tested and compared with the covert traffic for each encoding method with window size $w = 25$. The covert traffic for each method is deemed to pass the regularity test if its score is within 10% of the legitimate traffic's regularity score.

3.6 Parameters

The parameters of the system are the properties which, when changed, impacts the performance of the system. These include both system parameters and workload parameters. The workload and system parameters for the SUT are described below.

3.6.1 Workload Parameters

The workload of the SUT consists of the secret file to transmit over the channel, the encryption key, and the IRC cover traffic which will conceal the presence of the hidden data.

- Secret File: This research uses three 3 kilobyte randomly generated files as the secrets for experiments. A randomly generated file has the advantage of not making any assumptions on the formatting of a particular file and therefore is not file type dependent. The system is also not dependent on the file size, however larger size files require more cover traffic. A 3 kilobyte file is chosen because it is large enough to accurately measure capacity, detectability, and bit error rate for each of the encoding scenarios without requiring excessive redundant effort.
- IRC Cover Traffic: This is the representative cover traffic germane to the public IRC channel the covert communication will take place over. The length, in number of messages sent to the IRC channel, is dependent on the size of the secret being transmitted and the covert encoding method selected. Higher throughput methods require less cover traffic to transmit the secret. In preliminary tests, a 3 kilobyte secret file requires 1289 messages with the Baseline method, 30 messages with the Throughput method, and 27,648 messages with the Stealth method. When there are no more secret bits to transmit, the cover traffic will cease.
- Encryption Key: The encryption key is supplied to the system prior to encoding. To successfully decrypt the secret file, the receiver must use the same key used during encryption. The key used in the experiments is “thesis”.

3.6.2 *System Parameters*

System parameters that affect the performance of the SUT include:

- Cover Traffic Frequency: The frequency of cover traffic has a direct impact on the amount of secret messages sent. Each encoding technique uses different transmission frequencies to achieve their respective objectives of throughput or detection resistance. The Baseline and Throughput transmission frequency is one second per packet while the Stealth method transmits at approximately four seconds per packet. The details of these methods are discussed in Chapter 4.
- IRC Client Application: The top two IRC client applications for Windows and Linux, mIRC and Xchat, respectively, are tested for in-channel detection and cross-platform reliability.
- Client-Server-Client Latency: The distance between the sender, IRC server, and recipient may affect the reliability (BER) of the CTC encoding. As such, two different wide area networks (WAN) are tested for each encoding method. The first scenario, referred to as WAN-US, uses a public IRC server in Chicago, Illinois (66.225.225.225) and a transmitter/receiver located in Dayton, OH. The second WAN connection, WAN-EU, is between a public IRC server located in Amsterdam, Netherlands (194.109.129.220) and a transmitter/receiver are located in Dayton, Ohio. Table 2 shows the test network scenarios physical distance (round-trip) and average round trip time (RTT). Because ping and traceroute requests are routinely dropped by IRC servers, RTT, measured in milliseconds, is determined by transmitting/receiving on the same machine and averaging packet timing differences over 100 samples.

Table 2: Test Network Scenarios

Test Network	WAN-US	WAN-EU
Distance	600 miles	8,162 miles
RTT	16.458 ms	90.236 ms

3.7 Factors

Factors provide insight into the impact of the different encoding techniques without requiring excessive or redundant effort. Table 3 shows the factors used and their associated levels for the experiments. The factors include the three CUT encoding techniques, three randomly generated 3 KB secret files, and the test network scenario.

Table 3: Factor Levels

Factor	Level 1	Level 2	Level 3
Encoding Configuration	Baseline Encoding	Throughput Encoding	Stealth Encoding
Random 3 KB Secret File	File #1	File #2	File #3
Test Network Scenario	WAN-US	WAN-EU	

The encoding technique is designated as the Component Under Test. It controls all aspects of the system relating to covert throughput and the detectability of the covert communication. The encoding configuration has three levels: Baseline, Throughput, and Stealth encoding. The Baseline encoding technique is used as a reference point to compare the other encoding algorithms against. Throughput and Stealth encoding will show a range of capabilities maximizing either throughput or detection resistance. Three randomly generated 3 KB files test whether the encoder functions independently from the

secret file contents. The two network scenarios specified will determine the transmission reliability of the encoding techniques.

3.8 Evaluation Technique

Direct measurement is the evaluation technique used in this research. More accurate results can be obtained using actual servers rather than a simulator or analytic model. Simulation is not practical due to the overhead of creating a virtual computer and network environment. In addition, servers often respond differently in simulation due to network latencies. Thus, direct measurement is the most accurate way for analyzing performance in throughput, reliability, and detectability to evaluate the systems.

The VANISH system generates covert messages and transmits them to the IRC server where it propagates to the users in the channel. Wireshark, a network sniffer, is installed on the transmitting and receiving IRC client computers. Wireshark captures and validates that the system operates properly. Detectability with the Kolmogorov-Smirnov and regularity tests is determined offline through statistical analysis of the captured Wireshark network traffic using R [Rpr10]. Throughput and BER is evaluated using Wireshark captures of the transmission to obtain accurate time stamps and the received covert file. CPP is determined by taking the file size divided by the number of IRC messages required to send the file. CPS is determined by taking the file size divided by the time required to send the file.

Experiments use two computers running Windows XP Service Pack 2 in the configuration shown in Figure 13. Each client connects to the test IRC server through their network router and the internet.

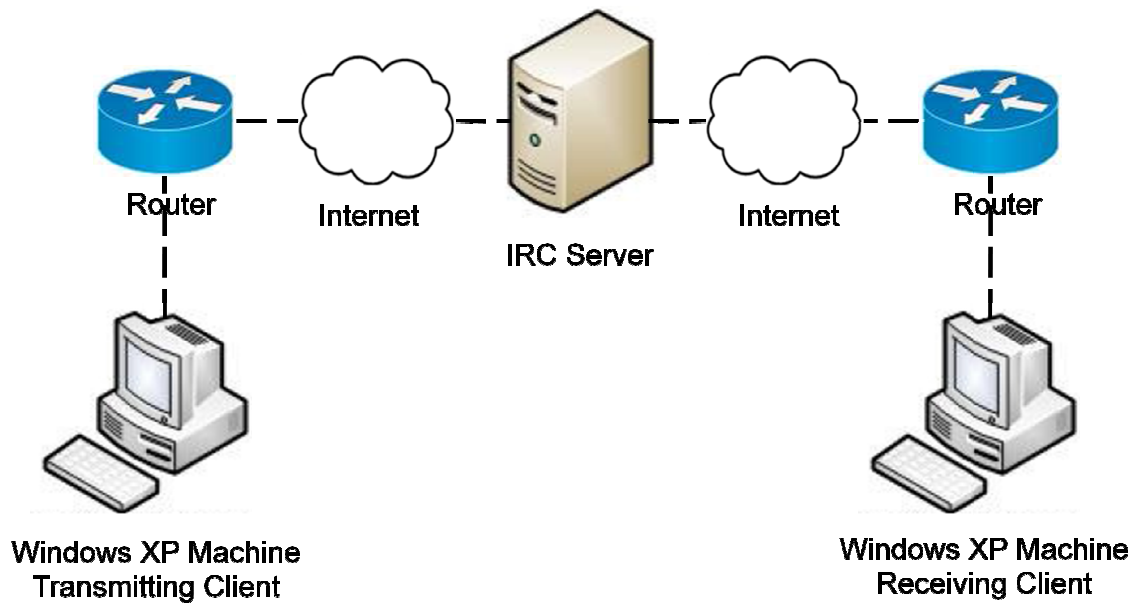


Figure 13: Experimental Configuration

The computer specifications for the transmitting and receiving clients:

- Hewlett-Packard C8510P Laptop
- Intel Core 2 Duo CPU TT700 @ 2.40 GHz
- 2 GB RAM
- Intel 82566 Gigabit Network
- 120 GB SATA Hard Drive
- Wireshark Packet Sniffer

3.9 *Experimental Design*

This experiment uses a full factorial design. Given the factors listed in Section 3.7, 18 trials are required to collect data for given combination of factors:

$$3 \text{ encoding techniques} * 3 \text{ Secret Files} * 2 \text{ network scenarios} = 18 \text{ trials}$$

Preliminary experiments show that network latency results in some variance in IRC server responses when forwarding messages. Therefore, three repetitions are used to characterize that variance and a confidence level of 95% is used. Thus, 54 experimental trials are needed to execute all cases.

3.10 Summary

This chapter introduces the methodology to accomplish the goals of this research. The design of the experiment is discussed in detail including the metrics, factors, parameters, and evaluation technique. Three unique information hiding techniques over IRC are evaluated by direct measurement based on throughput, detection resistance, and bit error rate. A full factorial experiment defines 18 trials to collect the metrics using the following factors: encoding technique, file type, file size, and network scenario. Each experiment is run three times.

IV. Software Architecture and Design

This chapter presents the pilot studies, experiments, and details of the VANISH system algorithms. Since many of the covert techniques are unique to this thesis, it is appropriate to describe the underlying software and algorithms in detail. Section 4.1 discusses an experiment to find platform independent ASCII characters not visible over IRC for use in the Baseline and Throughput algorithms. Section 4.2 describes the details of the Baseline encoding algorithm, followed by an experiment maximizing channel capacity and particulars of the Throughput algorithm in Section 4.3. Section 4.4 describes the third experiment which analyzes legitimate IRC traffic IPDs for best-fitting distribution parameters. The section continues with the details of the Stealth encoding algorithm using these parameters. Finally, the chapter is summarized in Section 4.5.

4.1 *Baseline Encoding Algorithm*

This section describes the experiment, pilot study, and results used to construct the Baseline encoding method which uses non-viewable ASCII characters to relay messages over IRC in Section 4.1.1. The details of the Baseline algorithm are presented in Section 4.1.2.

4.1.1 *Experiment #1: Determine IRC Hidden Characters*

There are two goals of this experiment: 1) to find which ASCII characters are not viewable in IRC client applications, and 2) to measure their effect on viewable characters. Because most users read chat messages over their IRC client, any non-viewable characters are effectively hidden.

In a pilot study, the choice of an IRC client made a difference in which characters are viewable in the channel. Due to the large number of IRC clients available, both standard and highly customizable, the experiment is performed using the two most widely used IRC applications for Windows and Linux OS: mIRC and Xchat [RAD⁺10]. ASCII characters 0x00h – 0xFFh are sent to an IRC channel via socket programming where clients using mIRC and Xchat determine channel detectability. For reference, a complete ASCII table is shown in Figure 14.

Regular ASCII Chart (character codes 0 - 127)

000d 00h	(nul)	016d 10h	▸ (dle)	032d 20h	sp	048d 30h	0	064d 40h	@	080d 50h	P	096d 60h	`	112d 70h	p
001d 01h	☉ (soh)	017d 11h	◀ (dc1)	033d 21h	!	049d 31h	1	065d 41h	A	081d 51h	Q	097d 61h	a	113d 71h	q
002d 02h	● (stx)	018d 12h	⋮ (dc2)	034d 22h	"	050d 32h	2	066d 42h	B	082d 52h	R	098d 62h	b	114d 72h	r
003d 03h	♥ (etx)	019d 13h	!! (dc3)	035d 23h	#	051d 33h	3	067d 43h	C	083d 53h	S	099d 63h	c	115d 73h	s
004d 04h	♦ (eot)	020d 14h	‡ (dc4)	036d 24h	\$	052d 34h	4	068d 44h	D	084d 54h	T	100d 64h	d	116d 74h	t
005d 05h	♣ (enq)	021d 15h	§ (nak)	037d 25h	%	053d 35h	5	069d 45h	E	085d 55h	U	101d 65h	e	117d 75h	u
006d 06h	▲ (ack)	022d 16h	■ (syn)	038d 26h	&	054d 36h	6	070d 46h	F	086d 56h	V	102d 66h	f	118d 76h	v
007d 07h	• (bel)	023d 17h	⋈ (etb)	039d 27h	'	055d 37h	7	071d 47h	G	087d 57h	W	103d 67h	g	119d 77h	w
008d 08h	◼ (bs)	024d 18h	↑ (can)	040d 28h	(056d 38h	8	072d 48h	H	088d 58h	X	104d 68h	h	120d 78h	x
009d 09h	(tab)	025d 19h	↓ (em)	041d 29h)	057d 39h	9	073d 49h	I	089d 59h	Y	105d 69h	i	121d 79h	y
010d 0Ah	(lf)	026d 1Ah	(eof)	042d 2Ah	*	058d 3Ah	:	074d 4Ah	J	090d 5Ah	Z	106d 6Ah	j	122d 7Ah	z
011d 0Bh	♂ (vt)	027d 1Bh	← (esc)	043d 2Bh	+	059d 3Bh	;	075d 4Bh	K	091d 5Bh	[107d 6Bh	k	123d 7Bh	{
012d 0Ch	♀ (np)	028d 1Ch	~ (fs)	044d 2Ch	,	060d 3Ch	<	076d 4Ch	L	092d 5Ch	\	108d 6Ch	l	124d 7Ch	
013d 0Dh	(cr)	029d 1Dh	↔ (gs)	045d 2Dh	-	061d 3Dh	=	077d 4Dh	M	093d 5Dh]	109d 6Dh	m	125d 7Dh	}
014d 0Eh	↓ (so)	030d 1Eh	▲ (rs)	046d 2Eh	.	062d 3Eh	>	078d 4Eh	N	094d 5Eh	^	110d 6Eh	n	126d 7Eh	~
015d 0Fh	□ (si)	031d 1Fh	▼ (us)	047d 2Fh	/	063d 3Fh	?	079d 4Fh	O	095d 5Fh	_	111d 6Fh	o	127d 7Fh	␣

