Air Force Institute of Technology

AFIT Scholar

Theses and Dissertations | Student Graduate Works

3-26-2020

Multi-Channel Security through Data Fragmentation

Micah J. Hayden

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

 Part of the Computer Engineering Commons

Recommended Citation

Hayden, Micah J., "Multi-Channel Security through Data Fragmentation" (2020). *Theses and Dissertations*. 3174.
https://scholar.afit.edu/etd/3174

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 AFIT.ENWL.Repository@us.af.mil.

Multi-Channel Security through Data Fragmentation

THESIS

Micah J. Hayden, Second Lieutenant, USAF

AFIT-ENG-MS-20-M-026

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

The views expressed in this document are those of the author and do not reflect the official policy or position of the United States Air Force, the United States 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-ENG-MS-20-M-026

MULTI-CHANNEL SECURITY THROUGH DATA FRAGMENTATION

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

Micah J. Hayden, B.S.

Second Lieutenant, USAF

March 2020

DISTRIBUTION STATEMENT A
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AFIT-ENG-MS-20-M-026

MULTI-CHANNEL SECURITY THROUGH DATA FRAGMENTATION

THESIS

Micah J. Hayden, B.S.
Second Lieutenant, USAF

Committee Membership:

Scott R. Graham, Ph.D.
Chair

Major Addison J. Betances, Ph.D.
Member

Robert F. Mills, Ph.D.
Member

AFIT-ENG-MS-20-M-026

Abstract

This thesis presents a novel security system developed for a multi-channel communication architecture, which achieves security by distributing the message and its associated message authentication code across the available channels at the bit level, to support systems that require protection from confidentiality and integrity attacks without relying solely on traditional encryption. One contribution of the work is to establish some helpful terminology, present a basic theory for multi-channel communications, describe the services provided by an optimal system, and then implement a proof of concept system to demonstrate the concept's validity. This proof of concept, focused on the splitting and recombination activities, operates by using existing key exchange mechanisms to establish system initialization information, and then splitting the message in fragments across each available channel. Splitting prevents the entirety of a given message from being transmitted across a single channel, and spreads the overall message authentication across the set of channels. This gives the end user the following unique service: the sender and receiver can identify a compromised channel, even in the presence of a sophisticated man in the middle attack wherein the adversary achieves fragment acceptance at the destination by altering the message's error detecting code. Under some conditions, the receiver can recover the original message without retransmission, despite these injected errors. The resulting system may be attractive for critical infrastructure communications systems as a holistic approach to both availability and a defense against integrity attacks. This system would be a natural fit as a cipher suite for a future iteration of the Transport Layer Security protocol targeting support for multi-channel communication systems.

AFIT-ENG-MS-20-M-026

This thesis is dedicated to my wife for her constant encouragement, love, and support.

Acknowledgements

To my research advisor, Dr. Graham, thank you. Your advice, discussion, and feedback helped me grow throughout this thesis process, and I would not be the student I am today without that mentorship. You kept me on track with my research's efforts, while allowing me to make it my own. This would not have been possible without you.

To my committee members, Major Betances and Dr. Mills, thank you for the feedback throughout this thesis process.

To my beautiful, loving wife, thank you. You supported me throughout my time at AFIT - classes, research, and everything in between. You encouraged me on the long days of writing and helped me focus on the bigger picture. I would not be where I am today without you and your love. Thank you.

And finally, I would like thank my Lord and Savior Jesus Christ for the opportunities He has given me, and for giving me the knowledge, perseverance, and wisdom to finish this thesis. To God be the glory.

Micah J. Hayden

Table of Contents

Page

Page

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ACL Average Channel Load

AES Advanced Encryption Standard

CBC Cipher Block Chaining

CIA Confidentiality, Integrity, Availability

CIP Critical Infrastructure Protection

CRC Cyclic Redundancy Code

DHKE Diffie-Hellman Key Exchange

DoS Denial of Service

ECC Error Correcting Code

FCS Frame Check Sequence

HMAC Hash-based Message Authentication Code

IEC International Electrotechnical Commission

IP Internet Protocol

IV Initialization Vector

LFSR Linear Feedback Shift Register

MAC Message Authentication Code

MITM Man in the Middle

MSB Most Significant Bit

MTU Maximum Transmission Unit

NERC North American Electric Reliability Council

PDU Protocol Data Unit

PRNG Pseudo-Random Number Generator

PSMT Perfectly Secure Message Transmission

QoS Quality of Service

RFC Request for Comment

RS Reed-Solomon

RSA Rivest–Shamir–Adleman

SDF Spatial Duplication Factor

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TLS Transport Layer Security

UDP User Datagram Protocol

I. Introduction

1.1 Background and Motivation

Consumers rely increasingly on the communication of private, confidential, and sensitive information over the internet for everyday activities. Accordingly, individuals and organizations rely on security services, such as encryption, to secure those communications. Encryption is the most prevalent service used for these purposes because it provides strong guarantees that an adversary cannot, in a reasonable time, transform the encrypted transmission into a usable format and extract information.

The fundamental strength of the cryptographic process is that decryption is computationally expensive in the absence of the session's key. A block cipher relies on the computational complexity of deducing the key given the algorithm, the plaintext, and a corresponding ciphertext. Common block ciphers may specify different length keys in order to achieve greater security at the expense of more computation. For example, Advanced Encryption Standard (AES) allows three separate key lengths - 128, 192, and 256 bits [1], with the larger sizes providing greater security.

The popular Rivest–Shamir–Adleman (RSA) Algorithm utilizes both a public key and private key to exchange information and generate a shared session key. This algorithm's security relies on the computational challenge of factoring increasingly large integers [2]. These two algorithms combine into a well-known encryption paradigm in which a form of RSA or public key encryption is used to generate a session key, which is then used to encrypt the data using a block cipher such as AES. There is

concern that quantum computing may be able to reduce the time and cost of decryption so much that encryption is rendered useless. Overlooking that threat, however, encryption may still not be suitable for every application. High bandwidth or latency sensitive communications may not tolerate the computational cost and imposed latency. Low power devices may not be able to afford the computation required for encryption. In addition, traditional communications rely on a single channel, representing a single point of failure for availability, as well as a single location of intercept for an adversary.

Related technologies are also used to provide authentication, providing assurance to one communicating entity that the other entity is who they claim to be. In addition to the traditional public key infrastructure approach to authentication, many systems also employ two factor authentication. This system functions by requiring a user to provide two of the following items:

1. Something they know
2. Something they have
3. Something they are

One common approach is the combination of a security password or PIN, which only the user knows, and a code sent to the user, often on a separate channel, such as a text message, which becomes something they have. This reduces the chance of an adversary compromising the system by gaining access to a single piece (channel) of information. Two-factor authentication demonstrates the security provided by using different pieces (channels) of information. This raises the following question: instead of relying on two-factor authentication, what if a system used two+ factor (or channel) communication?

The use of multiple channels implies that an adversary would require the compromise of multiple communication channels to conduct an attack. The multi-channel protocol presented in this research provides security by restricting the amount of information an adversary can access upon interception, as well as providing increased assurance of message integrity by duplicating and fragmenting information across the set of available channels. Previous work by Wolfe demonstrated the use of multi-channel communications for applications in low power devices [3]. In that research, security was achieved by splitting data across two channels, and if adversarial action was detected, limiting or stopping the flow of information on the affected channel. This work will extend the previous effort, as well as providing a new method of fragmenting data across the channels to achieve increased security.

1.2 Research Objectives

The first objective of this thesis is to identify specific security benefits and services a multi-channel, data fragmenting protocol/architecture provides. The transformation from the single channel to a multi-channel domain requires an acceptance of higher overhead – there are simply more connections to maintain and synchronize. Thus, there must be a significant security benefit driving that shift. Without this defined benefit, there would be no motivation for systems to make the switch. The second objective is to identify challenges impeding the creation and use of such a protocol. The shift to multi-channel communications would require existing network protocols to adapt to handle the splitting and assembling of data from multiple data connections, potentially through different mediums. However, a multi-channel protocol should be built on existing, well-researched, and well-trusted protocols. This would enable an efficient shift from single to multi-channel communications. To accomplish this goal, the specific challenges to a multi-channel communication protocol

must be understood.

1.3 Approach

The first step in accomplishing the objectives above is to define terminology for the discussion of multi-channel communications. This is necessary to ensure that all parties communicate precisely about the design decisions and operation of future iterations of the multi-channel protocol. This terminology is first applied to a theoretical multi-channel communication system, after which a proof of concept system will be developed to demonstrate the operation of such a system. This proof of concept system will identify challenges to the creation of a multi-channel communication protocol, demonstrate the security benefits for a user, and finally to discover areas where the end user can make design decisions based on their specific use case. The lessons of this system will be discussed.

1.4 Contributions

The primary contributions of this thesis are to demonstrate the security benefits of a multi-channel, data fragmenting protocol, as well as to identify remaining challenges that must be solved to implement such a system. Additionally, this effort will define a relevant vocabulary for describing a multi-channel system. This work can then provide a theoretical roadmap of the design considerations and trade-offs necessary to develop and field a multi-channel communication system, as well as the services gained through its use. These insights will serve as a foundation and an inspiration for multi-channel communication development in the years to come.