Extended ASCII Chart (character codes 128 – 255; Codepage 850)

128d 80h	Ç	144d 90h	Ë	160d A0h	á	176d B0h		192d C0h	¼	208d D0h	Ɔ	224d E0h	Ó	240d F0h	-
129d 81h	û	145d 91h	æ	161d A1h	í	177d B1h		193d C1h	½	209d D1h	Ɔ	225d E1h	Ô	241d F1h	±
130d 82h	é	146d 92h	Æ	162d A2h	ó	178d B2h		194d C2h	¾	210d D2h	Ɔ	226d E2h	Õ	242d F2h	ˆ
131d 83h	â	147d 93h	ô	163d A3h	ú	179d B3h		195d C3h	1	211d D3h	Ɔ	227d E3h	Ö	243d F3h	‰
132d 84h	ã	148d 94h	õ	164d A4h	ü	180d B4h	+	196d C4h	2	212d D4h	Ɔ	228d E4h	Ø	244d F4h	‰
133d 85h	ä	149d 95h	ö	165d A5h	ÿ	181d B5h	À	197d C5h	3	213d D5h	Ɔ	229d E5h	Ù	245d F5h	‰
134d 86h	å	150d 96h	ù	166d A6h	ÿ	182d B6h	Á	198d C6h	4	214d D6h	Ɔ	230d E6h	Ú	246d F6h	‰
135d 87h	ç	151d 97h	û	167d A7h	ÿ	183d B7h	Â	199d C7h	5	215d D7h	Ɔ	231d E7h	Û	247d F7h	‰
136d 88h	è	152d 98h	ÿ	168d A8h	ÿ	184d B8h	Ã	200d C8h	6	216d D8h	Ɔ	232d E8h	Ü	248d F8h	‰
137d 89h	ë	153d 99h	ÿ	169d A9h	ÿ	185d B9h	Ä	201d C9h	7	217d D9h	Ɔ	233d E9h	Ý	249d F9h	‰
138d 8Ah	è	154d 9Ah	ÿ	170d AAh	ÿ	186d BAh	Å	202d CAh	8	218d DAh	Ɔ	234d EAh	ÿ	250d FAh	‰
139d 8Bh	ï	155d 9Bh	ø	171d ABh	ÿ	187d BBh	Æ	203d CBh	9	219d DBh	Ɔ	235d EBh	ÿ	251d FBh	‰
140d 8Ch	ï	156d 9Ch	ø	172d Ach	¼	188d BCh	À	204d CCh	10	220d DCh	Ɔ	236d ECh	ÿ	252d FCh	‰
141d 8Dh	ï	157d 9Dh	ø	173d ADh	½	189d BDh	À	205d CDh	11	221d DDh	Ɔ	237d EDh	ÿ	253d FDh	‰
142d 8Eh	ÿ	158d 9Eh	×	174d AEh	¾	190d BEh	À	206d CEh	12	222d DEh	Ɔ	238d EEh	ÿ	254d FEh	‰
143d 8Fh	ÿ	159d 9Fh	f	175d AFh	ÿ	191d BFh	À	207d CFh	13	223d DFh	Ɔ	239d EFh	ÿ	255d FFh	‰

Figure 14: ASCII Table Reference

Measuring the effect of non-viewable characters on viewable ones is of primary importance for the steganographic encoding methods. If certain non-viewable characters make obvious and repeated changes to the cover traffic messages, then its perturbations could alert channel observers over time. To measure this effect, a control character,

ASCII ‘0’ (0x30h), is used both before and in-between ASCII test characters in the configuration 0xx0x, where 0 is the control character and x is the test ASCII character. Ideal characters identified in this experiment would only show the presence of the control characters in the IRC message (i.e., 00).

The results of the experiment are shown in Table 4. A total of eight ASCII characters are non-viewable in mIRC and seven in Xchat. Three categories of covert characters are identified: *whitespace*, *end of line*, and *flexible* characters. *Whitespace* characters produce extra whitespace in the channel but are non-viewable when used at the ends of a line. *End of line* characters mask viewable characters that are used after them. *Flexible* characters do not affect the control and can be used anywhere in the message.

Table 4: Hidden Character Categories

IRC Client Software	Whitespace Characters	End of Line Characters	Flexible Characters
mIRC	0x20h	0x03h, 0x1Bh	0x02h, 0x0Fh, 0x16h, 0x1Fh, 0xA0h
Xchat	0x20h, 0x09h	0x03h	0x07h, 0x0Fh, 0x16h, 0x1Fh

The differences between the three hidden character categories are shown in Figure 15, which displays three screenshots of IRC channel messages when a character from each category is transmitted: 0x20h, 0x03h, and 0x02h. In the left image, two whitespace characters offset the spacing between the control characters showing a viewable disturbance. The middle image shows the effect of the end of line characters, the second control ‘0’ character is hidden after the 0x03h characters are used. The image on the

right displays the results of flexible characters which perfectly mask their presence in IRC channel traffic without offsetting the control characters.

Whitespace	End of Line	Flexible
<test123> 0 0	<test123> 0	<test123> 00

Figure 15: Hidden Character Output, Test Characters 0x20h, 0x03h, and 0x02h

Whitespace characters can be used for covert communication over IRC, however there are two channel detectability issues. One, the whitespaces must be used at the end of the cover traffic to remain hidden to avoid excessively perturbing cover traffic. Two, the extra whitespace should not cause the message to wrap around to a new text line. For the Baseline encoding algorithm, these issues are accounted for to adhere to the Snow steganographic algorithm.

To ensure a cross-platform compatible solution, only characters which are non-viewable in both Xchat and mIRC are used in the steganographic encoding algorithms: 0x20h, 0x03h, 0x0Fh, 0x16h, and 0x1Fh. One limitation for all covert techniques that use non-viewable ASCII characters is that it is an easy way for a defender to detect and block these messages from traversing their network through simple intrusion detection rules filtering packets with these bytes in the payload.

4.1.2 Baseline Encoding Algorithm Details

This section discusses the details of the Baseline encoding algorithm. The Baseline encoding algorithm is created based off of the high-throughput whitespace steganographic tool Snow [Kwa06], which conceals messages in text documents by appending whitespaces (0x20h) and tabs (0x09h) to the ends of lines. Because spaces

and tabs are generally not visible in text viewers, the message is effectively hidden from casual observers. Figure 16 reveals extra whitespaces which were steganographically encoded into the test document when all characters in the file are selected. The secret message is successfully embedded in the whitespace of the test document because there is no overflow of whitespaces causing line wrapping or other cover traffic distorting effects. Snow overcomes the issue of line wrapping by keeping track of how many characters are in each line and only adds spaces or tabs when the length of the line will not exceed the default line width of 80.



Figure 16: Snow Steganographic Encoding

The Baseline encoding algorithm modifies Snow by exchanging tabs characters, 0x09h, with the character 0x0Fh because tab characters are viewable in mIRC as found in Experiment #1 (Section 4.1). Figure 17 shows a mIRC screen-capture with tab test characters displayed as black blocks. The black blocks are clearly viewable in the IRC channel. Therefore, tabs are not used in the encoding algorithm.

<test123> 0000

Figure 17: IRC Character Test - Tab 0x09h

To begin encoding a secret into the cover text, a 0x0Fh character is added immediately after the text on the first line where it fits. This prevents a problem from occurring if the user enters an extra space after their message since a trailing 0x0Fh must

be found before extraction begins. The 0x0Fh character cannot be input into an IRC message other than through socket programming.

Three bits of the secret are encoded at a time through a 3-to-1 multiplexer which inserts the number of whitespaces according to Table 5. Any message not a multiple of 3 bits is padded by zeroes. Additional 0x0Fh characters are then used to separate sequences of whitespaces. For example, a bit sequence of “010” is encoded as “0x0Fh 0x20h 0x20h 0x0Fh”.

Table 5: Bit Sequence 3-to-1 Whitespace Multiplexer

Bit Sequence	# of Whitespaces
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

The receiver client parses traffic from the sender and performs a de-multiplex operation to recover the original message. A Wireshark capture of the Baseline encoding algorithm in operation (minus encryption) is shown in Figure 18. The left capture is from the sender’s computer and the right is from receiver’s computer. Here, the cover traffic message “Generated with” is sent to the channel “#test1234q” and is appended with the covert characters 0x20h and 0x0Fh. The secret message for this test is “hi”. Trailing 0x0Ah and/or 0x0Dh characters denote the ends of messages to IRC applications. Using Table 5 to verify that the encoding algorithm performed correctly, the secret message

“hi” is converted to binary, 01101000 01101001*b*, and every three bits of the secret is encoded as a series of whitespaces separated by 0x0F characters. The manual encoding of this secret (“011” “010” “000” “110” “100” “1”) should have the following sequence of whitespaces: 3, 2, 0, 6, 4, 4 (the last bit is padded with two zeros). The results of the manual encoding agree with the Wireshark captures.

```

0000 00 24 b2 1d f6 78 00 21 5c 7d 3e d7 08 00 45 00 .$.x.! \}>...E. 0000 00 21 5c 7d 3e d7 00 24 b2 1d f6 78 08 00 45 00 .!\}>.$ ...x..E.
0010 00 64 d8 88 40 00 80 06 3b 9c c0 a8 01 04 42 e1 .d.@... ;....B. 0010 00 98 96 6a 40 00 36 06 c7 86 42 e1 e1 c0 a8 ...j0.6. ..B....
0020 e1 e1 0f 95 1a 0b 0c 68 a8 d1 b0 a2 95 2d 50 18 .....h .....-P. 0020 01 04 1a 0b 0f 94 cf a5 be ef 41 2f 26 cc 50 18 ..... ..A/&.P.
0030 ff ff a8 76 00 00 50 52 49 56 4d 53 47 20 23 74 ...v.,PR IMSG #t 0030 ff ff e7 43 00 00 3a 63 61 70 74 31 32 33 21 7e ...C.:c apt123!~
0040 65 73 74 31 32 33 34 71 20 3a 47 65 6e 65 72 61 est1234q :Genera 0040 63 61 70 74 31 32 33 40 63 70 65 2d 31 38 34 2d capt1230 cpe-184-
0050 74 65 64 20 77 69 74 68 0f 20 20 20 0f 20 20 0f 35 38 2d 32 38 2d 31 37 32 2e 77 6f 68 2e 72 65 58-28-17 2.woh.re
0060 0f 20 20 20 20 20 0f 20 20 20 20 0f 20 20 0f 73 2e 72 72 2e 63 6f 6d 20 50 52 49 56 4d 53 47 s.rr.com PRIVMSG
0070 20 0a . . . 0070 20 23 74 65 73 74 31 32 33 34 71 20 3a 47 65 6e #test12 34q :Gen
    . . . 0080 65 72 61 74 65 64 20 77 69 74 68 0f 20 20 0f erated w ith. .
    . . . 0090 20 20 0f 0f 20 20 20 20 20 20 20 0f 20 20 0f .. with. .
    . . . 00a0 20 20 20 20 0d 0a . . .

```

Figure 18: Baseline Encoding Wireshark Capture

Figure 19 displays the Baseline encoded message in the IRC channel. The presence of the steganographically hidden message is concealed to channel observers.

<capt123> Generated with
 Figure 19: Baseline Encoding IRC Channel Traffic

Estimating the theoretical throughput of the Baseline algorithm is done via the average case capacity per line and overall encoding ratio. The average capacity per line is the amount of covert bits that encodes into a single line. Assuming a default line length of 80 characters and no cover traffic maximizes the covert bits per line. The size of the secret file in bits is

$$(Y - 1) \times 3 \tag{5}$$

where *Y* is the number of 0x0Fh characters in the line. Taking the number of whitespace segments multiplied by 3 gives the encoded bits in the line.

In the best case scenario, a secret file composed of all 0's encoded into a line 80 characters in width would have 80 0x0Fh characters. Using (5), the size of this secret file is 237 bits. In the worse case, a secret file composed of all 1's would have 11 0x0Fh characters, resulting in a secret file 30 bits in size encoded into an 80 character line. The average case capacity per line encodes 134 bits of data into 80 bytes, a 1:4.77 ratio.

The overall Baseline encoding ratio is the size of the secret file in bytes to the number of bytes in the Baseline encoded file. Assuming a 3-byte secret file, which is equivalent to 24 bits, the Baseline encoder produces 8 whitespace sequences using the algorithm in Table 5. On average, the number of whitespaces per sequence is 4, producing a total of 32 whitespaces in the encoded file. Every group of whitespaces must be separated by 0x0Fh characters, requiring 9 total (for 8 whitespace sequences). Therefore the total size of the Baseline encoded file is $32 + 9 = 41$ bytes resulting in a 3:41 ratio. This average encoding ratio is consistent regardless of the size of the secret file.

4.2 *Throughput Encoding Algorithm*

4.2.1 Experiment #2: Maximize IRC Throughput

This section describes the experimental methodology and results used to construct the Throughput encoding algorithm. In this scenario, a fictional adversary intends to exfiltrate a great amount of information over IRC with little regard to detectability beyond hiding their covert messages from channel observers. Maximizing capacity requires sending the most information per packet as well as per time-slice. Non-viewable characters found during experiment #1 are used to achieve this high throughput objective.

The theoretical capacity is determined and compared to the traffic generated by the Throughput encoder.

Maximizing capacity per packet necessitates encoding every bit of the secret file as efficiently as possible as well as utilizing entire packet contents for each IRC transmission. Section 4.2.2 analyzes these requirements, presents the Throughput encoding algorithm, and finds the theoretical limit for capacity per packet.

Maximizing capacity per second requires transmitting messages just under the IRC server throttling limits. Public IRC servers, by default, throttle all incoming traffic per user or per channel to prevent spammers or bots from disrupting channels or causing denial of service to legitimate users. The throttle limits are examined by transmitting packets to the IRC server at regular intervals and observing whether the server relays them to channel clients or not. This experiment and results are discussed in Sections 4.3.3 and 4.3.4, respectively.

4.2.2 Maximizing Capacity per Packet

Section 4.1 finds the ASCII characters 0x03h, 0x0Fh, 0x16h, and 0x1Fh are non-viewable in the top two IRC client applications. Using these characters, the Throughput encoding scheme is presented in Table 6. Each byte of the secret file is deconstructed into four two-bit elements. There are four possible two-bit combinations of data which are each mapped to the specific IRC hidden characters. In this scheme, 1 byte of secret data encodes to 4 covert bytes, a 1:4 ratio. This is a significant improvement over the Baseline encoding algorithm which, on average, requires 41 bytes to encode three bytes of secret data.

Table 6: Throughput Encoding Scheme

Character Bits	Byte Encoding
00	0x03
01	0x0F
10	0x16
11	0x1F

Further throughput gains are achieved through a pilot study determining the maximum packet size of a single IRC message. Messages of increasing size are transmitted to the test IRC servers where an observer client determines if the entire message was relayed correctly. This study finds that the maximum payload size for *response* packets is 512 bytes, including IRC protocol header and trailer information. Response packets are the IRC packet messages forwarded from the IRC server. When messages are transmitted larger than this size, the IRC server only relays a part of the message - up to the maximum of 512 bytes (including heading and trailer) to clients in the channel. This limitation is the system bottleneck in maximizing capacity per packet.

Accounting for the header and trailer bytes which include the end-line characters (0x0D, 0x0A), user name, channel name, and PRIVMSG command, gives the available space for payload and covert storage. In the pilot study, 72 bytes of information are reserved for the IRC protocol. Each Throughput-encoded IRC message uses all available space after the cover traffic to store the non-viewable steganographic characters, up to the maximum IRC packet size. Figure 20 shows Wireshark captures of the Throughput encoder in operation. The image on the left shows the message as it is transmitted to the IRC server with command “PRIVMSG #test1234q”, the cover traffic “Password”, and

the embedded covert data. The right image displays the IRC server response packet containing the IRC protocol header and trailer information, as well as the intact cover message and the steganographically embedded data. Note the payload length from the server is maximized at 512 bytes while the transmitted packet is only 461 bytes in length. This is because more bytes are needed for packet overhead in IRC server response packets, which include the user and nickname of the sender, than in packets sent to the server.

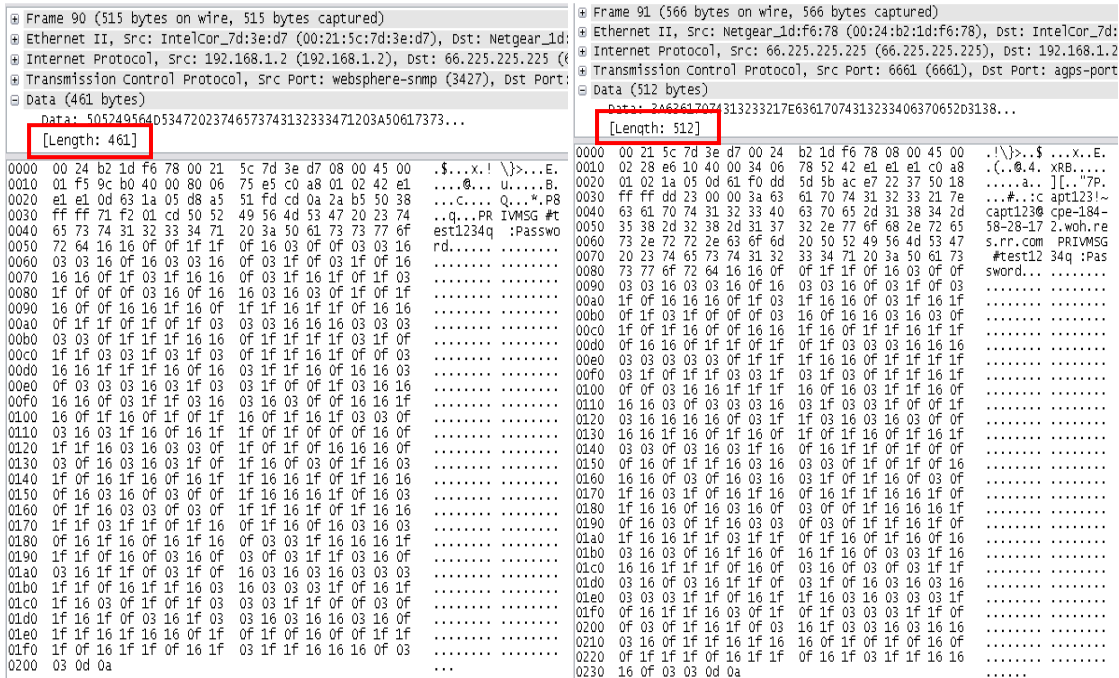


Figure 20: Throughput Transmitted Packet (left), Received Packet (right)

Figure 21 shows the Throughput encoded message in the IRC channel. The cover text “Password” is viewable while the presence of the steganographically hidden message is concealed to channel observers.

|| <capt123> Password

Figure 21: Throughput Encoded Channel Traffic

The presence of the covert traffic should be non-viewable to channel users, therefore, messages should contain at least one byte of viewable cover traffic to maintain the façade of a legitimate user. Assuming 72 bytes of header/trailer data, this allows for a maximum of 440 bytes for the combined cover traffic and covert storage out of a 512 byte message. Each covert byte represents two bits of the secret file with the Throughput encoding scheme resulting in a top capacity limit of 880 bits per packet. This upper capacity limit can be achieved only if there is no cover traffic in the messages and all available packet space is used for covert encoding. However, the lack of viewable traffic in a long series of messages could alert channel observers and therefore violates the covert objective. Similar objections apply to using only one byte of cover traffic per message, a long series of messages with just a single viewable character could also alert channel observers. Experimentally, cover traffic with an average line length of 34 bytes per message is used for all covert channel methods to better compare the techniques. This cover traffic length is the average length of messages in a large legitimate traffic sample discussed in more detail in Section 4.3.1.

4.2.3 Maximizing Capacity per Second

This section discusses a pilot study used to determine the maximum number of packets that can be transmitted during a given time-slice. The goal is to determine the minimum inter-packet delay separating groups of messages before server throttling intercedes. The study is performed using the IRC clients in Dayton, Ohio and an IRC server (irc.servercentral.net) located in Chicago, Illinois. Using a public IRC server, as opposed to a controlled test server in a simulated environment, introduces realism issues

that can degrade performance and affect the overall results of the system. However, realistic environmental conditions will more accurately determine performance characteristics and real-world limitations. These experiments measure the actual time messages are received versus their expected arrival time. The sender's client uses a high precision software-based microsecond timer to transmit messages at set intervals. On the receiver's client, another high precision software-based microsecond timer records when the sender's messages are received.

The accuracy of these software-based timers is dependent on the processing load of the computer running the software. Due to limited OS scheduling resources and timing constraints with other processes on the computer, the timer mechanism may not always be granted highest priority and therefore can produce slight deviations. This issue could be resolved using a Real-Time Operating System (RTOS) and specialized hardware; however this task is out of the scope of this research and is left as future work. Instead, this test determines the *best estimate* for timing reliability while adhering to more realistic operational conditions.

The experiment consists of the sender transmitting 500 messages at 0.5, 1.0, and 2.0 second intervals to a test channel. In preliminary testing, sub-second intervals cause server throttling, therefore this test has transmission intervals on the order of seconds. The receiving client waits for messages from the sender and records the time of receipt for every message. The time difference is calculated by the receiver using the recorded elapsed time between two packets (IPD) subtracted by the expected elapsed time. Each test is repeated three times.

4.2.4 Results and Analysis of Server Throttling Experiments

This section presents the results of the server throttling pilot study and its conclusions. Table 7 shows the results of a one-variable t-test performed on each of the experimental configurations of the study. The table shows the number of messages sent over all trials, the mean time IPD since the last received message, standard deviation, standard error of the mean, and a 95% confidence interval for the mean in seconds.

Table 7: Timing Reliability for Three Time Interval Configurations

Configuration	N (500 Messages * 3 Trials)	Mean IPD	Standard Deviation	Standard Error of the Mean	Confidence Interval (95%)
0.5 Second Interval	1500	1.067	2.426	0.092	(0.888, 1.248)
1.0 Second Interval	1500	1.000	0.028	0.001	(0.999, 1.002)
2.0 Second Interval	1500	2.000	0.028	0.001	(1.999, 2.002)

Looking at the table, the timing reliability for the 0.5 second interval configuration is the least reliable case with a mean much greater than the expected interval and a standard deviation of 2.426 seconds. Examining the plotted data for this configuration shows the reason for the discrepancy in Figure 22. The results from all three 0.5 second trials as the receiver accepts the server IRC messages. The Y-axis displays the variance, in seconds, between each message's actual time of receipt subtracted by the expected time of 0.5 seconds. The x-axis displays the message number. While the majority of the messages have a very small variance from the expected 0.5 second IPD, approximately every 20 messages results in the IRC server throttling the traffic to the channel. While the server throttles messages, no messages are serviced

resulting in less than 250 messages received out of the total 500 expected. These results indicate that 0.5 seconds is an unreliable timing interval due to frequent packet loss from server throttling.

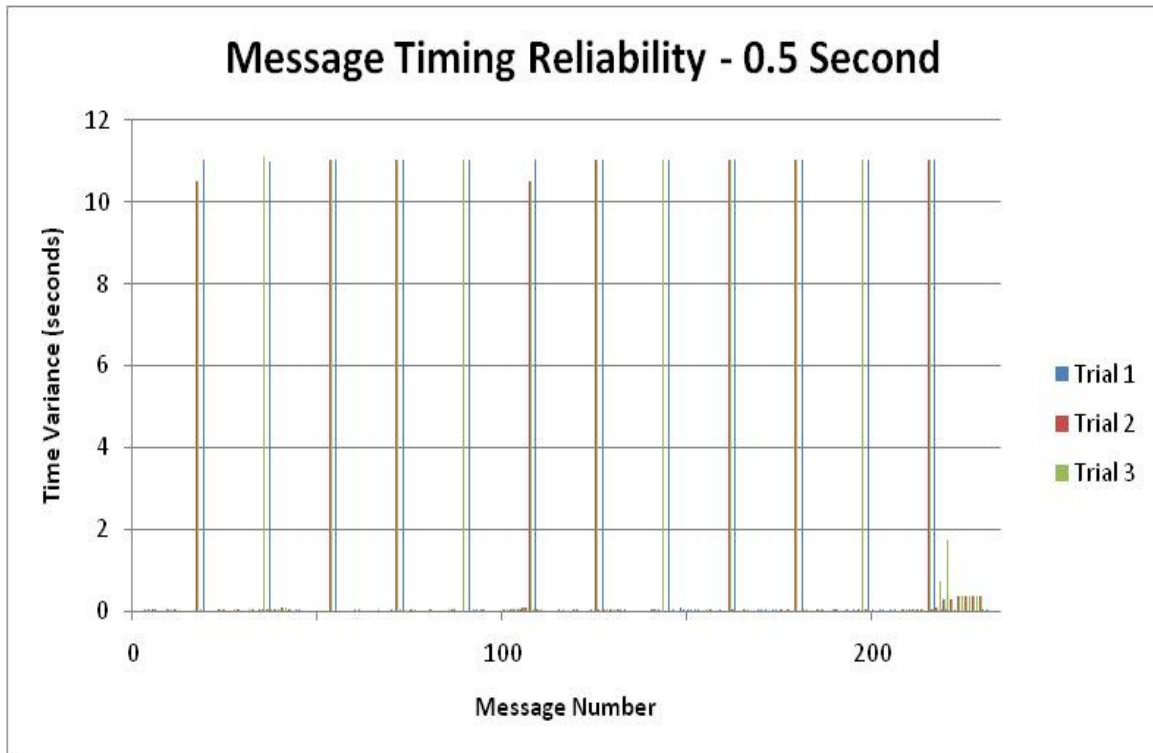


Figure 22: Timing Variance for the 0.5 Second Trials

The results from the 1.0 and 2.0 second interval configurations produce much more consistent results as indicated by their 95% confidence intervals. The results from the 1.0 second interval plot in Figure 23 show its packet timing reliability. Most messages are received in the 0 to ± 0.16 seconds time variance from expected. There are two outlier data points at the end of trial 3 that take approximately 0.6 seconds longer to receive than expected. This delay is likely caused by packet loss and retransmissions since the IRC messages use the TCP protocol to guarantee delivery.