1.5 Organization

This thesis is organized as follows:

Chapter II introduces the relevant background information of the topic, beginning with the relationship between aspects of the Confidentiality, Integrity, Availability (CIA) Triad, several widely-utilized categories of attacks, and the specific security configuration required by an end user. Note that these security configurations are typically reliant on single channels for communication. The chapter presents preliminary work into the developing field of multi-channel communications, including work into a two-channel system, as well as the theoretical implications of an n channel system. Several security mechanisms are considered for application within a multi-channel communication protocol, as well as the potential area for its implementation.

Chapter III presents the theory/terminology required for the discussion of multi-channel communications, and utilizes that vocabulary to describe a theoretical system. This theoretical system allows an investigation into the benefits of a multi-channel system, without regard for specific design decisions or system requirements, and highlights the potential benefits of such a system. This theoretical model is then followed to build a prototype system which can achieve several of the suggested benefits, demonstrating the possibility of creating this system through existing technologies.

Chapter IV describes the operation of the proof of concept model, which was created using the Python scripting language, and represents the first step toward building a fully functional multi-channel system. The model makes several design decisions, such as the Trivium cipher as the splitting mechanism and Message Authentication Codes (MACs) such as the 32-bit Cyclic Redundancy Code (CRC) or Reed-Solomon (RS) Code providing error detection and error correction, respectively. By using a scripting language such as Python, it isolates the operation of the system from all other environmental factors. This chapter concludes by describing the specific test cases which will be utilized to verify and learn from the model's operation.

Chapter V presents the design trade-offs: what drove specific design decisions, and at what cost? It analyzes the trade space between various parameters and the services they provide such as the percentage of incurred overhead, or the ability to defeat adversarial actions. It explores the operation of the proof of concept system, illustrating that it does indeed function as designed. Finally, it analyzes the insights gained through the creation and use of the proof of concept system, despite the rather deterministic nature of the test cases.

Chapter VI concludes the work. It details the contributions to the field such as demonstrating the security benefits of a multi-channel, data fragmenting protocol. It describes the future work that can follow this thesis, specifically what needs to be done to take this prototyped system to a fielded, operational system.

II. Background and Related Work

2.1 Overview

This work follows an initial effort to create a multi-channel communication protocol [3], which proposed multi-channel communications as a viable security alternative for systems that may not use encryption, such as low-power devices. However, for devices which cannot reconcile the additional computational and power costs, a multi-channel system might not actually be better. The introduction of additional transceivers might have an adverse effect on the system's power efficiency, performing worse than a single channel system with encryption. Thus, a holistic approach must be used to determine the appropriate security scheme for a given use case. The trade-off between power, computational complexity, and latency must be considered for future applications and desired services, such as security in a post-quantum world, providing availability guarantees, and providing a warning system for integrity attacks. *Wolfe et al.* describe how the use of multiple channels can thwart both an eavesdropping attack by splitting the data across two channels, and an integrity attack by duplicating the information across two channels. However, they do not mention specific mechanisms to split or duplicate the data, nor do they consider performing both tasks simultaneously. This work will extend that effort by proposing a specific mechanism for splitting the message across available channels, and identifies several tunable characteristics that the user can select based on their specific security needs. This chapter will establish the necessary background information to place this work in the context of ongoing research in network communications and security.

2.2 Security Paradigms

There are three primary components of security generally considered in practice and in the literature - namely the CIA triad. There are also three associated components of attack, or threats which an adversary may attempt to employ. The security needs or requirements of organizations or individual end users of communication services may vary significantly. These specific needs can be evaluated by which aspects of the CIA triad they require, as well as the specific vulnerabilities their communication system faces. The relationship between these needs and the specific security provisions of a given communication system demonstrates the tunable nature of security: a system which might be considered secure for one user, may be entirely unacceptable for another.

2.2.1 CIA Triad

The CIA Triad presents key aspects that a customer/consumer needs to consider, namely

1. Confidentiality – requires that only the sender and intended recipient(s) can correctly decode/decrypt the transmitted information.

2. Integrity – requires that the information received is correct and not modified in any way, shape, or form.

3. Availability – requires that the information arrives within a certain time/latency window. This is typically quantified using traditional Quality of Service (QoS) metrics.

The specific CIA requirements, which may vary from person to person, or operation to operation, dictate the balance of characteristics that a security service might need to provide, and hence the level and type of security required.

2.2.2 Transmission Threats

Just as CIA requirements vary from person to person or system to system, so too are the vulnerabilities to the various attacks. Each of the three following attacks affects the aspects of the CIA triad differently. Using the classic computer security model, Alice and Bob represent the "good" sender and receiver entities, while Eve is the adversary, as used in [4] and [5]. The figures for these three transmission threats are from [3]

1. Interception

   An interception, or eavesdropping, attack refers to the act of passively accessing communications between two parties [4]. This violates the confidentiality of the transmission. These attacks can be challenging to identify because the recipient will still receive the communications and may have no way in which to discover the interception.

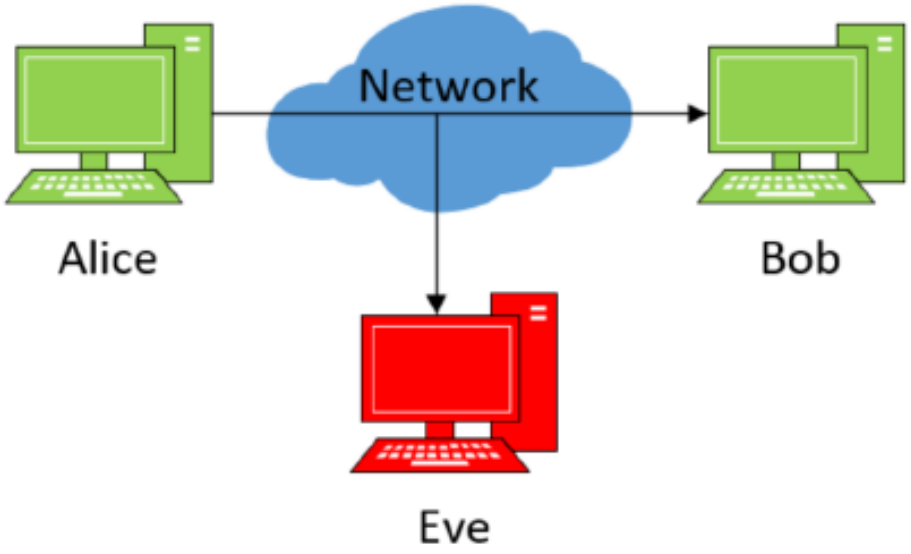

Figure 1. Eavesdrop Attack

2. Jamming

   A jamming attack is a form of denial of service attack. In this attack, Eve attempts to prevent Bob from receiving the communication from Alice. This attack is observable because unless Eve can also send alternate data, as in the Man in the Middle attack, Bob will know he is not receiving the communications.

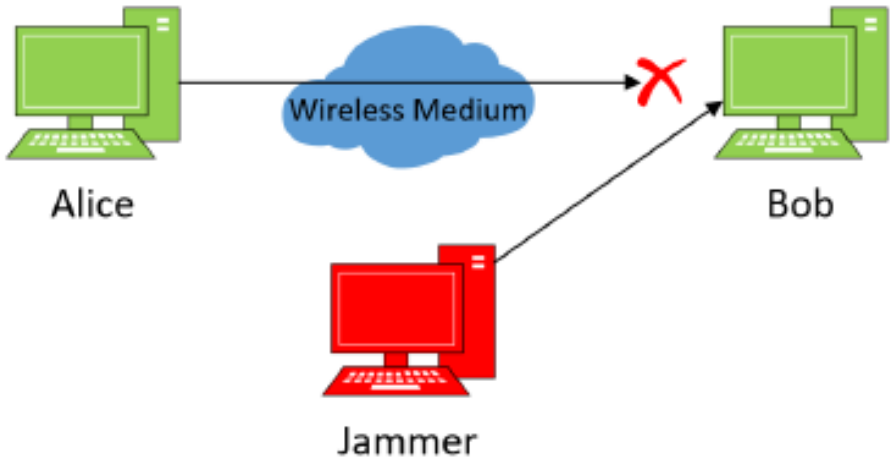

Figure 2. Jamming Attack

3. Man in the Middle (MitM)

   A MitM attack is similar to the Interception attack. However, in this scenario Eve injects, removes, or otherwise modifies the intercepted transmission, and then transmits the modified message to Bob.

   These attacks are particularly dangerous to Bob because if the attack is not identified, Bob will assume the information he receives is how Alice intended it to be.

Figure 3. Man in the Middle Attack

2.2.3 Tunable Security

The concept of tunable security stems from the continuously changing nature of security. Whether or not a given system is secure is determined at run-time by an end-user. This implies that security can have different meanings for different tasks or at different times. The CIA triad refers to three main "security" requirements; yet defining the "security" of a task requires an analysis of the end-user's goals. [6] states how many services "only provide one security configuration at run-time", preventing the user from any decisions regarding a performance security trade-off. The conceptual model the author provides is shown below in Equation 1

$$TS : T \times E \rightarrow S \tag{1}$$

Lundin equates tunable security to a function of the Tuner Preferences (T), set by the user; the environmental features such as QoS parameters, the ability to encrypt, or an adversary's access to the channel; (E). This particular, tuned, configuration maps to a given security level S. This concept of tunable security as a function of user preferences and services will be utilized in future work to describe the specific security provided by a multi-channel communication system's configuration.

2.3 Multi-Channel Communication

Multi-channel communication systems utilize sets of communication links to provide greater diversity in transmission mediums. The use of multiple channels can increase the confidentiality of a message by ensuring the entire message is not sent over a given link. This ensures that an adversary with access to one or more channels cannot determine an entire transmission's contents. Multiple channels can also improve integrity or availability by providing redundancy. A single channel relies primarily on encryption to secure its transmissions. However, if an adversary compromises that particular channel and is able to obtain the key, the transmission is unsecure. *Wolfe et al.* describe in [3] how transmitting information across two, distinct channels can allow the end user to achieve better CIA attributes than a corresponding single channel system. *Dolev et al.* prove that one can attain perfect secrecy and perfect resilience if there are a sufficient number of "wires", or channels, between the sender and the receiver. This implies that there exists a tradespace between the single channel system and the multi-channel system proven by Dolev, where the end user can achieve *better* security by increasing their number of channels. As the number of channels increases, there would be a corresponding increase in cost, along with the increase in security. This provides an end user the ability to determine what costs they are willing to incur for their given security requirements and use case.

2.3.1 Perfectly Secure Message Transmission

The Perfectly Secure Message Transmission (PSMT) problem was formalized in [7], which proposed an answer to the following problem: given a Sender and a Receiver, does a method exist whereby the sender can transmit a message m while satisfying the two conditions below:

Perfect Secrecy - For all sets L of at most σ wires, no listening adversary A_L

listening to all wires of L, learns anything about m.

Perfect Resiliency - For all sets D of at most ρ wires, the Receiver correctly learns m, regardless of the disrupting adversary A_D controlling and coordinating the behavior of the wires in D.

The authors assert that those two conditions can be met using a three phase protocol, FastSMT, which identifies compromised or faulty wires in phases 1 and 2, and utilizes phase 3 to transmit the secure message. Dolev's work implies that a multi-channel communication system can provide guaranteed secrecy and resiliency, even with adversarial action, provided there are sufficient channels. It is a simple extension then, that if there exist fewer channels, those guarantees transform into probabilities, specifically when $L > \sigma$ and $D > \rho$. The exact probability depends on the structure and contents of the transmitted data. This insight serves as a primary motivation for the development and use of multi-channel communications.

2.3.2 Cost-Performance Trade-Offs

Communication security research has traditionally focused on single-channel transmissions. I.e., there exists a single channel carrying data that may or may not be encrypted. *Wolfe et al.* provided an interesting exploration into the services provided by a two channel system, hinting at the potential for more channels. *Dolev et al.* demonstrate and proved that given enough wires (or channels), a communication system can provide perfect secrecy and perfect resilience. Thus, there must be a trade-space between a single channel, and Dolev's n channels, that can provide more security to the user, at the cost of performance.

2.4 Implementation/Security Mechanisms

Assuming one could identify security requirements, and establish an approach to handle multiple channels, the next logical concern would be how to implement it in a meaningful way within the existing ecosphere of communication systems. A multi-channel security system, by definition, includes more than one channel, hence it would likely be handled above the data link layer in a network stack. It may be a common service, suggesting that it should be handled below the application layer. Which layer to assign it to is an open question. This research proposes to place it at the transport layer, perhaps by inclusion in the popular Transport Layer Security (TLS), which is also at the transport layer. This would allow existing applications to utilize this system without changes. Additionally, this would allow for reliability guarantees provided by the lower layers, such as Transmission Control Protocol (TCP)'s guaranteed delivery and error detection at both the network and data link layers.

2.4.1 Transport Layer Security (TLS)

Transport Layer Security (TLS) 1.0, specified in January of 1999 in Request for Comment (RFC) 2246, provides communications privacy throughout the Internet. It "*allows client/server applications to communicate in a way that is designed to prevent eavesdropping, tampering, or message forgery*" [8]. Operationally, TLS is a widely used method employed to protect communications. RFC 2246 specifies the four goals of the TLS Protocol, in order of their priority, [8]:

1. Crytographic security: TLS should be used to establish a secure connection between two parties.

2. Interoperability: Independent programmers should be able to develop applica-

tions using TLS that will then be able to successfully exchange cryptographic parameters without knowledge of one another's code.

3. Extensibility: TLS seeks to provide a framework into which new public key and bulk encryption methods can be incorporated as necessary.

4. Relative efficiency: Cryptographic operations tend to be highly CPU intensive, particularly public key operations. For this reason, the TLS protocol has incorporated an optional session caching scheme to reduce the number of connections that need to be established from scratch.

To achieve the goals stated above, TLS relies on the TLS Record Protocol to provide connection security, and TLS Handshake Protocol to authenticate the two parties. TLS Record Protocol client specifications provide for two basic properties [8]:

- Private connections. Symmetric cryptography is used for data encryption. The keys for this symmetric encryption are generated uniquely for each connection and are based on a secret negotiated by another protocol (such as the TLS Handshake Protocol).

- Reliable connections. Message transport includes a message integrity check using a keyed MAC.

The TLS Record Protocol allows the encapsulation of various higher level protocols. One of these protocols is the TLS Handshake Protocol, which provides the user guarantees of authentication, confidentiality, integrity, and availability. The use of a multi-channel protocol could accomplish these same goals, perhaps better, if properly designed, implemented, and deployed. Some of the specific approaches which could be used will be described in Chapter III.

2.4.1.1 Deprecation of Secure Sockets Layer (SSL):

Transport Layer Security (TLS) became the *de facto* system for securing transport layer communications after the deprecation of Secure Sockets Layer (SSL) described in RFC 7568 [9]. SSL version 3, described in RFC 6101 was attacked over years in both its key exchange mechanism, and its supported cipher suites. For this reason, TLS 1.0 and 1.1, specified in RFC 2246 and 4346 respectively, was created. However, there was not widespread support of these replacement protocols, which allowed continued use of SSLv3 [9].

Starting with RFC 5246, TLS removed backwards compatibility with SSL. This ensured that no TLS sessions would allow the negotiation and use of SSL security. RFC 7568 states that "**SSLv3 Is Comprehensively Broken**". SSL has flaws in both its Cipher Block Chaining (CBC) modes, as well as a weakness of its stream ciphers. Its key exchange is vulnerable to Man in the Middle (MITM) attacks through two specified methods: renegotiation or session resumption. Its hashing functions rely on SHA-1 and MD5, which are considered weak and are being replaced with stronger hash functions [9]. TLS addresses each of these weaknesses and fixes them with newly researched cryptographic methods and features. RFC 7568 states that SSL must not be used, indicating a complete shift to TLS.

2.4.1.2 TLS Development:

TLS 1.0, first defined in 1999 in [8], did not indicate significant shifts from SSL. Specifically, it allowed for the negotiation of an SSL connection. The first major shift, made to secure several vulnerabilities to SSL, was made with TLS 1.1. These changes are summarized in [10]:

- Replacement of the implicit Initialization Vector (IV) with an explicit IV for protection against CBC attacks

- Changed handling of padding errors to protect against CBC attacks

The principal goals and properties of the TLS protocol remain the same from TLS 1.0-1.2. However, [11] includes a large list of changes from TLS 1.1. It allows for improved flexibility, specifically for the negotiation of cryptographic algorithms and the specification of cipher suite specific pseudorandom functions. There is support for authenticated encryption and additional data modes. TLS 1.2 removes support for certain cipher suites, such as IDEA and DES. And finally, TLS 1.2 lowers the support for SSLv2 backwards-compatibility from a "SHOULD" to a "MAY", with the assumption that it will become a "SHOULD NOT" in the future [11].

As the security environment continued to develop, written standards were needed to ensure that entities communicate security parameters using the same language to ensure clarity and efficient communication. These guidelines were specified in RFC 3552, "The Guidelines for Writing RFC Text on Security Considerations" [12]. TLS 1.3 contains both security updates from version 1.2, and a change in the goals of the TLS protocol to align with the language specifications in RFC 3552. The updated goals are listed in [13]:

- Authentication: The server side of the channel is always authenticated; the client side is optionally authenticated. Authentication can happen via asymmetric cryptography...or a symmetric pre-shared key (PSK).

- Confidentiality: Data sent over the channel after establishment is only visible to the endpoints. TLS does not natively hide the length of the data it transmits, though endpoints are able to pad TLS records in order to obscure lengths and improve protection against traffic analysis techniques.

- Integrity: Data sent over the channel after establishment cannot be modified by attackers without detection.

The major changes enumerated in TLS 1.3 reflect significant research in the field of secure communications [13]. It modifies the cipher suite concept to separate the authentication and key exchange mechanisms from the record protection algorithm. It prunes the list of allowable cipher suites to remove legacy algorithms, and removes static RSA and Diffie-Hellman cipher suites, instead only allowing public-key based mechanisms which provide forward secrecy - the assurance of past sessions' secrecy, even if future sessions are compromised. It requires handshake messages to be encrypted, and restructures the handshake state machine to be more consistent and remove overhead.

TLS 1.0 started as a response to the vulnerabilities in the SSL system, and has adapted to the now current TLS 1.3 as new vulnerabilities and capabilities have matured. This pattern demonstrates a willingness to adapt to an ever-changing security environment to develop new methods and protocols to maintain secure communications. This pattern should continue with the support of multi-channel communications in the years to come.

2.4.2 Message Authentication Codes

Receivers of any given security protocol need the capability to determine message integrity. At the minimum, this allows a system to determine whether or not messages have been altered in transit, whether by channel degradation or adversarial action. This section outlines two popular form of MACs - the CRC which provides error detection, and the RS Code which provides error correction.