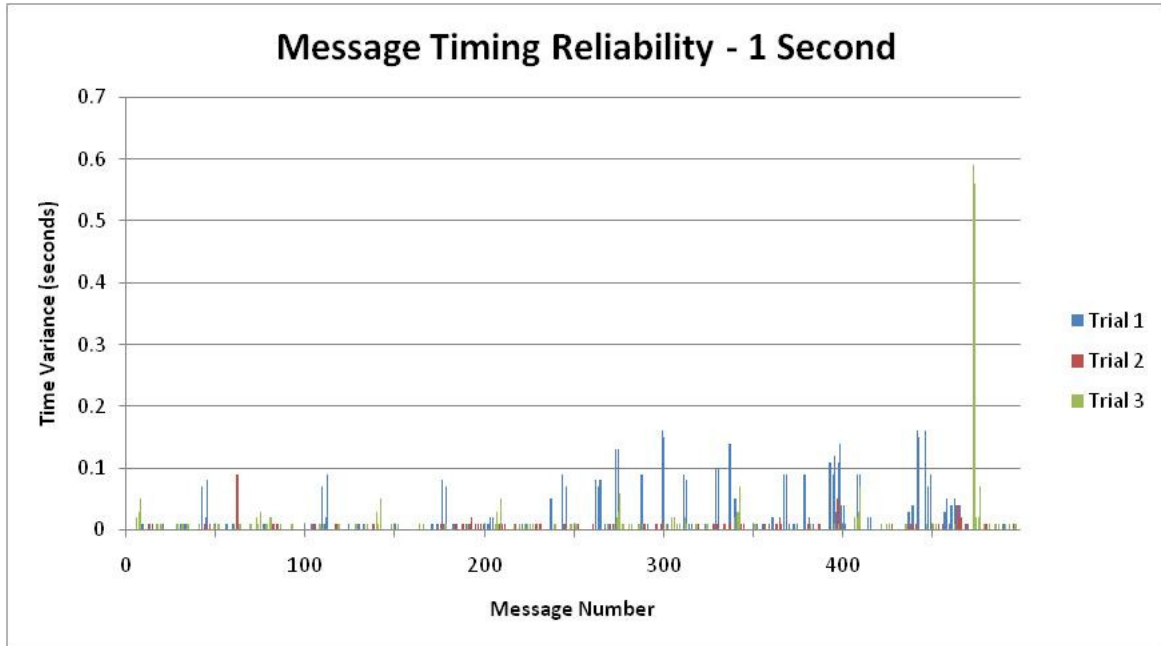


Figure 23: Timing Variance for the 1.0 Second Trials

Based on the results of this experiment, the Throughput encoding algorithm can utilize inter-packet delays as small as 1.0 second without being inhibited by server throttling to achieve maximum transmissions per second. The 2.0 second interval provided little additional robustness, so the 1.0 second interval is chosen.

The Throughput encoder allows an adversary to send fewer messages to transmit large amounts of secret information quickly at the expense of increasing the size of the messages. Since the transmission rate and packet sizes are maximized, this case is, by far, the noisiest. Every packet transmitted may provide a clue of the covert transfer to network administrators or forensic specialists.

4.3 *Stealth Encoding Algorithm*

4.3.1 *Experiment #3: Identify Legitimate IRC Traffic Distribution*

This section describes an experiment which identifies a legitimate IRC traffic distribution for use in the Stealth encoding algorithm. The intuition behind the Stealth algorithm is to classify observed IPD IRC traffic distributions and then mimic the shape of that distribution to decrease detectability. To determine the distribution of legitimate IRC traffic, chat logs of an observed IRC channel, #teamliquid, were captured from November 1 – 3, 2010. The key measurement in this study is IPD. In total, the chat logs contain 6,734 human messages. The probability density function (PDF) of the observed traffic is shown in Figure 24. By visual inspection, the histogram appears to follow a lognormal curve where there is a large spike in the smaller IPD frequencies which then asymptotically approaches zero as the IPD values increase.

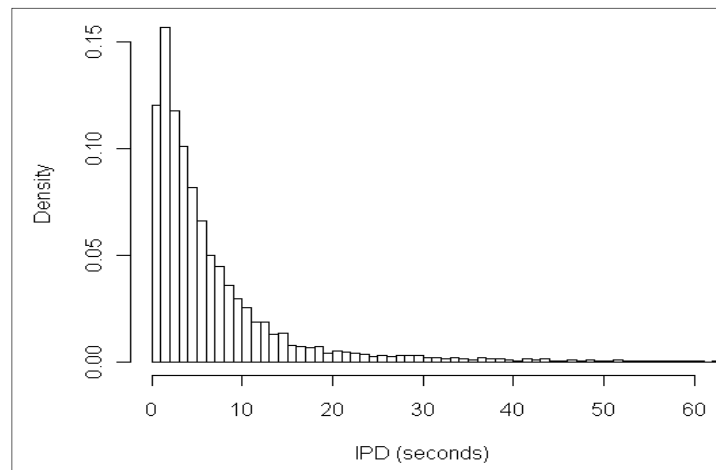


Figure 24: PDF of Observed IRC Traffic

An analysis of the inter-packet delays fits the observed distribution to various other distributions. The fitting process uses maximum likelihood estimation (MLE) to

determine the parameters for each model. The model with the smallest root mean squared error (RMSE), which measures the difference between the model and the estimated distribution, is chosen as the traffic model. The model selection process is automated and performed offline using the R statistical program [Rpr10].

The observed IRC traffic pattern fits closest to a lognormal distribution which has a PDF of

$$f_x(x; \mu, \sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x - \mu)^2}{2\sigma^2}}, x > 0. \quad (6)$$

The mean, μ , is 1.370 and standard deviation, σ , is 1.122. By definition, the log of a lognormally distributed function is normally distributed. To verify this assumption, the normal Quantile-Quantile (Q-Q) plot of the log transformed data versus normal theoretical quantiles is plotted. As shown in Figure 25, the plot follows a strong normal distribution between the -2 and +2 theoretical quantiles. Data points outside of this range are not expected to be strongly modeled in the covert timing channel framework; however, as this is the best fitting model, it is hypothesized to be strong enough to avoid shape-based detection.

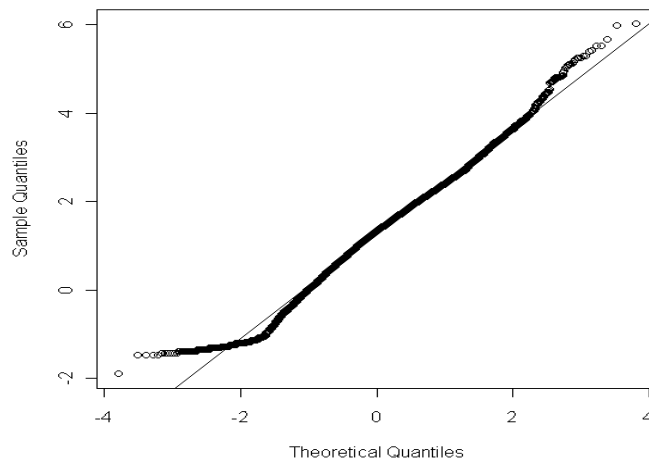


Figure 25: Q-Q Plot of Log Observed Traffic versus Normal Distribution

Figure 26 shows the best-fitting lognormal distribution in red over the observed histogram of the legitimate traffic. By visual inspection from Figures 25 and 26, there is strong confidence that legitimate traffic distribution follows the best fit curve.

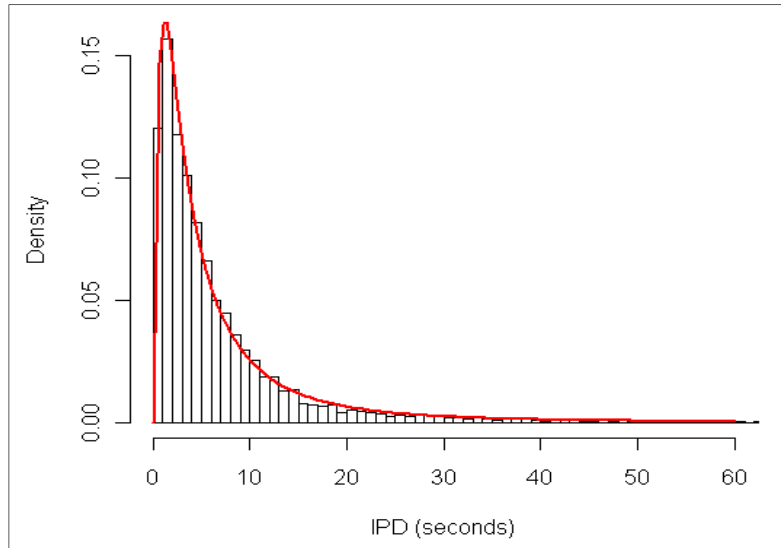


Figure 26: PDF of Observed IRC Traffic versus Best Fit Curve (red)

4.3.2 *Stealth Encoding Algorithm Details*

To evade shape and regularity detection measures, the Stealth encoding algorithm should emulate the legitimate traffic distribution. This traffic has a log-normal distribution with mean $\mu = 1.370$ and standard deviation $\sigma = 1.122$. The GNU Scientific Library (GSL) C function, `gsl_ran_gaussian` [GNU10], generates Gaussian values that match the *log* of the observed traffic's pdf. The choice to model the Gaussian distribution instead of the Log-normal one is arbitrary since the exponential of the Gaussian distribution gives its Log-normal, and there is a high confidence in this property of the GSL C functions based on preliminary testing and its 15 year heritage.

The Stealth encoding pseudocode is shown in Algorithm 1. This algorithm creates a covert timing channel that models the pdf of the observed network traffic. A while loop continues as long as there is an unread bit, b , within the covert file. Next, a randomly generated data point from the Gaussian distribution is calculated, x . If the unread bit, b , is 0 and the value of x is less than or equal to the mean, then an IRC packet is transmitted after e^x seconds. Otherwise, the IRC packet is sent after $e^{(2\mu-x)}$ seconds. On the other hand, if the unread bit is 1 and the value of x is less than or equal to the mean, then an IRC packet is sent after $e^{(2\mu-x)}$ seconds. If the value of x is greater than the mean, then the packet is transmitted after e^x seconds. Finally, the encoded bit is marked as read, B , to prevent it from being processed again. Essentially, this algorithm places all 0 bit delays to the left of the mean and all 1 bits to the right.

Algorithm 1 Stealth Encoder:

```

while  $\exists b \in \text{covert file}$ 
  Generate random Gaussian value,  $x$ 
  if  $b = 0$ 
    if  $x \leq \mu$ 
      Transmit after  $e^x$  seconds
    else if  $x > \mu$ 
      Transmit after  $e^{(2\mu-x)}$  seconds
  else if  $b = 1$ 
    if  $x \leq \mu$ 
      Transmit after  $e^{(2\mu-x)}$  seconds
    else if  $x > \mu$ 
      Transmit after  $e^x$  seconds
  Mark  $b$  as  $B$ 
end while

```

The equation $2\mu - x$ calculates the value on the opposite side of the mean from x with equal distance from the mean. Figure 27 graphically displays this property with a sample Gaussian distribution, $\mu = 0$, $\sigma = 1$. Assuming an x value of 1 is chosen, ($2 *$

0) $-1 = -1$ which has the same distance from the mean as the starting x value. Given a roughly equal proportion of 1 and 0 bits in a given file, this algorithm ensures that the encoded values are always centered about the mean.

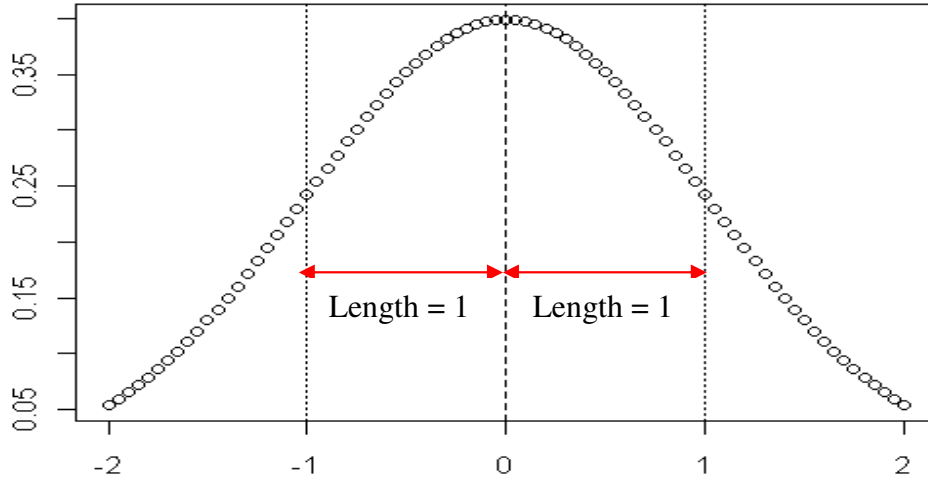


Figure 27: Sample Gaussian Distribution

The Stealth decoder pseudocode, shown in Algorithm 2, decodes packets based on when they arrive in relation to the mean. First, the decoding client waits for a message from the sending client, $client_s$. Once this first message is received, future packets from $client_s$ are analyzed until they disconnect from the server. Each packet's timing is recorded relative to the previous packet to determine the observed IPD. When the IPD is less than or equal to the mean, μ , that it is interpreted as an encoded 0 bit, otherwise the IPD is interpreted as an encoded 1 bit.