2.4.2.1 Cyclic Redundancy Codes

Simple error detection relies on appending some bit or bits of overhead to a data packet. This overhead represents a "checksum" of the data to be transmitted. Com-

monly used examples of this are an XOR, two's complement addition, and a CRC. For purposes of this thesis, we will be using a 32-bit CRC for error detection.

A CRC utilizes polynomial division to compute a Frame Check Sequence (FCS) value representing the data. The sender uses polynomial division to divide the data by the CRCs's initializing polynomial. When the division is complete, the remainder (extended out to 32 bits) is appended to the data, and transmitted across a channel. The receiver computes the same value on the received data. If the corresponding output is 0, it indicates that the data was sent without error [14]. The strength of the CRC relies upon the polynomial chosen for the initialization, leading to the concept of a "weak" versus a "strong" CRC. Many commonly utilized polynomials are explored extensively by Koopman in [15] and updated results can be found at [16].

CRCs provide no error correction. Thus, they are commonly used in communications where retransmission is preferential to the increased overhead/computation required to correct transmission errors. Despite their strong ability to detect errors in transmission, they do not provide protection against MITM attacks. Because the initializing polynomial is assumed to be public, an adversary with access to a given channel can modify information, recalculate the CRC, and then transmit the new packet to the destination. The receiver will calculate the CRC based on the information it received, and it will be accepted. Thus, a CRC alone is not a good enough defense against MITM attacks.

2.4.2.2 Reed-Solomon Codes

RS codes are one of the first error correcting codes, proposed by Reed and Solomon in [17]. They have widespread use in many digital communication and storage applications. Figure 4 shows the structure of a Reed-Solomon data packet.

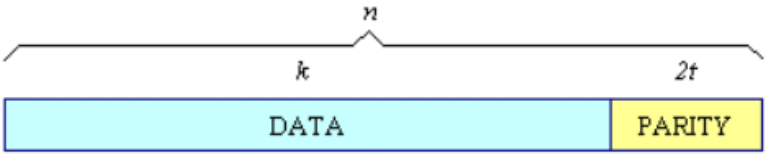

Figure 4. Reed-Solomon Structure

Given a symbol size s bits, the maximum length of a given RS codeword is $n = 2^s - 1$ bytes. Given an n byte packet, there are k data bytes, with $n - k = 2 \cdot t$ bytes of overhead. Given the properties of RS codes, this can correct errors in up to t bytes. As shown in [18], a popular RS code is RS(255, 223), with $s = 8$. This gives the following data values: $n = 2^8 - 1 = 255$, $k = 223$, $2 \cdot t = 32$. Thus, with 32 bytes of overhead, per 223 bytes of data, the RS code can correct errors in up to 16 data bytes, with a maximum number of errors of $16 \cdot 8 = 128$ bit errors.

2.4.3 Encryption

Encryption is a branch of Cryptography, "the science of secret writing with the goal of hiding the meaning of a message" [5]. The goal is to obfuscate the meaning of a message, so that even if an adversary has access to a transmission's contents, the meaning is hidden: the message appears to be a random collection of bits. To be a successful cryptosystem, it should follow *Kerckhoffs' Principle*:

> **Kerckhoffs' Principle:** A cryptosystem should be secure even if the attacker (Oscar) knows all details about the system, with the exception of the secret key. In particular, the system should be secure when the attacker knows the encryption and decryption algorithms.

Figure 5 below shows how encryption would prevent a simple eavesdropping attack:

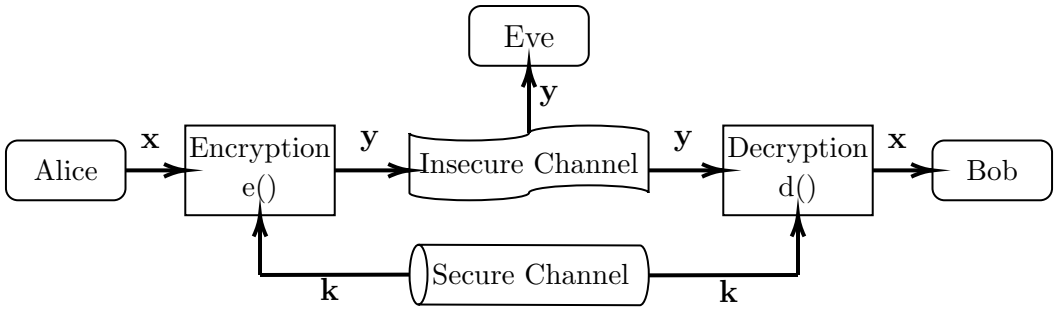

Figure 5. Blocking an Eavesdropping Attack Through Encryption

By encrypting the message through the use of some secure key k, and some encryption/decryption function ($e()$ and $d()$), Eve intercepts y, that has no apparent relation to the original message x.

At a very simplistic level, encryption relies on the scrambling of the data. If one disregards message padding, each bit of input (or block of input) correlates directly to a block of output in the ciphertext. Its security relies on the low probability of an adversary decrypting the contents in a reasonable time period. Most modern cryptographic systems rely on the mathematical challenge of factoring a very large number into its associated primes. As processing technology develops, they simply shift to larger and larger primes. However, eventually these technologies will be less effective. One example of a technology that would render this encryption obsolete is the finished development of the quantum computer. Thus, a future time is anticipated where security must be maintained through methods which do not rely on traditional encryption.

2.4.4 Data Fragmentation

The use of a multi-channel communication system presents the opportunity to fragment a message into unique, discrete fragments, and then distribute those fragments across the available channels. While there may be many methods to fragment

the message, ranging from deterministic splitting to elaborate functions, an efficient and ideally a cryptographically secure method can provide significant obfuscation without excessive overhead. This cryptographic security implies that an adversary cannot predict future outputs by observing and recording past outputs. A Linear Feedback Shift Register (LFSR) provides an elegant way of realizing long, pseudo-random sequences with minimal software/hardware requirements, making it an ideal candidate for data fragmentation. A LFSR consists of a series of flip flops and a feedback path. Figure 6 shows a simple LFSR consisting of three flip flops, with a feedback path consisting of the output of FF_1 and FF_0, as described in [5]. The LFSR outputs a single bit s after each clock cycle.

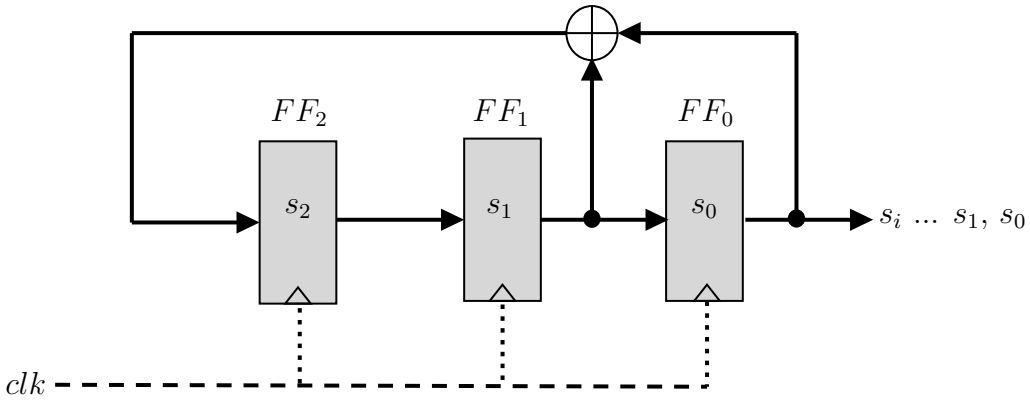

Figure 6. Simple LFSR

Given an m-bit LFSR, the maximum output length is shown below [5]:

$$Length = 2^m - 1 \tag{2}$$

After $2^m - 1$ values, the sequence will repeat itself, this length is the LFSR's period. Because of the linear progression of its internal state, the LFSR's sequence can be broken with $2 \cdot m$ [5].

This leads to the Trivium cipher. Trivium is a "hardware oriented synchronous

stream cipher" [19]. By chaining together three LFSRs, the internal state of each LFSR does not evolve in a linear fashion. Figure 7 shows the internal structure of the Trivium stream cipher [19], where s_n indicates the n^{th} bit of state, and z_i is the i^{th} bit of output. This construction gives the Trivium cipher a period of 2^{64} bits.

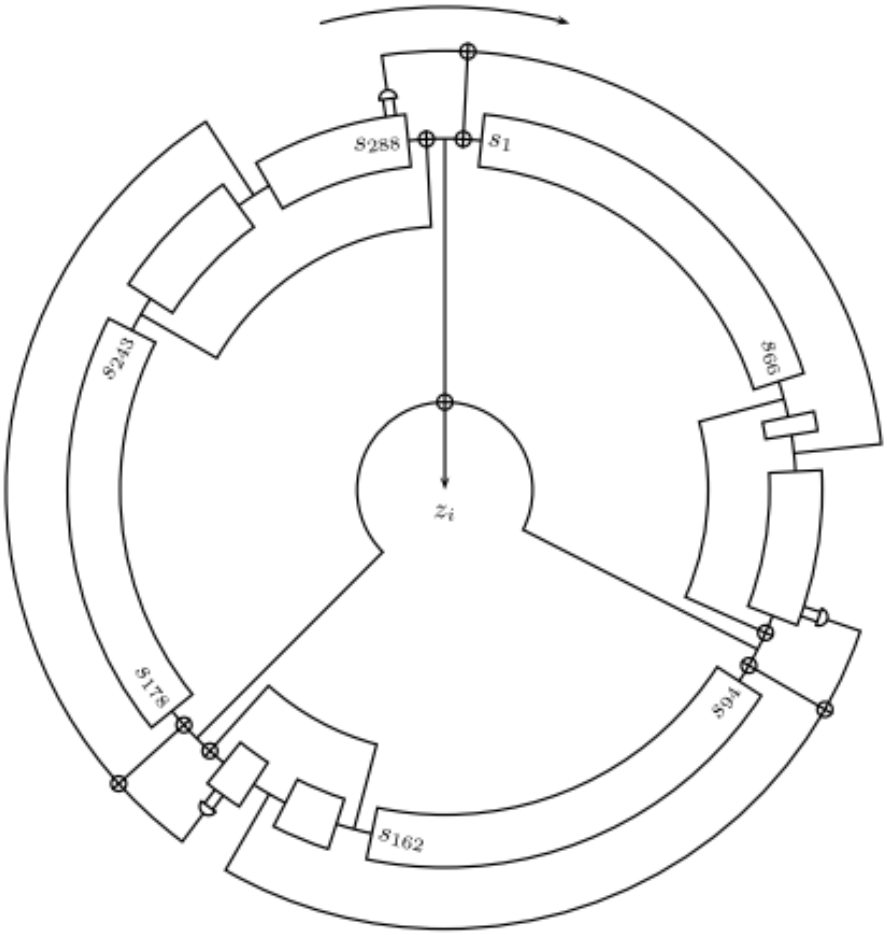

Figure 7. Trivium Internal State

This cipher's simple construction allows for low power implementations in hardware, while providing cryptograpically secure outputs.

2.4.5 Diffie-Hellman Key Exchange

As mentioned in 2.4.3, the most important aspect of a cryptosystem is the key. Thus, a secure method must be utilized to exchange the initialization parameters for the system over an insecure channel. The Trivium cipher requires 160 bits of information to initialize the system. If a sender and receiver must utilize Trivium in a synchronized fashion, they require the exchange of that initialization information – a key exchange. The DHKE algorithm is a one-way function that relies on the commutative property of exponentiation. This algorithm has two phases - a setup and the exchange [5].

Algorithm 1 DHKE Setup

Choose a large prime p
Choose an integer $\alpha \in \{2, 3, ..., p-2\}$
Publish p and α

Once the public parameters p and α are published, both Alice and Bob do the following operation (shown for Alice)

Algorithm 2 DHKE Exchange

Choose $a = k_{priv,A} \in 2, ..., p-2$
Compute $A = k_{pub,A} \equiv \alpha^a \mod p$
Send A to Bob, and receive $B = k_{pub,B}$ from Bob

Both Alice and Bob can compute the key K_{AB} using the following operation:

$$k_{AB} = (k_{pub,B})^k_{priv,A} \equiv B^a \mod p \tag{3}$$

The proof of this key exchange's validity stems from the exponentiation utilized:

DHKE Proof: Alice computes the following:

$$k_{AB} = B^a \equiv (\alpha^b)^a \equiv \alpha^{ab} \mod p$$

Bob computes the following:

$$k_{AB} = A^b \equiv (\alpha^a)^b \equiv \alpha^{ba} \mod p$$

Thus, both Alice and Bob share the session key k_{AB} [5].

This method can be utilized to exchange information for a data fragmentation scheme. Alice and Bob will both compute a session key k_{AB}, and can utilize the Most Significant Bits (MSBs) to generate the key and IV for the data fragmentation scheme.

2.5 Quality of Service (QoS)

Quality of Service is defined as is the "Capability to control traffic-handling mechanisms in the network such that the network meets the service needs of certain applications and users subject to network policies and strategies" [20]. QoS metrics typically include bandwidth, packet delay, timing jitter, and packet loss. A customer's needs can change, requiring different levels of QoS metrics, and ignoring others.

Providers need to determine methods to provision their services, in order ot satisfy their customers. [21] describes an analytical method to combine services along different channels of an end-to-end route.

2.6 Regulatory Standards

The North American Electric Reliability Council (NERC) created the Critical Infrastructure Protection (CIP) security standards CIP-002-014 [22] to formalize the security requirements for the entire energy sector, ranging from personnel & training requirements in CIP-004-6 to Information Protection, outlined in CIP-011-2, that will be the focused standard in this article. The purpose CIP-011-2, is "*To prevent*

unauthorized access to BES Cyber System Information by specifying information protection requirements in support of protecting BES Cyber Systems against compromise that could lead to misoperation or instability in the Bulk Electric System (BES)".

Fries, in [23], references a set of standards created by the International Electrotechnical Commission (IEC), specifically IEC 62443-3-3, that states two requirements directly related to secure communication:

- Requirement 3.3.1 Communication Integrity: "*The control system shall provide the capability to protect the integrity of transmitted information*"
- Requirement 4.4.1 Communication confidentiality: "*The control system shall provide the capability to protect the confidentiality of information at rest and remote access session traversing an untrusted network*"

2.7 Summary

There is clearly a vested interest in developing mechanisms to guarantee the confidentiality and integrity of communications, specifically in the field of critical infrastructure. There are regulatory standards that dictate the requirements for secure communications. An understanding of the interaction between operational CIA requirements and possible attack vulnerabilities allow the user to determine their own, unique security requirements. Given these security requirements, a communication system ought to provide the ability to meet those requirements efficiently. One of the primary methods securing networked communications currently is TLS. The TLS protocol allows a sender and receiver to negotiate a specific security configuration, demonstrating tunability. TLS has gone through several iterations to reflect the changing needs of the security environment, reflected in the three main goals of TLS 1.3: authentication, confidentiality, and integrity.

Security research traditionally has only focused on the security of individual channels. A given channel has its own security attributes and threats, but the relationship between channels has not been widely studied. *Wolfe et al.* investigated a two channel system, while *Dolev et al.* theorized about the merits of an n channel system. This work will investigate the area between those two works - multi-channel communications. This work proposes a data fragmentation scheme based on the existing Trivium cipher, which will be utilized to distribute a message's contents across the available channels. The remainder of this thesis develops a common terminology to describe a multi-channel communication system using data fragmentation. This theory leads to an investigation into the theoretical benefits of a multi-channel protocol, and a proof of concept design which demonstrates the challenges to its implementation, and the security services gained through its use.

III. Multi-Channel Communication Terminology and Theory

3.1 Introduction

This chapter outlines the terminology and theory required for the discussion of a multi-channel communication system. and which informs the proof of concept design and the creation of a specific protocol to allow for precise communication of session parameters. Additionally, this enables description of the primary motivations for the use of a multi-channel protocol, independent of specific design decisions. These motivations are firstly the resilience of this protocol against adversarial action, and secondly the tunable security it provides to a user.

3.2 Multi-Channel Theory

This section will establish the relationships and terms required for the discussion of a multi-channel protocol. Additionally, these relationships will be applied to an example message transmission to demonstrate their reliability.

3.2.1 Terminology

Assume a user wants to send message M, with $|M| = m$. This message is broken into a set of k fragments F_M, satisfying the following relationships:

$$F_M = \{f_1, f_2, f_3, ..., f_k\}$$

$$f_i \subseteq M \text{ for } 1 \leq i \leq k$$

$$f_1 \not\subseteq f_2 \not\subseteq ... \not\subseteq f_k \qquad \text{Each fragment is unique}$$

The Fragmentation Factor (FF_M) will indicate the fragmentation level of the message. This metric is simply the number of fragments which compose M.

$$FF_M = k$$

A high fragmentation factor indicates a higher tunability for the end user because there are simply more ways to distribute the message fragments across the available channels. Let C be the set of n available channels.

$$C = \{C_1, C_2, C_3, ..., C_n\}$$
$$C_j \subseteq F_M \text{ for } 1 \leq j \leq n$$

This allows the calculation of the channel load, which indicates the percentage of M present on a given channel.

$$L_j = \frac{\sum |f| \text{ for } f \in C_j}{m}$$

The Average Channel Load (ACL) presents the expected percentage of M transmitted on any given channel, and is the average of L_j for all channels:

$$ACL_M = Average(L_j) \text{ for } 1 \leq j \leq n$$

The ACL increases as the user decides to utilize duplication. This leads to two

important items for consideration:

1. How many times the user transmits a given fragment. This matters because this is a measure of the reliability/redundancy of the system. If fragment f_1 is transmitted five times, but f_2 is only transmitted once - then the loss of f_2 results in the inability to recreate the message. This metric is simply the $Count(f_i)$, indicating the number of times that f_i occurs across all the channels in C. The minimum $Count(f)$ for $f \in F_M$ indicates the minimum number of channels required for an adversary to prevent the successful receipt of the message.

2. An important consideration for efficiency is the overhead of transmitted data. i.e., how much message data is being sent, relative to the size of M. This is the Spatial/Static Duplication Factor (SDF)

$$SDF(M) = \frac{Count(f_i) \cdot |f_i| \text{ for } 1 \leq i \leq k}{|M|} - 1 \tag{4}$$

   Equation 4 quantifies the amount of duplication the user transmits. $SDF(M) = 0$ indicates that there is no duplicate information, while $SDF(M) = n - 1$ indicates that each channel carries an entire copy of the message.

The construction of the equations and relationships in this section allow for larger analysis of a multi-channel protocol which transmits multiple messages, as well as fragmentation into unique-sized fragments.

3.2.2 Example Transmission:

This section will step through the transmission of a single message to demonstrate how the metrics described above can be applied. Note that this uses a fragmentation

scheme which presents unequal sized fragments to better illustrate the insights these metrics provide.