Algorithm 2 Stealth Decoder:

```
wait until  $\exists$  message from  $client_s$ 

while  $client_s$  is connected to server
    Observe  $client_s$  message IPDs

    if IPD  $\leq \mu$ 
        Record encoded bit = 0
    else if IPD  $> \mu$ 
        Record encoded bit = 1
end while
```

4.3.3 Reed-Solomon Forward Error Correction

In preliminary testing, the WAN-US and WAN-EU scenarios perform with a BER of 0.3% and 3.1%, respectively. In the WAN-EU scenario, long latencies and bursts of packets from the server cause frequent bit flips. Bit errors from packet drops and duplicates are not observed due to the reliable packet delivery from TCP/IP below IRC in the Open Systems Interconnection (OSI) stack. Reliably transmitting a 3 KB file with this BER is unacceptable because even one bit error is enough to prevent successfully decoding the file. Therefore, forward error correction is examined to improve reliability.

The Reed-Solomon forward error correction algorithm is used to improve the reliability of the Stealth encoder algorithm. The Reed-Solomon algorithm has several properties that make it attractive for this application. For one, it is used to detect and correct multiple bursts of errors in applications ranging from deep-space communications to CD reading [PoH92]. The Reed-Solomon algorithm treats multiple bit errors within a single byte as one error. Second, the amount of overhead used by the parity is configurable. Every byte of parity can detect one byte of errors and every two bytes of parity can correct one byte [WsB94]. This allows the amount of parity needed to correct

the observed errors to be adjusted without significantly affecting throughput. Third, errors in the parity do not negatively affect the decoding of the message, as long as the number of parity bytes is greater than two times the total byte errors in the *codeword*. A codeword is the bytes of data plus the number of bytes of parity. Reed-Solomon codes are represented by the codeword-message pair (N, K) where there are $(N-K)$ parity bytes.

The Reed-Solomon encoding is performed after RC4 encryption, just prior to the transmission phase. Fixing each codeword size at 64 bytes for every 56 bytes of the message, a $(64, 56)$ encoding, provides 8 bytes for parity. Accounting for the 8 extra bytes needed for parity for every 56 message bytes increases the duration for file transfers and reduces the throughput by 12.5% versus traffic without the forward error correction. Eight parity bytes corrects up to 4 byte errors or 6.25% of byte errors per codeword. In the preliminary Stealth encoding trials without Reed-Solomon, the worst case BER observed is 3.1%, therefore the $(64, 56)$ encoding can correct errors over 2 times the worst BER observed in the trials.

4.4 Summary

This chapter presents the design and experiments used to construct and test the VANISH system. The first section finds ASCII characters which are not viewable over IRC using the top two leading IRC chat clients for Windows and Linux. The Baseline algorithm is discussed which steganographically hides information over IRC. Then, experiments to determine the highest throughput per message and per second out of each IRC for the Throughput encoding algorithm are presented. Finally, Section 4.4 identifies

the distribution parameters that most closely resembles the legitimate IRC traffic pattern and describes the details of the Stealth encoding algorithm.

V. Results and Analysis

This chapter presents and analyzes experimental results from the three encoding configurations under test. First, an overall analysis of the inter-packet delay (IPD) characteristics is discussed in Section 5.1. Next, the results of the throughput and bit error rate (BER) performance metrics are presented in Section 5.2. The results of the detectability performance of each configuration are given in Section 5.3. Finally, the chapter is concluded in Section 5.4.

5.1 Results and Analysis of IPD Characteristics

This section presents the results of the IPD statistical summary for the legitimate traffic as well as the summary of the Baseline, Throughput, and Stealth encoded traffic experiments for comparison.

Table 8 shows the results of a one-variable t-test on legitimate IRC traffic samples collected between 8am – 5pm, November 1 – 3, 2010 for comparison with the experimental results. As found in Chapter 4, the legitimate traffic most closely follows a log-normal distribution. However, data analysis using t-test requires normally distributed data; therefore the legitimate network traffic is logarithmically transformed for normalization. To report the results in seconds, an exponential transformation is applied: $Y = e^{\ln x}$. Estimating the values this way produces the geometric mean and geometric standard deviation of the legitimate sample. The number of packets, mean IPD, standard deviation, and 95% confidence interval are reported in seconds.

Table 8: Legitimate Traffic Statistical Summary

	N (Packets)	Mean IPD	Standard Deviation	Confidence Interval (95%)
Legitimate Traffic	6748	3.972	3.074	(3.865, 4.081)

In each experiment, the sender covertly transmits one of three randomly generated 3 kilobyte files to the IRC server during a single trial. The set of experiments consists of three trials per configuration for each network scenario using an IRC server located in Chicago IL, referred to as WAN-US, and in Amsterdam NL, referred to as WAN-EU. The server forwards all messages to the IRC channel where the receiver records the messages and packet timing properties. Identical cover traffic is used for each configuration's trials averaging 34 bytes per packet. The content of the cover traffic is arbitrary, but in actual use it should be germane to the channel to avoid arousing suspicion. A one-variable t-test is performed on the results shown in Tables 9 and 10. The tables include number of packets, mean IPD, standard deviation, and the 95% confidence interval for each configuration in seconds.

Table 9: WAN-US Inter-Packet Delay Statistical Summary

Component Under Test	Secret 3 KB	N (Packets)	Mean IPD	Standard Deviation	Confidence Interval (95%)
Baseline	File 1	1289	1.001	0.037	(0.999, 1.002)
	File 2	1289	1.001	0.015	(0.999, 1.001)
	File 3	1290	1.001	0.031	(1.000, 1.002)
Throughput	File 1	30	1.006	0.033	(1.002, 1.008)
	File 2	30	1.005	0.033	(1.002, 1.009)
	File 3	30	1.006	0.032	(1.001, 1.007)
Stealth	File 1	27648	3.943	3.175	(3.637, 4.275)
	File 2	27648	3.968	3.059	(3.753, 4.199)
	File 3	27648	4.053	3.040	(3.786, 4.339)

Table 10: WAN-EU Inter-Packet Delay Statistical Summary

Component Under Test	Secret File 3 KB	N (Packets)	Mean IPD	Standard Deviation	Confidence Interval (95%)
Baseline	File 1	1289	1.001	0.181	(0.991, 1.011)
	File 2	1289	1.001	0.166	(0.992, 1.010)
	File 3	1290	1.001	0.208	(0.990, 1.012)
Throughput	File 1	30	1.005	0.182	(0.987, 1.022)
	File 2	30	1.005	0.166	(0.989, 1.021)
	File 3	30	1.004	0.161	(0.989, 1.019)
Stealth	File 1	27648	4.028	3.174	(3.714, 4.370)
	File 2	27648	3.971	3.046	(3.754, 4.202)
	File 3	27648	3.975	3.039	(3.671, 4.305)

The amount of time required to send 27,648 packets with the Stealth configuration and its slow average transmission time is excessive. Therefore, the Stealth analysis uses 100-byte secret files, resulting in 1,024 packets per trial. In preliminary tests, changing the size of the secret file does not significantly change the IPD statistical properties of the transfer.

Figure 28 shows the 95% confidence intervals of legitimate traffic compared to the Stealth encoder results for both server configurations. The following qualitative observations are noted:

- The Baseline configuration encodes secret files #1 and 2 into 1289 packets and File #3 into 1290 packets. The discrepancy in packets is because the Baseline algorithm does not treat all secret bytes equally: the more one bits in the secret increases the amount of whitespace needed to encode them. Given the 3 kilobyte secret files, the covert byte per packet ratio for the Baseline algorithm is approximately 1:0.429: every byte of secret data encodes into approximately 0.429 of a packet.

- The Throughput configuration encodes all three 3 kilobyte files into 30 packets. Unlike the Baseline algorithm, the Throughput method encodes all bytes equally in the message regardless of the number of zero or one bits. The Throughput algorithm also increases the available space per packet for the covert message using non-viewable ASCII characters. It performs with an encoding ratio of 1:0.01 secret bytes per packet: every byte of secret data encodes into one-hundredth of a packet.
- The Stealth configuration encodes all three 3 kilobyte files into 27,648 packets with forward error correction (FEC) enabled, 24,000 without. Since every bit equates to one packet transfer in this method, there are a total of 3,648 extra packets needed for overhead from the Reed-Solomon parity. The benefits and drawbacks from adding FEC are discussed in more detail in Section 6.2 where throughput and reliability are analyzed. Overall, the Stealth configuration performs with an encoding ratio of 1:9.216 secret bytes per packet: every byte of secret data results into approximately 9.216 packets.
- In both server scenarios, regardless of the encoding configuration, the mean IPD difference from transferring the files is minimal: 1.41% in the worst case during the WAN-EU Stealth experiments. Thus, the contents of the secret file are independent from the rate at which the covert transfer takes place. The results agree with the design for each covert method, where the contents of the secret file are treated as a black box enabling them to encode any file type and contents.
- The mean IPD for the Baseline and Throughput configurations are approximately equal in both server scenarios, within 0.4%. However, the standard deviations in

the WAN-EU case are 5 to 11 times greater than with the WAN-US sever. This same increase in standard deviation is not present in the Stealth trials; therefore it is likely that this increase is due to the Baseline and Throughput configuration's attempt to transmit packets at the constant rate of one per second. It is important to note that as the latency and delay increases in the WAN-EU scenario, the variability of the one second transmission intervals is amplified.

- All of the Stealth configuration trials produce traffic whose mean IPD is within the 95% CI of the legitimate traffic sample. This data alone is not sufficient to say that the distributions are similar, which is answered in the detectability metrics in Section 6.3. However, this figure does point to the similarity of the means. The deviations from the mean can be attributed to several factors: the accuracy of the distribution number generator, the presence of lost and retransmitted packets incurring extra delays, the workload of the IRC server at the time of the experiments (higher workload would incur extra delay in routing packets to the channel), and network latencies between the sender to IRC server and IRC server to receiver.

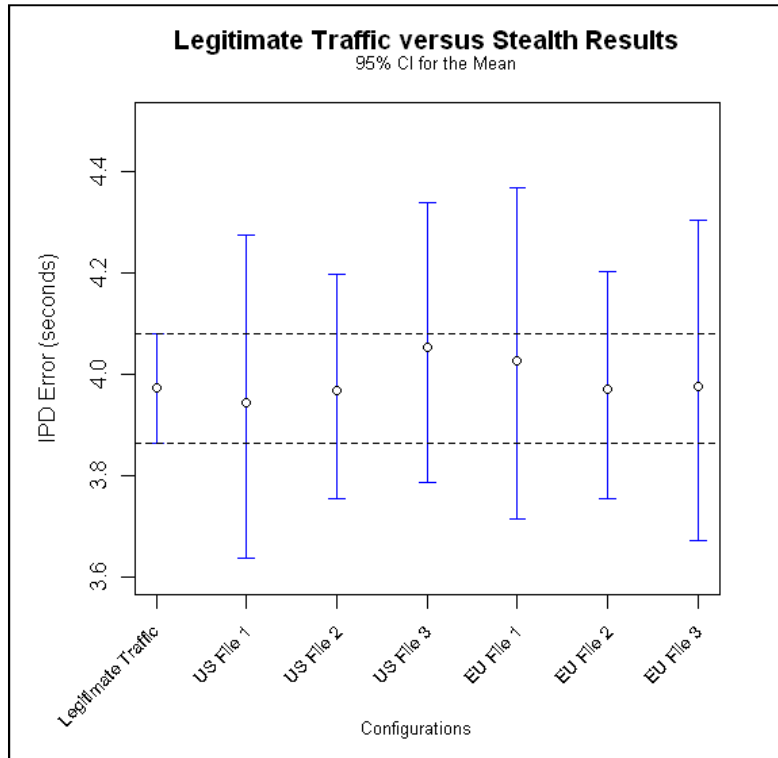


Figure 28: Legitimate Traffic versus Stealth Results

5.2 Throughput and BER Results

Two metrics for bandwidth are collected for each encoding method: capacity per packet (CPP) and capacity per second (CPS). Capacity per packet is derived by dividing the size of the secret file in bits by the number of packets used to transmit the secret. Capacity per second takes the number of covert information in bits, divided by the time required to complete the transmission.

The BER is determined after the transmission by comparing the original bits of the encoded secret file to the received bits of the file after forward error correction. If a received bit does not match its transmitted bit, either through omission, duplication, or bit-flip, it is counted as an error. The BER is found by dividing the number of errors by

the total number of packets sent. The following observations are found based on the throughput and BER results in Tables 11 and 12:

Table 11: Throughput and BER Results – WAN-US

Test Server Scenario		WAN-US		
System Metric	Secret 3 KB	Baseline	Throughput	Stealth
Capacity per Packet	File 1	18.619	803.733	0.999
	File 2	18.619	800.000	0.999
	File 3	18.604	800.000	0.999
Capacity per Second	File 1	18.583	802.502	0.137
	File 2	18.589	798.801	0.134
	File 3	18.567	798.961	0.134
Bit Error Rate	File 1	0%	0%	0%
	File 2	0%	0%	0%
	File 3	0%	0%	0%

Table 12: Throughput and BER Results - WAN-EU

Test Server Scenario		WAN-EU		
System Metric	Secret 3 KB	Baseline	Throughput	Stealth
Capacity per Packet (bits)	File 1	18.619	803.733	0.999
	File 2	18.619	800.000	0.999
	File 3	18.604	800.001	0.999
Capacity per Second (bits)	File 1	18.587	802.983	0.126
	File 2	18.587	804.829	0.133
	File 3	18.572	799.094	0.132
Bit Error Rate	File 1	0%	0%	0%
	File 2	0%	0%	0%*
	File 3	0%	0%	0%

* Three uncorrected bit errors after FEC are found in one trial but are too small to report with 3 significant digits after the decimal; the aggregate BER for File 2 is approximately 36.168×10^{-8} .