Parameters.

Let M be a message of size m, with $n = 3$ channels, and $|F| = 3$.

The following relationships describe the fragmentation scheme and the mapping of the individual message fragments:

$$F = \{f_1, f_2, f_3\}$$
$$|f_1| = \frac{5 \cdot m}{7}, \text{ and } |f_2| = |f_3| = \frac{m}{7}$$
$$C_1 = \{f_1\}, C_2 = \{f_1\}, C_3 = \{f_2, f_3\}$$

Channel Load.

The following calculations demonstrate the load on each channel:

$$L_1 = \frac{\frac{5 \cdot m}{7}}{m} = \frac{5}{7} \qquad L_2 = \frac{\frac{5 \cdot m}{7}}{m} = \frac{5}{7} \qquad L_3 = \frac{\frac{m}{7} + \frac{m}{7}}{m} = \frac{2}{7}$$

This gives:

$$ACL_M = \frac{\frac{5}{7} + \frac{5}{7} + \frac{2}{7}}{3} = \frac{4}{7} = 0.57$$

This result indicates that the average channel transmits 57% of the message, which accounts for both Channels 1 and 2 sending a higher percentage of the message.

Duplication Factors.

First, obtain a count of each fragment:

$$Count(f_1) = 2 \qquad Count(f_2) = 1 \qquad Count(f_3) = 1$$

Because the minimum fragment count is 1, if either f_2 or f_3 is lost/damaged, the message cannot be recreated.

The final calculation is the SDF based on size of the fragments:

$$\begin{aligned} SDF &= \frac{2 \cdot \frac{5m}{7} + 1 \cdot \frac{m}{7} + 1 \cdot \frac{m}{7}}{m} - 1 \\ &= \frac{\frac{12 \cdot m}{7}}{m} - 1 \\ &= \frac{12}{7} - 1 = 1.714 - 1 = 0.714 \end{aligned}$$

This result indicates that there was actually a total of 71% of additional information transmitted.

The relationships described in this section will allow the end user to quantify the performance of their system relative to standard QoS metrics. Optimally, the end user ought to be able to prioritize high bandwidth, secure channels by assigning them more message fragments. The equations and relationships developed allow for the analysis of unique channels, carrying different sized message fragments, across multiple message transmissions.

3.3 Duplication Considerations

One important consideration for a multi-channel architecture is the ability to transmit duplicate information to facilitate the recovery of lost or damaged informa-

tion. The amount of duplication presents a tunable parameter that can be set by the user to meet their specific security requirements. Any duplication produces an increased risk to confidentiality because an adversary can obtain a larger share of the message by intercepting or listening to a given channel. However, it also provides the user an increased guarantee of integrity and availability because there exist multiple copies of the same data. This highlights a fundamental trade-off between confidentiality, integrity, and availability.

By limiting the user to the strict duplication of information, there is a finite range of values an end user can utilize. The minimum value would be no duplication, $SDF_M = 0$. This would indicate that each channel carries unique information, and the user requires all channels' information to re-assemble the message. Alternatively, the user can send a full copy of the message on each available channel, $SDF_M = n-1$. These two extremes indicate the maximum and minimum levels of confidentiality, integrity, and availability available to the end user. The area between these extremes represents the system's tunability for the end user.

3.4 Theoretical Multi-Channel Framework

This section will outline the operation and benefit of a multi-channel communication framework, independent of specific design decisions, using the traditional cyber security characters of *Alice*, *Bob*, and *Eve*.[1] This will identify the benefits of a generic multi-channel protocol which allows for the fragmentation and duplication of data, using the previous terminology. This will establish what is possible using such a protocol, and provide a comparison for the design decisions made in the proof of concept system. The concept of a channel is intentionally vague. A channel could simply utilize a unique port from other channels, but could also indicate a physically

[1]When referring to "best" and "worst" case, this will be from the perspective of *Alice* and *Bob*, not *Eve*.

distinct system. The security services, specifically the resilience to adversarial action, considered in this chapter are independent of those design decisions. The specific definition of a "channel" only affects the probability of an adversary observing or affecting the channel's transmission. For example, if a system utilized a single Ethernet line with different TCP/User Datagram Protocol (UDP) ports for each "channel"; an adversary could observe each channel if it compromises the ethernet line.

3.4.1 Resilience to Adversarial Action

Traditional security measures rely heavily on the use of encryption to protect data and sensitive information. Encryption, in turn, relies entirely on the protection of a key. If an adversary has access to a communication channel, they theoretically have the capability to decrypt and decode the message, given enough time or computing resources. This protocol aims to provide increased security by allowing the user to not transmit the entire message across a given communication channel, leaving the attacking without portions of the data stream, assuming the attacker does not have access to each of the channels employed. Three examples illustrate the protocols' resilience against an interception attack, a jamming attack, and a MITM attack.

3.4.1.1 Interception Attack

Consider a communication session between a sender, *Alice*, and a receiver, *Bob*. Alice wants to send a message to Bob while providing a guarantee of Confidentiality. Let there be $n = 3$ channels between Alice and Bob, $Channels = \{C_1, C_2, C_3\}$; and a listening adversary *Eve*, who listens to the set of channels L, with $1 \leq L \leq n$, and knows the total message length m. The communications between Alice and Bob will be fragmented at the bit level using a cryptographically secure splitting mechanism. The length of the data sent on each channel is $|C_1| = |C_2| = |C_3| = \frac{m}{3}$. We will now

investigate the best and worst case scenario for this adversarial setup.

1. **Single Channel Interception:** $|L| = 1$

   Eve, the listening adversary has access to exactly $\frac{1}{3}$ of the data sent, let $L = C_1$. If the message had simply been divided into thirds and sent, this could represent a significant amount of information leakage, with leakage being some combination of interception, decoding, and decryption. The adversary has already accomplished two of those three tasks: they intercepted information, and it was never encrypted, so the information they have is actual data from the original message. Because the protocol split the message data at the bit level, this ensures that the adversary does not know **which** bits they have. This indicates that the challenge for the adversary lies in the decoding of the message bits they intercepted. To decode the entire message, the adversary needs to place the m bits correctly.

   The worst case scenario would occur if Eve knows the placement of C_1 - knowing which bits of M were assigned to C_1. From there, Eve would be forced to guess the value of the remaining $\frac{2 \cdot m}{3}$ bits. With equal probabilities for each bit, this gives the following:

   $$P(decode) = (\frac{1}{2})^{\frac{2 \cdot m}{3}}$$

   The best case scenario would occur if Eve was unsure of the placement of C_1. This would cause a corresponding decrease in $P(decode)$ because in addition to guessing the values of the bits assigned to C_2 and C_3, Eve must also guess their placement to recreate M.

2. **Three Channel Interception:** $|L| = 3 = |C|$

   This assumes Eve has access to all of the data, unencrypted. The worst case

scenario would still be if *Eve* knows the placement of the message bits. This would lead to a successful interception/eavesdropping, with *Eve* learning the message's contents. The best case scenario would occur if *Eve* does not know the placement of the bits, which would require properly placing all m bits into their proper locations. Even though *Eve* knows the contents, this would not be trivial.

One can clearly see the security benefits of a fragmenting or splitting protocol. Even in the presence of an adversary with access to the original, unencrypted data, the adversary still requires an almost brute-force search to decode the information into its meaningful components when only intercepting a partial message. *Eve* must know the session key used to map each data bit to its corresponding fragment and channel to gain meaningful information.

3.4.1.2 Jamming Attack

Another benefit of this protocol is the resiliency to a jamming attack. Consider again a communication session between *Alice* and *Bob*. Alice wants to send a message to Bob while providing a guarantee of Integrity, indicating that *Alice* will duplicate information. Let the message M be split at the bit level into three message fragments f_1, f_2, and f_3.[2]. Let there be $n = 3$ channels between Alice and Bob, $Channels = \{C_1, C_2, C_3\}$, with the following fragment structure: $C_1 \rightarrow \{f_1, f_2\}$, $C_2 \rightarrow \{f_2, f_3\}$, $C_3 \rightarrow \{f_1, f_3\}$. The length of the data sent on each channel is $|C_1| = |C_2| = |C_3| = \frac{2 \cdot m}{3}$.

[2]These fragments will be assumed to have identical sizes

Using the equations developed in Section 3.2.1, we derive the following values:

$$L_1 = L_2 = L_3 = \frac{2}{3}$$
$$ACL_M = \frac{2}{3}$$
$$Count(f_1) = Count(f_2) = Count(f_3) = 2$$
$$SDF_M = \frac{2 \cdot \frac{m}{3} + 2 \cdot \frac{m}{3} + 2 \cdot \frac{m}{3}}{m} - 1 = 1$$

Now consider a jamming adversary *Eve*, with the capability to jam C_1. Despite the presence of jamming on the particular channel, Bob can recreate the message using only the fragments transmitted on C_2 and C_3. This can also be seen by investigating the values of the fragment counts, or the SDF_M: because all fragments occur twice, a single deletion cannot prevent the full message from being received; however, the calculation of $SDF_M = 1$ indicates *Alice* transmitted 100% of additional data to secure the transmission against a jamming adversary, which represents a significant bandwidth cost.