- The throughput from each configuration is similar despite using different randomly generated files; within 5% in the worst case. This is by design, since

each encoding algorithm is designed to be content independent. Each secret file is deconstructed into its binary form prior to encoding; the transmission speeds are therefore not dependent on the secret file contents.

- The throughput of the File #3 experiment in the Baseline configuration is 0.08% less than the other two file transfers. The reduction in CPS is due to one more packet being needed to encode the file and therefore another second of total transmission time. The slight reduction in CPP is due to the Baseline method encoding the contents of File #3 less efficiently than the other two files. The Baseline algorithm uses more message space to encode consecutive one bits instead of zero bits.
- The Throughput configuration has the greatest CPP and CPS of the three methods. It boasts approximately 43 times more CPS and CPP than the Baseline encoding method and dwarfs the Stealth method's average CPP and CPS by 802 and 6155 times, respectively. The cover traffic, on average, uses approximately 34 bytes per message. This reduced the available capacity per packet for the Baseline and Throughput methods.
- As discussed in Chapter 4, the Throughput method has a maximum capacity per packet of 440 bytes when no cover traffic is used. These experiments use cover traffic with an average of 34 bytes per message, leaving 406 bytes available for the steganographic channel. The Throughput algorithm encodes two bits of the secret file into a single non-viewable ASCII character (one byte); therefore the maximum capacity with this cover traffic is 812 bits per packet. The results show that this method approaches the limit but never maximizes capacity per packet.

One reason for this is because the last message of the covert transfer is only partially filled with the remaining covert data, which reduces the overall capacity results.

- Unlike the Baseline and Throughput configurations, the Stealth method does not rely on the available space in an IRC packet to covertly communicate; therefore its throughput is not affected by the size of the cover traffic per message but on maintaining the shape of its distribution.
- The Baseline and Throughput configurations perform with 100% accuracy during all trials regardless of the test server. The reason they have such high reliability is because their covert methods utilize the packet contents of IRC messages, which rely on TCP/IP for data integrity and guaranteed packet delivery.
- The reliability of the Stealth encoded traffic is significantly improved with Reed-Solomon FEC. In preliminary Stealth trials without FEC, the worst case BER observed is 3.1% in the WAN-EU scenario. With Reed-Solomon error correction in a (64, 56) configuration, only one trial, WAN-EU File #2, produced errors that were not correctable. There are a number of possible causes for the increased number of errors in this test: lost and retransmitted packets between the European server and US-based clients, increased network latency between hops in the chain, or increased IRC server workload can cause network delays increasing packet arrival times and lead to inaccuracies in the decoding process.
- Improving the Stealth method's reliability with FEC reduces its overall throughput because of the extra bytes needed for parity. The Stealth method's

results show that the increased overhead is worth the tradeoff because it achieves very high reliability.

5.3 *Detection Results*

Detectability is determined based on the shape and variance of the experimental methods. To examine the shape of a distribution, the Kolmogorov-Smirnov test is performed, which is a non-parametric goodness-of-fit test. The regularity test is used to examine the variance of the traffic pattern. The results from the Stealth encoding are of primary interest in this section because it was designed to evade shape-based detection, unlike the Baseline and Throughput configurations.

The legitimate traffic sample is compared with traffic captures from each encoding experiment. The Kolmogorov-Smirnov test is run 100 times against the results from each trial and with randomly generated distributions with the mean and standard deviation of the best fit distribution from the legitimate traffic. Each test produces a one-sided p-value, the measure of evidence against the null hypothesis that the two compared traffic distributions are equal. The smaller the p-value, the greater the certainty that the distributions are different. High p-values suggest that the data is *consistent* with the null hypothesis but not necessarily that the two distributions are equal. In general, interpreting the meaning of a particular p-value is challenging because the p-value confidence levels can vary from situation to situation. This thesis uses the suggested p-value weighting scale in Figure 29, where a p-value from 0 to 0.01 shows convincing evidence of a difference, 0.01 to 0.05 is moderately convincing, 0.05 to 0.10 is suggestive

but inconclusive, and values greater than 0.10 show less evidence of a difference but does not rule out the possibility.

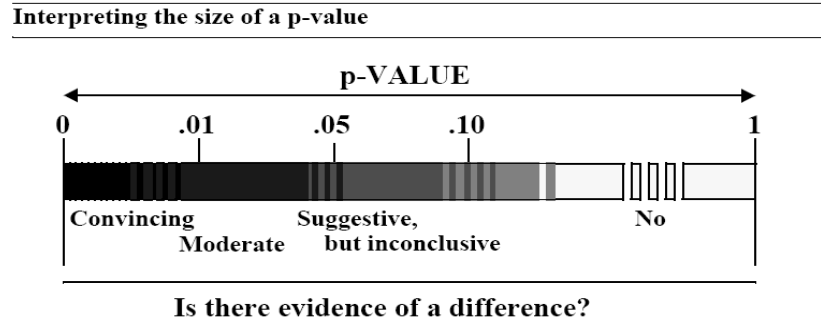


Figure 29: p-Value Weighting Scale [RaS02]

The regularity test assesses the variance of the data set’s IPD. A higher regularity score above an established threshold, depending on the traffic of interest, indicates that the traffic pattern has more variability over time, while lower scores indicate less variability. For the regularity test, a sample is separated into sets of windows of size $w=25$ (larger window sizes were also tested and had little effect on the results). Then the standard deviation of each set is computed. The regularity score is the standard deviation of the pair wise differences between each σ_i and σ_j for all sets $i < j$ or

$$regularity = STDEV \left(\frac{|\sigma_i - \sigma_j|}{\sigma_i} \right), i < j, \forall i, j \quad (7)$$

where σ_i is the standard deviation of the i^{th} window. The greater the difference in standard deviation between pairs of windows (greater variance), the greater the traffic’s overall regularity score.

Establishing a baseline for the Kolmogorov-Smirnov (KS-test) and regularity tests is important to determine whether the covert methods would remain undetectable in the legitimate data stream or not. The KS-test p-value and regularity score are computed for

the legitimate traffic, shown in Table 13. A Kolmogorov-Smirnov test p-value of 0.203 indicates that the legitimate traffic distribution is *consistent* with its own best-fitting traffic distribution. This particular test is not necessary because the best fit distribution is already found to be in agreement with the legitimate traffic in Chapter 4, but is included for completeness. A regularity score of 3.746 serves as the baseline for the experimental configurations. To effectively remain undetectable to the regularity test, the covert channels should produce a regularity score similar to the legitimate traffic; especially high or low values could provide evidence of covert activity.

Table 13: Legitimate Traffic Detection Results

	Legitimate Traffic
KS-Test p-value, mean 100	0.203
Regularity Score, $w = 25$	3.746

Figure 30 shows a graphical display of the standard deviations for each window of the observed legitimate traffic. Most of the windows in the sample have standard deviations smaller than 40; however, a small percentage of the windows are significantly larger.

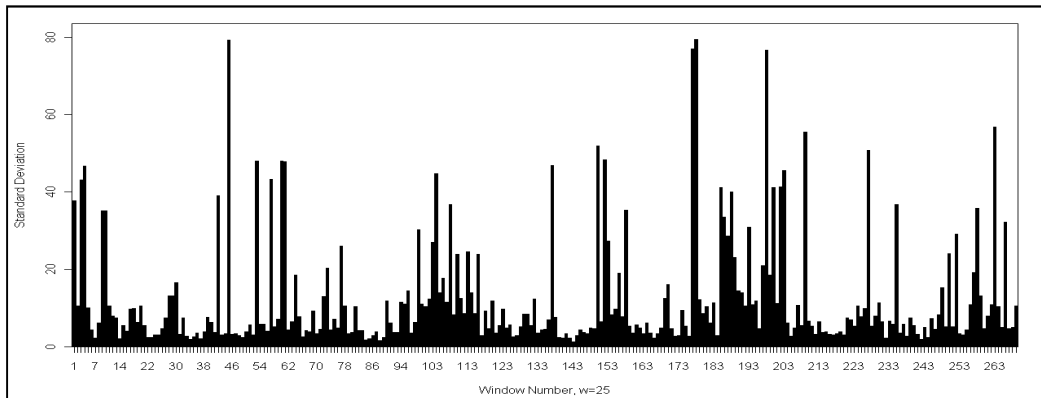


Figure 30: Legitimate Traffic Variance

The detectability results from the experiments are shown in Table 14 and 15. To get an accurate regularity score for the Throughput case, 40 kilobyte files are used since the 3 kilobyte files only produce 30 packets (less than two full windows at $w=25$). Figure 31 shows how the generated Stealth traffic (red bars) approximates the best fit curve of the legitimate traffic (black line) in the WAN-US and WAN-EU scenario. The following observations and interpretations are derived from the results in Tables 14 and 15 and Figures 31, 32, and 33:

Table 14: Detection Results - WAN-US

Detection Metric	Secret File	Baseline	Throughput	Stealth
KS-Test p-value, mean 100	File 1	< 2.2e-16	< 2.2e-16	0.579
	File 2	< 2.2e-16	< 2.2e-16	0.581
	File 3	< 2.2e-16	< 2.2e-16	0.509
Regularity Score, $w = 25$	File 1	3.413	2.908	0.825
	File 2	2.692	5.125	0.696
	File 3	7.141	1.318	0.836

Table 15: Detection Results - WAN-EU

Detection Metric	Secret File	Baseline	Throughput	Stealth
KS-Test p-value, mean 100	File 1	< 2.2e-16	< 2.2e-16	0.577
	File 2	< 2.2e-16	< 2.2e-16	0.600
	File 3	< 2.2e-16	< 2.2e-16	0.619
Regularity Score, $w = 25$	File 1	9.344	19.036	0.764
	File 2	14.695	14.574	1.015
	File 3	20.207	5.260	0.651

- The Kolmogorov-Smirnov test p-values for the Baseline and Throughput configurations show that there is convincing evidence that the data sets are different. The difference between the traffic patterns produced by these encoding

configurations is significantly different from the legitimate traffic as to convincingly rule out chance. Based on these results, the Throughput and Baseline encoded traffic are easily detected with the KS-test. The reason why the Baseline and Throughput configurations fail to produce a traffic distribution similar to the legitimate traffic is because they are designed to transmit packets as fast as possible over the channel, making no attempts to prevent shape-based detection.

- There is significant similarity between the p-values for each configuration despite using different randomly generated files and varying server configurations for the Stealth traffic. This is by design; the shape of the covert channel traffic is designed to be independent from the contents of the secret file. Further, the additional latency in the WAN-EU scenario does not perturb the traffic distribution enough that it is sufficiently different from the legitimate traffic, as seen in Figure 31.
- Through visual inspection of the probability density of the log-transformed Stealth encoded traffic in Figure 31, the Stealth traffic follows the shape of the legitimate traffic pattern closely in both server configurations which is corroborated by their high KS-test p-values. The increased latency in the WAN-EU scenario has little effect on the shape.
- The Stealth encoded traffic produces traffic consistent with the best fit distribution of the legitimate traffic with Kolmogorov-Smirnov p-values ranging from 0.509 to 0.619. This is because the Stealth algorithm uses the best-fitting parameters of the legitimate sample to maintain the shape of the distribution.

Based off of the p-value weighting scale in Figure 29, the Stealth method is not detectable using the Kolmogorov-Smirnov method.

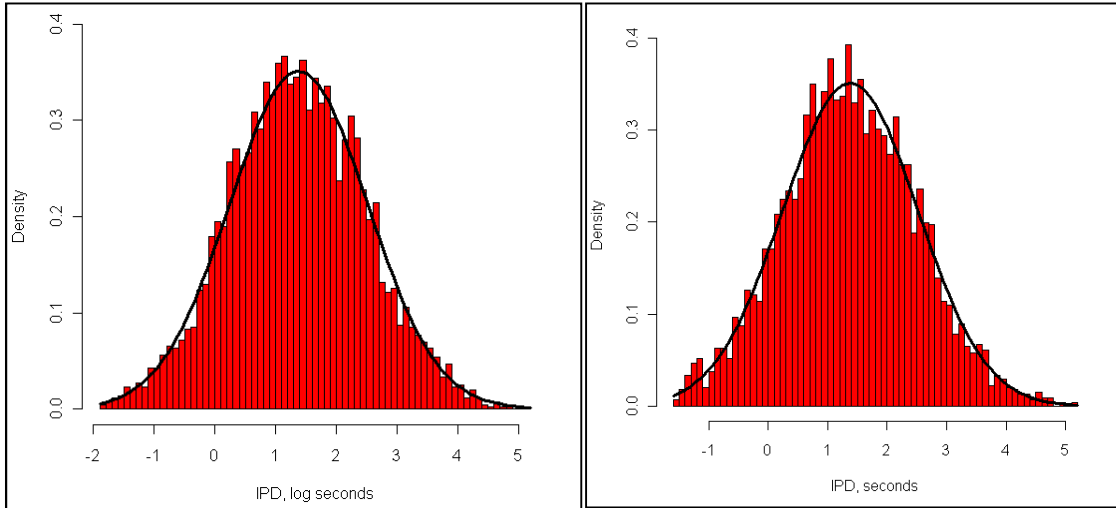


Figure 31: Stealth Probability Density (red bars) versus Legitimate Traffic Density (black line) - Log Transformed, WAN-US (left) and WAN-EU (right)

- The regularity scores for the legitimate traffic are, on average, 4.69 times greater than the regularity of the Stealth traffic in the WAN-US and WAN-EU scenarios, as a whole. The significant difference in regularity scores shows that Stealth encoded traffic can be detected with this metric and therefore fails the regularity test. The primary reason why the Stealth encoded traffic fails is because its algorithm is only designed to defeat shape-based, as opposed to regularity-based, detection. As such, the random number generator that produces IPD values matching the shape of the legitimate traffic does not account for the regularity of the original traffic sample. This issue can be overcome by calculating the regularity as IPDs are computed and adjusting the delay such that it emulates the legitimate traffic's regularity score. This task is left for future research.