Consider a similar, 10 channel scenario, with the same fragmentation scheme (fragments comprising $f_1 - f_{10}$ instead of $f_1 - f_3$). This gives the following relationships:

$$L_1 = L_2 = L_3 = ... = L_{10} = ACL_M = \frac{2}{10}$$
$$Count(f_1) = Count(f_2) = Count(f_3) = ... = Count(f_{10}) = 2$$
$$SDF_M = 2 - 1 = 1$$

In the worst case, the message can be lost if an Eve jams two channels sharing the same message fragment. However, *Eve* must correctly guess which channels to jam, a probability of $\frac{2}{9}$. In the best case, *Alice* and *Bob* could lose 5 channels, and still properly receive the message using the fragments on the remaining 5 channels.

This can lead to the following relationship regarding the resiliency of the multi channel architecture to a jamming attack, specifically the minimum and maximum number of channels needed to either destroy or recreate the message. The minimum number of channels required for *Eve* to destroy the message equals the minimum $Count(f)$ for $f \in F_M$; however, *Eve* can guarantee message destruction by compromising $x > \frac{|F_M|}{Avg.Count(M)}$ channels. Thus, for the 10 channel scenario, *Eve* can destroy M by compromising 2 channels, and can guarantee success by compromising 6 channels. The minimum number of channels needed to recreate the message is $\frac{|F_M|}{Avg.Count(M)}$: a single copy of each fragment must be received. However, *Alice* and *Bob* can guarantee successful receipt as long as *Bob* receives $> n - Min.Count(M)$ channels' transmissions. Thus, for this transmission scenario, *Bob* can decode M from 5 channels, and can guarantee success by receiving 9 channels.

3.4.1.3 Man in the Middle Attack

Typically, if an adversary conducts a MITM attack successfully, the recipient is unaware of the presence of the attack upon receipt. There are several protections which this architecture provides to a MITM attack, as well as several features which provide added utility to the end users.

Assume again that *Alice* wants to send a message to *Bob*, and that *Eve* will be attempting a MITM attack. We start with the message M, and append some form of Message Authentication Code. Thus, *Alice* will distribute $Data = M + MAC_M$ to the available channels, which distributes both the message bits and message MAC. Each channel will then append a channel-specific MAC to its transmission. We will again assume three channels, C_1, C_2, C_3, and three fragments f_1, f_2, f_3.

The first scenario is that with no duplication, $SDF_M = 0$: $C_1 = \{f_1\}, C_2 = \{f_2\}, C_3 = \{f_3\}$. Let *Eve* modify $C_1 = f_1 \rightarrow f_1'$, to include the modification of the

channel MAC to ensure *Bob* will accept the channel contents upon receipt. Thus, when *Bob* attempts to recombine f_1', f_2, and f_3 into M, the recombined message will have an incorrect Message Authentication Code, and will fail. However, *Bob* will not know which channel caused the error. As shown in Section 2.2.2, these types of attacks can be severely harmful. Thus, even with no information duplication, this architecture provides a baseline of resilience against a MITM attack by detecting the attack while recombining the message, even if the receiver initially accepts the modified message fragment.

The second scenario is that with a $SDF_M = 1$: $C_1 = \{f_1, f_2\}, C_2 = \{f_2, f_3\}, C_3 = \{f_1, f_3\}$ Let *Eve* modify $C_1 = \{f_1, f_2\} \rightarrow \{f_1', f_2'\}$. When *Bob* attempts to recombine the fragments into M, the recombined message will have an incorrect MAC, and thus will fail. However, now because of duplication, *Bob* can attempt to recreate the message only using the fragments transmitted on C_1 and C_2, which will fail; C_1 and C_3, which will fail; and finally C_2 and C_3, which will succeed, with no additional transmissions. Because he was successfully able to recombine the message from those two channels, *Bob* can identify that the error was caused on C_1, and adjust his transmission posture appropriately.

Both of the above scenarios demonstrate the effectiveness of this architecture to detect injection attacks which were previously unidentifiable. However, as shown in Scenario #2, through duplication, the receiver can determine which channel is compromised, while still recovering the original message without retransmission. It must be noted that, with duplication, *Eve* can leverage the duplicate information to decode the transmitted message. To mitigate this threat, additional entropy might need to be added to address this weakness.

3.5 Fragmentation/Splitting Tunability

With regards to the fragmentation of the message, the duplication, and the number of channels, it is clear to see the relationship between them, and the limits that imposes on the tunability of the system. The user does not have the ability to set all three aspects independently. However, the user may select any values for two of the three "tunable" factors, and that selection drives the third.

This requires that we analyze the bounds of the three terms, and see how those limits affect the overall system.

3.5.1 Bounds

The channel load indicates the percentage of M that is transmitted on a given channel. The lower bound for this is that $L_{ch} > 0$. If $L_{ch} = 0$, this indicates that none of the message M was sent over the channel, which would remove it from the set of utilized channels. The upper bound is also easy to determine, and it occurs when a given channel sends the entire message M, giving $L_{ch} = 1$.

The lower bound for the spatial duplication factor indicates the scenario when none of the message is duplicated. Thus, every fragment of the message is sent once, giving a total $SDF_M = 1 - 1 = 0$. The upper bound for the duplication would occur when every channel sends the entire message. Thus, the upper bound is $SDF_M = n - 1$.

The number of channels has no upper bound; however, n must be greater than or equal to 1. If $n = 0$, there would be no connection between sender and receiver.

Term	Bounds
Channel Load	$0 < L_{ch} \leq 1$
Spatial Duplication Factor	$0 \leq SDF \leq n - 1$
# Channels	$1 \leq n \leq \infty$

Table 1. Fragmentation Bounds

3.6 Tunability

The three terms bounded above illustrate some of the tunable characteristics for different aspects of a theoretical system: the Channel Load applies to a given channel, the Spatial Duplication Factor applies to a given message, and the number of channels applies to a given session. By investigating the causes for each value's maximum and minimums, one can see that the selections of certain parameters drives the result of others. For example, by setting the Channel Load to its maximum value of 1, it drives the Spatial Duplication Factor to its maximum of $n - 1$, independent of the number of channels. By setting the number of channels and the SDF, it determines the required Channel Load. Thus, the end user would be able to set values according to the specific constraints of their use case and security needs.

3.7 Summary

This section defined several important terms and characteristics of a multi-channel, data fragmenting protocol. These terms need to be established prior to making specific design decisions to ensure precise communication, and to allow comparison between various systems with a unified language. A theoretical multi-channel system was outlined, establishing how a multi-channel protocol can provide resiliency to the end user, and the bounds of that resiliency given a session's characteristics. This theoretical baseline is independent of specific operational constraints such as the method of fragmentation, the method of error detection/correction, and the characteristics of the communication channels. These issues are factored in to specific design decisions in the creation of the proof of concept system.

IV. Proof of Concept Design

4.1 Objective

This chapter details the design decisions made in the proof of concept system, and the transmission scenarios for which it will be evaluated. It outlines the consideration of various tools or languages to be used in the construction of the prototype system, as well as how to evaluate the performance. A multi-channel system could be built within an event-based network simulator. This would simulate system use in a realistic environment, incorporating the concepts of buffering, processing, and transmission delays into the system's operation. Additionally, it would ensure each channel was unique. However, this would have proven more cumbersome than necessary to test the security concepts examined, because a statistical experiment was not strictly necessary. Packet loss from channel noise and router drop are already handled at the Transport Layer through TCP's message delivery guarantees. Use of a simulator could provide insight into delay or other availability performance parameters, but that was not the intent of this effort. Instead, these tests are confirmatory in nature. Given a transmission scenario - a number of channels, a fragment to channel mapping, an amount of duplication, and adversarial actions - the outcome ought to be deterministic. If the protocol operates at the Transport Layer, using existing protocols such as TCP, the sender is guaranteed that the packet arrives at the destination. At this stage of development there is no need for a stochastic experiment, and the additional realism of a simulator's environment would have provided data that was outside of the scope of this research. Such studies are left for future work.

Although nearly any higher level language could have been used to construct the system, Python was chosen, allowing a simple approach to simulate network communications in specific configurations, while ignoring extraneous information. One of the

primary goals of this system was to demonstrate that the splitting and recombination elements of the proposed system could function as intended. This indicates the need for confirmatory test cases, rather than statistical experiments. Given a set of transmission scenarios, the system should identify an affected message, potentially identify which channels have been compromised, and if duplication allows - reconstruct the message. Additionally, the amount of information on each channel can be measured, to calculate the percentage of overhead incurred, verifying the mechanisms function as desired. Another benefit to this constructing a propotype is learning about the issues in the actual implementation of this system by simulating an end-to-end system. This end-to-end system must be able to fit into an existing communication security system which could benefit from its use, as well as adopt its use without significant changes to existing infrastructure. This rationale led to the investigation into the TLS protocol, which already allows negotiation between a sender/receiver to identify "acceptable" security levels for a given communication. As mentioned in Chapter II, TLS also requires guarantees of Confidentiality, Integrity, and Authentication [13]. This dictates the need for a crytographically secure splitting mechanism. Unlike a simple Pseudo-Random Number Generator (PRNG), which is easily implemented, a crytographically secure mechanism prevents an adversary from predicting future message fragments by monitoring the system's previous outputs.

4.2 Fragmentation and Duplication Considerations

Fragmentation.