- The regularity results from the WAN-EU test scenarios are, on average, 1.03 times greater than their WAN-US counterparts in the Stealth configuration, 3.34 times greater with the Baseline, and 4.15 times greater with the Throughput configuration. The wide range of regularity score differences suggests that high latency scenarios with fixed packet IPDs have a greater effect on this metric than with varying IPDs, as is the case with the Stealth configuration. These variances produce noticeable differences in the traffic regularity, increasing detectability if the regularity scores become significantly higher or lower than the legitimate traffic.
- The higher regularity scores for the Baseline method show it has more variance than the Stealth method but less than the legitimate traffic. Examining Figures 30, 32 and 33, however, reveals that the greatest standard deviation for the Baseline method is approximately 0.04 seconds; much smaller than the legitimate and Stealth traffic samples. Given that the Baseline scenario's design to transmit packets at static one second intervals, the high regularity score does not agree with the results in [CBS04]. In that research, a significantly lower regularity score than a legitimate sample shows that the sample has less variance, indicating a possible covert channel with a statically set traffic pattern. These results show that the regularity score is not as closely tied to the variance of the windows as originally thought. The Baseline method produces closer regularity scores to the legitimate traffic than the dynamic time intervals of the Stealth method, despite the sample having much less variance.

- One factor considered is that regularity may not be a strong secondary metric for detecting these covert channels. However, regularity is tied to the standard deviation of each sample window. The average standard deviation for all windows of the legitimate traffic gives a value of 12.358 versus 10.119 for the Stealth, 0.073 for the Baseline, and 0.075 for the Throughput configurations. This shows that the average standard deviation of the Stealth encoded traffic is actually similar to the legitimate traffic to within 20%, whereas the Baseline and Throughput configurations are significantly different from the legitimate traffic. Given the similarities between the average standard deviations of the legitimate and Stealth traffic, the difference in regularity scores is counterintuitive. Further analysis and examination of the regularity formula against various sets of data is left for future research.

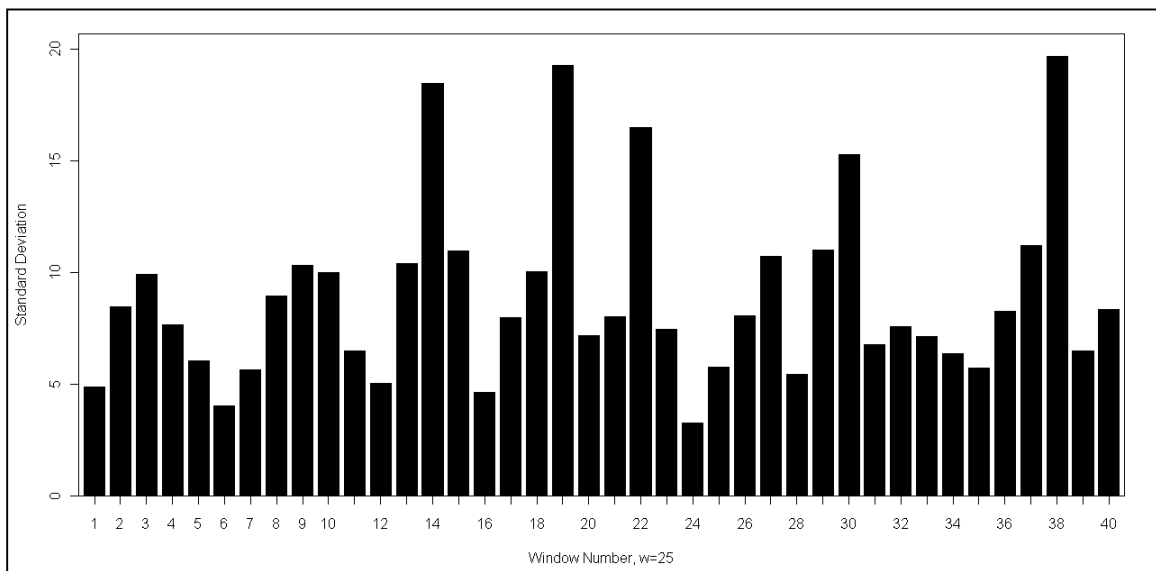


Figure 32: Stealth WAN-US Traffic Regularity

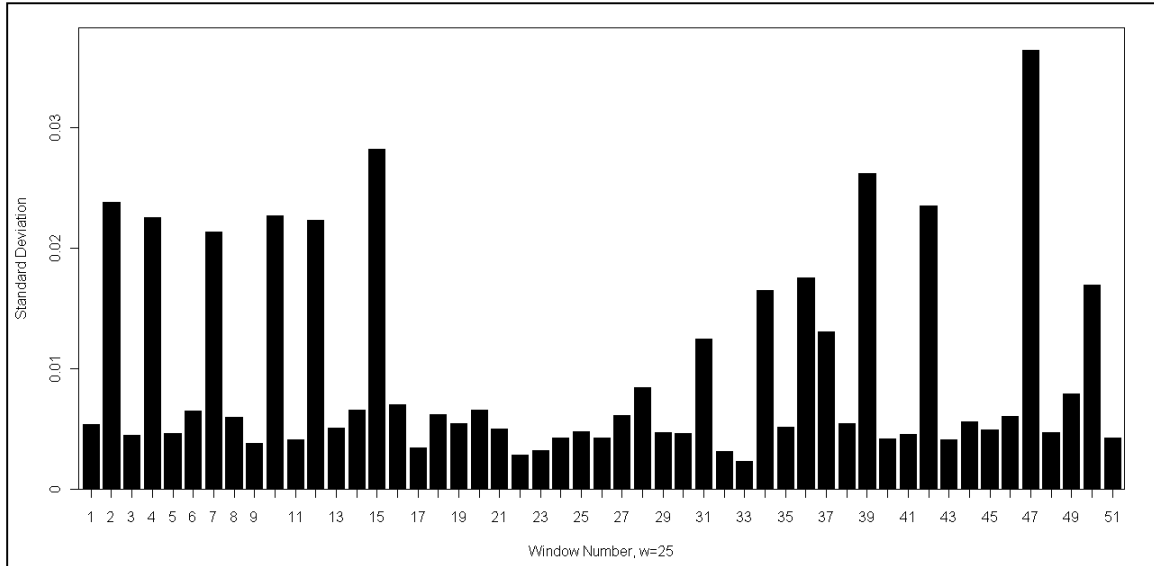


Figure 33: Baseline WAN-US Traffic Regularity

5.4 Summary

This chapter presents and analyzes the data collected from each of the three covert hiding method's experiments. An overall analysis and discussion of the results is presented. A statistical analysis of the performance metrics for each test is performed. Finally, the detectability of the methods is evaluated using the Kolmogorov-Smirnov and regularity tests. The results show that Throughput configuration exfiltrates data 43 times faster than the Baseline and 6155 times faster than the Stealth configuration. When maintaining a low detectability profile is the priority, the Stealth encoder is able to avoid shape-based detection, unlike the Throughput and Baseline configurations, but is vulnerable to regularity detection.

VI. Conclusions

This chapter presents the overall conclusions of this research. Section 6.1 provides several recommendations for extending this research. The significance of the research is discussed in Section 6.2. Section 6.3 summarizes the findings from the experimental results and determines if the research objectives have been met.

6.1 Recommendations for Future Research

There are many possible directions for future research. Expanding the system to other unexplored text-based protocols such as America Instant Messenger (AIM), Microsoft Network (MSN) Messenger, Trillian, the Facebook chat client, and even cell phone text messages may show promise as steganographic or covert timing channel avenues.

Compression is another avenue which could drastically improve covert transmission throughput. In a preliminary study, files compressed with the PAQ8PX routine are significantly smaller than other compression schemes. Compression was not used to more accurately measure encoding throughput. Compression would skew the encoding performance since different files of the same size compress to different sizes.

A hybrid covert channel which utilizes IPD and steganography techniques to achieve higher throughput while retaining its detection resistance to shape-based countermeasures could be developed.

Improving the detection resistance of the system is another valuable area for future research. The Stealth algorithm could model a traffic sample's regularity by keeping track of the generated traffic's regularity as it is running and adjusting IPD

values accordingly. The results obtained herein indicate a possible problem with the published formula for calculating regularity [CBS04]. Therefore, an analysis of this formula is in order to determine if it is a good measure of regularity. Other possible detectability improvements may include e-similarity, entropy, conditional entropy and corrected conditional entropy measures.

Although many hours of chat data were collected and analyzed, a larger corpus with a wider variety of channels and channel participants would be beneficial for modeling various sized chat rooms. To this end, further contributions of chat profiles are encouraged to progress this research. Additionally, a system which analyzes live traffic to produce an equivalent best fitting distribution would be ideal because less preparation would be needed to create undetectable covert timing channel traffic.

Lastly, future research should investigate covert channel encoding methods that do not rely on the entropy of the secret file, i.e., files that have significantly more one or zero bits than the other should not have a negative effect on the detection resistance of the method. This is one limitation of the covert timing channel herein. However, the count of '1' and '0' bits in the randomly generated test files happened to be approximately equal (within 300 bits in a 3 kilobyte file), and was not significant enough to affect the detectability.

6.2 *Research Summary*

This research develops the Variable Advanced Network IRC Stealth Handler (VANISH) system which covertly exfiltrates information over IRC. The implementation of the covert channel methods is split into two main parts. The first consists of

techniques that use non-viewable ASCII characters in IRC messages to covertly transmit data, specifically the Baseline and Throughput methods. The Baseline method uses whitespaces to steganographically encode data into IRC messages. The Throughput method attempts to maximize capacity per packet and capacity per second using other non-viewable characters in IRC messages. The second part uses methods for minimizing detectability of the channel traffic such that it would evade the shape-based detection tests, referred to as the Stealth method. The shape of a legitimate traffic sample is analyzed and used as the basis for the shaping algorithm. In this way, all IRC covert channel messages from the sending client are used to maintain the shape of the legitimate sample.

Channel throughput, reliability, and detectability are evaluated with each covert channel using public IRC servers located domestically and abroad. Empirically, the results show that the Throughput method exfiltrates covert data at nearly 800 bits per second (bps) compared to only 18 bps with the Baseline method and 0.13 bps for the Stealth method. This data rate can be further improved by eliminating or reducing the amount of cover traffic, up to a maximum of 880 bits per second.

Reed-Solomon forward error correction (FEC) is implemented on the Stealth encoder using a (64, 56) configuration to improve its reliability. Prior to implementing FEC, bit errors approached 3.1% in WAN-EU tests. Since a single bit error in the received message can cause data corruption, rendering the message unreadable, FEC is used. The results show nearly 100% data transfer reliability with minimal additional overhead.

The detectability results show that the Stealth encoded traffic successfully evades shape-based detection based on the Kolmogorov-Smirnov test. However, the variance of the Stealth traffic is detectable with the regularity test because it produces traffic with a significantly lower regularity score than the legitimate traffic. Given the wide ranging regularity scores in high-latency environments, regularity may not be the metric of choice for detecting covert channels.

6.3 *Significance of Research*

The primary motivation for this research is that no tools exist for transmitting or detecting covert or steganographic channels over IRC streams. However, botnet masters are expected to begin using steganography in 2011 [Lew10]. Therefore, it is important that steganographic methods and detection techniques over IRC and peer-to-peer protocols be analyzed. Given the covert channel methods proposed, certain countermeasures can be instituted to detect or prevent these methods. Creating deep-packet inspection rules on a network gateway scanning for specific non-viewable ASCII characters in IRC messages can be one effective detection technique. Additionally, approaches maximizing stealth require a significant amount of time and messages to transmit covert secrets. Therefore, time or message restrictions enforced on a per-client basis would further slow down the covert channel's throughput. Finally, and most importantly, organizations using IRC should restrict connections to only trusted or internal servers to prevent unauthorized entry and examine server logs. Further protection measures include password protected channels and Secure Socket Layer (SSL) encrypted traffic.

This research presents methods for embedding covert messages into an IRC stream using several different approaches. While IRC messages are generally small in nature, they present a clear and present danger to businesses and academic institutions which allow IRC traffic through their communications network. The novelty of VANISH stems from its dual purpose design enabling it to achieve either high capacity or high detection resistance over IRC depending on the needs of the user.

Bibliography

- [Ada08] P. Adams, "Conversation Thread Extraction and Topic Detection in Text-based Chat," Master's thesis, Naval Post Graduate School, 2008.
- [BGC05] V. Berk, A. Giani, and G. Cybenko, "Covert channel detection using process query systems," in *Proceedings of FLOCON 2005*, September 2005.
- [BGM⁺96] W. Bender, D. Gruhl, N. Morimoto, and A. Lu, "Techniques for Data Hiding," in *IBM Systems Journal*, Vol. 35, Issues 3 & 4, pp. 313-336, 1996.
- [BLM⁺95] J. Brassil, S. Low, N. F. Maxemchuk, and L. O'Garman, "Electronic Marking and Identification Techniques to Discourage Document Copying," in *IEEE Journal on Selected Areas in Communications*, Vol. 13, pp. 1495-1504, 1995.
- [BLM⁺99] J. Brassil, S. Low, N. F. Maxemchuk, and L. O'Garman, "Copyright Protection for the Electronic Distribution of Text Documents," in *Proceedings of IEEE*, Vol. 87, 1999.
- [Bol04] I. A. Bolshakov, "A Method of Linguistic Steganography Based on Collocationally-verified Synonymy," in *Information Hiding: 6th International Workshop*, Vol. 3200 of Lecture Notes in Computer Science, pp. 180-191, 2004.
- [Bro94] R. Browne. "An entropy conservation law for testing the completeness of covert channel analysis," in *CCS '94: Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pp. 270-281, 1994.
- [Cac05] C. Cachin, "Digital Steganography," *IBM Research*, Zurich Research Laboratory, February 17, 2005.
- [CaK01] A. E. Cha, J. Krim, "Terrorists' Online Methods Elusive U.S. Agencies Seek Experts' Help in Tracing Encrypted Messages," in *The Washington Post*, pg. A14, September 19, 2001.
- [CBS04] S. Cabuk, C. E. Brodley, and C. Shields, "IP covert timing channels: design and detection," in *CCS '04: Proceedings of the 11th ACM conference on Computer and Communications Security*. New York, NY, USA: ACM, 2004, pp. 178-187.
- [CBS09] S. Cabuk, C. E. Brodley, and C. Shields, "IP Covert Channel Detection," in

ACM Transactions Information System Security, vol. 12, no. 4, pp. 1-29, 2009.

- [Cha97] M. Chapman, "Hiding the Hidden: A Software System for Concealing Ciphertext as Innocuous Text," Master's thesis, University of Wisconsin, Milwaukee, 1997.
- [Con03] M. Conway, "Code Wars: Steganography, Signals Intelligence, and Terrorism," in *Knowledge, Technology and Policy*, Vol. 16, No. 2, pp. 45-62, 2003.
- [Cve04] N. Cvejic, "Algorithms for Audio Watermarking and Steganography," Master's thesis, University of Oulu, Oulu, 2004.
- [Eat10] K. Eaton, "Deep Inside Alleged Russian Spies' Tech and Techniques," Retrieved 25 July, 2010 from <http://www.fastcompany.com/1665066/russian-spy-ring-code-james-bond-sleeper-agents-steganography>.
- [Enc07] "Encryption Basics Using RC4," Retrieved 28 May, 2010 from <http://www.security-freak.net/encryption/encryption-rc4.html>.
- [Eov06] B. Eovito, "Assessment of Joint Chat Requirements from Current Usage Patterns," Master's thesis, Naval Postgraduate School, Monterey, California, 2006.
- [FFP⁺03] G. Fisk, M. Fisk, C. Papadopoulos, and J. Neil, "Eliminating steganography in internet traffic with active wardens," in *IH '02: Revised Papers from the 5th International Workshop on Information Hiding*, London, UK: Springer-Verlag, pp. 18-35, 2003.
- [FPK07] J. Fridrich, T. Pevny, and J. Kodovsky, "Statistically Undetectable JPEG Steganography: Dead Ends, Challenges, and Opportunities," in *Proceedings of the 9th workshop on Multimedia & Security*, Dallas, TX, 2007.
- [Gel10] A. Gelhausenby, "IRC Network Summary," Retrieved 20 May, 2010 from <http://irc.netsplit.de/networks/summary.php>
- [GiW07] S. Gianvecchio and H. Wang, "Detecting covert timing channels: an entropy-based approach," in *CCS '07: Proceedings of the 14th ACM conference on Computer and Communications Security*. New York, NY, USA: ACM, pp. 307-316, 2007.
- [GNU10] GNU Scientific Library, Retrieved 10 September, 2010 from

<http://www.gnu.org/software/gsl/>.

- [GWW⁺08] S. Gianvecchio, H. Wang, D. Wijesekera, and S. Jajodia, "Model-based covert timing channels: Automated modeling and evasion," in *RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*. Berlin, Heidelberg: Springer-Verlag, pp. 211-230, 2008.
- [Hu91] W. M. Hu, "Reducing timing channels with fuzzy time," in *Proceedings of 1991 IEEE Computer Society Symposium*, pp. 8-20, May 1991.
- [IRC91] "IRC Logs During 1991 USSR coup d'état," Retrieved 23 May, 2010 from <http://www.ibiblio.org/pub/academic/communications/logs/report-ussr-gorbachev>.
- [IRC94] "IRC Logs During 1994 Desert Storm," Retrieved 24 May, 2010 from <http://www.ibiblio.org/pub/academic/communications/logs/Gulf-War/>.
- [Jam03] D. Jamieson, "RC4 Encryption Algorithm," Retrieved 27 May, 2010 from http://www.vocal.com/data_sheets/RC4.pdf.
- [KaM93] M. H. Kang and I. S. Moskowitz, "A pump for rapid, reliable, secure communication," in *CCS '93: Proceedings of the 1st ACM conference on Computer and Communications Security*, New York, NY, pp. 119-129, 1993.
- [KaP00] S. Katzenbeisser, F. A. P. Petitcolas, "Information Hiding Techniques for Steganography and Digital Watermarking," in *Electronic Data Processing Audit, Control and Security*, Vol. 28, pp. 1 – 20, 2000.
- [KCC⁺07] D. Kleiman, K. Cardwell, T. Clinton, M. Cross, M. Gregg, and J. Varsalone, "The Official CHFI Study Guide (Exam 312-49) for Computer Hacking Forensic Investigators," Published by: Syngress Publishing, Inc., 2007.
- [Kem83] R. Kemmerer, "Shared resource matrix methodology: an approach to identifying storage and timing channels," in *ACM Transactions Computer Systems*, vol. 1, no. 3, pp. 256-277, 1983.
- [Kem02] R. Kemmerer, "A practical approach to identifying storage and timing channels: Twenty Years Later," in *Proceedings Computer Security Applications Conference*, pp. 109-118, 2002.
- [KMC05] M. H. Kang, I. S. Moskowitz, and S. Chincheck, "The pump: A decade of covert fun," in *ACSAC '05: Proceedings of the 21st Annual Computer*

Security Applications Conference. Washington, DC, pp. 352-360, 2005.

- [Kwa06] M. Kwan, "Snow," Retrieved 20 May, 2010 from <http://www.darkside.com.au/snow>.
- [Lew10] D. Lewis, "2011 Trends: Botnets Evolve with Steganography," Retrieved 8 December, 2010 from <http://www.symantec.com/connect/blogs/2011-trends-botnets-evolve-steganography>.
- [MoH06] A. Mousa, A. Hamad, "Evaluation of the RC4 Algorithm for Data Encryption," in *International Journal of Computer Science & Applications*, Vol. 3, No. 2, pp. 44-56, June 2006.
- [NFN⁺04] H. Noda, T. Furuta, M. Niimi, and E. Kawaguchi, "Video Steganography Based on Bit-Plane Decomposition of Wavelet-transformed Video," in *Security, Steganography, and Watermarking of Multimedia Contents VI*, San Jose, CA, 2004.
- [Oik05] J. Oikarinen, "IRC History by Jarkko Oikarinen," Retrieved 25 May, 2010 from http://www.irc.org/history_docs/jarkko.html.
- [PAD08] L. Y. Por, T. F. Ang, and B. Delina, "WhiteSteg: A New Scheme in Information Hiding Using Text Steganography," in *WSEAS Transactions on Computers*, Vol. 7, pp. 735-745, 2008.
- [PoH92] A. Poli, L. Huguët, "Error Correcting Codes: Theory and Applications," Prentice Hall International, pp. 433, 444, 1992.
- [RAD⁺10] RuyDuck, Apatrix, Dracus, and Jolo, "IRC clients primarily for the Unix shell," Retrieved 27 Jun, 2010 from <http://www.irchelp.org/irchelp/ircii/>.
- [RaS02] Fred L. Ramsey, Daniel W. Schafer, "The Statistical Sleuth : A course in methods of data analysis, 2nd Edition," Publisher: Brooks/Cole Cengage Learning, ISBN-10:0534386709, pg 47, 2002.
- [RaS10] K. F. Rafat, M. Sher, "Survey Report – State of the Art in Digital Steganography Focusing on ASCII Text Documents," in *International Journal of Computer Science and Information Security*, Vol. 7, No. 2, pp. 63-72, 2010.
- [Rpr10] R Project for Statistical Computing, Retrieved 10 June, 2010 from <http://www.r-project.org>.
- [RSA10] "What is RC4?" RSA Laboratories, Retrieved 28 May, 2010 from <http://www.rsa.com/rsalabs/node.asp?id=2250>.

- [Sel62] A. D. Selincourt, "The World of Herodotus," Published by: Little Brown and Company, Boston. pp. 217-230, 1962.
- [Sha08] M. Shirali-Shahreza, "Text Steganography by Changing Words Spelling," in *International Conference on Advanced Communication Technology*, 2008.
- [ShS08] M. H. Shirali-Shahreza, M. Shirali-Shahreza, "A New Synonym Text Steganography," in *International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, 2008.
- [Sim84] G. J. Simmons, "The Prisoners' Problem and the Subliminal Channel," in *Advances in Cryptology: Proceedings of Crypto 83*, pp. 51–67, 1984.
- [SMB06] G. Shah, A. Molina, and M. Blaze, "Keyboards and covert channels," in *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*. Berkeley, CA, 2006.
- [Sta05] M. Stamp, "Information Security-Principles and Practice," Wiley Interscience, 2005.
- [Sti08] R. Stillman, "Detecting IP covert timing channels by correlating packet timing with memory content," in *IEEE Southeastcon*, pp. 204-209, April 2008.
- [SWT01] D. X. Song, D. Wagner, and X. Tian, "Timing Analysis of Keystrokes and Timing Attacks on SSH," in *SSYM'01: Proceedings of the 10th conference on USENIX Security Symposium*, Berkeley, CA, pg. 25, 2001.
- [TTA06] U. Topkara, M. Topkara, and M. J. Atallah, "The hiding virtues of ambiguity: Quantifiably resilient watermarking of natural language text through synonym substitutions," in *Proceedings of ACM Multimedia and Security Workshop*, pp. 164-174, 2006.
- [TTA07] M. Topkara, U. Topkara, and M. J. Atallah, "Information Hiding Through Errors: A Confusing Approach," in *Proceedings of SPIE-IS&T Electronic Imaging SPIE*, pp. 6505, 2007.
- [WrW09] Walls R., Wright M., "Liquid: A Detection Resistant Covert Timing Channel Based On IPD Shaping," Technical Report CSE2009-8, May 2009.
- [WsB94] S. B. Wicker, V. K. Bhargava, "Reed-Solomon Codes and their Applications," IEEE Press, ISBN 0-7803-1025-X, 1994.

- [ZLC08] Z. Zhu, G. Lu, and Y. Chen, "Botnet Research Survey," in *Annual IEEE International Computer Software and Applications Conference*, pp. 967-972, 2008.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of the collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY) 24-03-2011		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From - To) Sep 2009 - Mar 2011	
4. TITLE AND SUBTITLE Covert Channels Within IRC			5a. CONTRACT NUMBER		
			5b. GRANT NUMBER		
			5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Wayne C. Henry, Capt, USAF			5d. PROJECT NUMBER		
			5e. TASK NUMBER		
			5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB OH 45433-7765				8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/GCE/ENG/11-04	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Robert Kaufman 318 Information Operations Group/DD 688 th Information Operations Wing (AFSPC) 102 Hall Blvd, Suite 311, San Antonio, TX 78243-7078 (210) 925-4425 Robert.Kaufman@us.af.mil				10. SPONSOR/MONITOR'S ACRONYM(S) 318 th IOG/DD (688 th IOW)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S)	
12. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT The exploration of advanced information hiding techniques is important to understand and defend against illicit data extractions over networks. Many techniques have been developed to covertly transmit data over networks, each differing in their capabilities, methods, and levels of complexity. This research introduces a new class of information hiding techniques for use over Internet Relay Chat (IRC), called the Variable Advanced Network IRC Stealth Handler (VANISH) system. Three methods for concealing information are developed under this framework to suit the needs of an attacker. These methods are referred to as the Throughput, Stealth, and Baseline scenarios. Each is designed for a specific purpose: to maximize channel capacity, minimize shape-based detectability, or provide a baseline for comparison using established techniques applied to IRC. The effectiveness of these scenarios is empirically tested using public IRC servers in Chicago, Illinois and Amsterdam, Netherlands. The Throughput method exfiltrates covert data at nearly 800 bits per second (bps) compared to 18 bps with the Baseline method and 0.13 bps for the Stealth method. The Stealth method uses Reed-Solomon forward error correction to reduce bit errors from 3.1% to nearly 0% with minimal additional overhead. The Stealth method also successfully evades shape-based detection tests but is vulnerable to regularity-based tests.					
15. SUBJECT TERMS Network timing channels, Steganography, Evasion, IRC, Covert channel detection					
16. SECURITY CLASSIFICATION OF:		17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 111	19a. NAME OF RESPONSIBLE PERSON Dr. Barry E. Mullins, ENG	
REPORT U	ABSTRACT U			c. THIS PAGE U	19b. TELEPHONE NUMBER (Include area code) (937) 255-3636 x7979; Barry.Mullins@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18