The only requirement for the splitting mechanism is that the sender and receiver need to share the same mechanism, as well as have a method to exchange initialization information. This indicates that the system can be tuned for specific use cases. *Wolfe et al.* in [3] describe how the encryption of a single gigabyte of information requires

$0.41Wh$ of energy just for the encryption of the data. Similarly, using 128-bit AES to encrypt a single, 128-bit block of data took a low-power sensor 1.1 milliseconds, 946 bytes of random access memory, and 23.57 microjoules. These amounts can be detrimental to a device with limited power or memory. Low power devices could utilize non-crytographically secure methods such as PRNGs, and simply exchange the seed value. This would enable them to secure their transmissions purely through the use of different connections, and not require the use of any encryption or cipher system. However, this would imply that the additional connections incur a lesser cost than the process of encryption. This depends on the type of connections available and the provided hardware, and is outside the scope of this work. Devices which require more robust communications security can utilize more secure stream ciphers and key exchange mechanisms. This allows the system designer to determine the overhead they are willing to incur from the splitting mechanism itself, relative to the specific use case, with the understanding that there exists a relationship between the specific splitting mechanism used, the amount of overhead required, and the security it provides given an adversary.

The Trivium cipher was chosen as the basis of the splitting mechanism to assign each data bit to a specific fragment. This implementation requires 288 bits of internal state, as well as two, 80 bit numbers to initialize the system. Because it is constructed of simple LFSRs, the hardware requirements are minimal and could be a suitable construction for a low-power device which can accomplish the DHKE.

Logical Duplication.

If the duplication of information is limited strictly to bit-level duplication, there is no method to recover a lost fragment without full duplication of the message information (bits). By utilizing logical operations between fragments, it is possible to recover

lost information with fewer overhead bits than message bits. For example, consider a 3-channel, 3-fragment system. Alice sends information with the following mapping: $C_1 = f_1, C_2 = f_2, C_3 = f_1 \oplus f_2$. If any single channel's information is lost, the full message can be recovered. This demonstrates a method in which full recovery can occur with only 50% overhead, or 1 bit of overhead per 2 bits of data. Additionally, by introducing a link between fragments, there is a significant decrease in entropy of the transmitted information. The link between f_1 and f_2 to produce $f_3 = f_1 \oplus f_2$, indicates that an adversary can determine the entire message by intercepting two of the channels. In order to mitigate this weakness, an additional obfuscating mechanism such as in [24] could be created. This would prevent an adversary from identifying the link between bits, however it would cause a significant increase in overhead and complexity. Logical duplication also provides no allowance for unequal fragment sizes. In the above example, f_1, f_2, and f_3 must be equal sizes in order to satisfy the logical relationship. Thus, an adversary knows the length of the transmission must be an integer multiple of whatever data they intercept.

For these reasons, the proof of concept system only utilizes non-logical duplication. This requires the user to transmit no fewer than two copies of the data if they wish to recover from adversarial action on a single channel. This system, in conjunction with the Trivium splitting mechanism, provides the adversary limited information about the contents of intercepted channels. By utilizing a probabilistic method to determine the bit-fragment mapping, each channel can carry different data lengths.

4.3 Proof of Concept Design

The Proof of Concept will use *Alice* and *Bob* as the sender and receiver, with *Eve* as an adversary. *Alice* and *Bob* will utilize the DHKE algorithm to calculate a session key. The given session key will allow *Alice* and *Bob* to extract the required

information to initialize the splitting mechanism - the Trivium stream cipher - by using the upper 160 bits from the session key as the 80 bit Trivium key, and 80 bit initialization vector. Given the number of fragments $|F_M|$, *Alice* and *Bob* will output $x = \lceil \log_2(|F_M|) \rceil$ bits from the Trivium cipher for each data bit, which will assign the bit to a specific fragment. Then, each channel carrying the given fragment will append the data bit to its transmission.

There are two available methods for message authentication. A 32 bit CRC will be used for error detection. Alternatively, a (245,255) RS code will be used for error correction, indicating that for every 255 transmitted bytes, 245 bytes are data, and 10 bytes are error correcting information. When using forward error correction, the CRC will still be used for the message Error Correcting Code (ECC), while each channel uses the RS code to secure its transmission.

Three notional example messages were used: a text file containing the final six lines of the poem "Ulysses", by Alfred, Lord Tennyson; an image of the Maroon Bells mountains in Colorado; and a text file of "Hamlet", by William Shakespeare. These illustrate the measurement of different configurations' overhead costs. The "Ulysses" transmission simulates a short transmission of 275 bytes, while the two other messages simulate longer sessions by using a significantly larger amount of data: 196.83 KB and 183.86 KB, respectively. To "transmit" a message, *Alice* will write the channel contents to channel-specific text files. *Eve* will be able to read those files, simulating an eavesdropping attack; will be able to modify those files, simulating a MITM attack; and will be able to delete any portion of those files, simulating a jamming attack. *Bob* will receive the message using the available channel files, and then write the final output to a separate "received" file. This will document the state of the system at each step of its operation. However, because the system under evaluation never actually transmits data, the effects of network traffic, latency, and bandwidth can

not be observed and are not studied here. Additionally, while the three attacks will be analyzed for our system, there will be no experiments concerning the attacks themselves because their success or failure depends purely on the chosen transmission scenario, and will not change .

4.3.1 Initialization

The user starts by specifying the following parameters:

- file - file path and name of message file
- crc - mode of operation. True indicates "Wired" mode with no forward error correction, False indicates "Wireless" mode which conducts forward error correction.
- silent - Allows the adversary, *Eve*, to recalculate channel CRCs after modifying information to prevent the receiver, *Bob*, from detecting a MITM attack.

Alice and *Bob* then compute a session key using the DHKE algorithm. They each extract the top 160 bits of the shared key. The first 80 bits are the trivium key, and the next 80 bits are the trivium IV. The user defines the number of channels, the number of fragments, and the Spatial Duplication Factor (SDF). Given the specifications, the sender program generates a fragment to channel mapping, and the receiver program generates the same mapping.

4.3.2 Sending

In order to send the file, *Alice* takes each bit of the message, and assigns it a fragment identifier, based on the output of the Trivium cipher. That bit is appended onto each channel which will transmit the corresponding fragment. If a channel has a complete byte, it updates that channel's CRC with the value, and stores the output

byte in the channel output. With each byte of message input, the overall message CRC updates. Finally, the sender assigns each bit of the message CRC a fragment identifier in the same manner as the message data bits. *Alice* then writes the output to the corresponding channel files, computes the channels' CRCs or RS codes, and then appends those values to the channels' output.

4.3.3 Receiving

Bob begins by assuming each channel is valid. If a file cannot be opened, caused by a successful jamming attack, that channel is set to invalid. *Bob* then inspects each channel's contents using either the CRC error detection, or RS error correction. If that shows an error (or cannot correct it in the case of RS), that channel is marked as invalid. *Bob* then attempts to recombine the message using the given fragmentation scheme, and channel mapping. For a given bit, if there are differences between channels carrying that bit (ie. Channel 1 and 2 both carry bit X, but their contents differ), that particular fragment is flagged as being modified. After finishing the recombination process, *Bob* checks the message CRC. If the message passes that check, *Bob* knows he received the message properly. However, in the event of a failure, the protocol goes into recovery mode. If the modified fragments can be isolated to a single channel, it attempts a "smart recovery" by turning off that channel. If that succeeds, *Bob* flags the adversarially-modified channel as being insecure.

If *Bob* is unable to determine a single modified channel, he attempts to recombine the message using all combinations of the channel contents which can produce the entire message, until he discovers the affected channel. *Bob* will report whether or not he is able to successfully decode the message from the channels' contents; and, if possible, report any modified/insecure channels.

4.3.4 Adversarial Action

Eve can conduct any of the three attacks: interception, jamming, or MITM. The amount of information she gains through an interception, or eavesdropping attack depends on the fragmentation of the message. For a jamming, or Denial of Service (DoS) attack, *Eve* selects a channel(s) to jam, and simply deletes the channel file's contents. This prevents *Bob* from receiving the selected channel(s). *Eve* can conduct both a loud or silent MITM attack. For a loud attack, *Eve* selects a channel, and modifies as many bits as she desires (specified by the user), and writes the result back to the channel file. However, *Bob* will be able to detect the modification with the CRC, and can potentially recover from the modifications using the RS code, depending on the number of errors introduced. If *Eve* utilizes a silent attack, after modifying the data, she recalculates the channel's CRC or RS code. This will cause *Bob* to initially accept the channel's contents.

4.3.5 Assumptions

This protocol currently assumes that there are no timing delays or buffering between the sender and the receiver. These delays could have been added in Python, however, the information they provide is outside the scope of this research. Concepts such as packet queuing, delays, and buffering have been thoroughly researched by information theorists and network architects. Their effects would only affect the delivery of specific messages on a given channel, but that delivery is guaranteed through protocols such as TCP. Thus, each channel's data represents the entirety of that channel's communication for a given session, without any per-channel fragmentation due to Maximum Transmission Units (MTUs) of a given transmission method. This represents the data passed from the TCP packet to all higher-ordered protocols, which include security protocols such as TLS. For our system, all channels' data will be

brought together for decryption and reconstruction. The receiver, *Bob*, knows the length of the message. This ensures that the sender and receiver are synchronized. The receiver would know the length of packets on each channel, and can utilize that information to verify the length of the corresponding message. This proof of concept assumes that the channels are homogeneous - having identical characteristics for bandwidth, latency, and reliability. This work does not consider the initialization, synchronization, and tear down of the channels between the sender and the receiver. Similarly, this work does not address the use of non-uniform channels. The fragmentation of data unevenly due to unique channel characteristics is important to consider when taking a holistic view of the communication scenario including available QoS parameters and specific channel vulnerabilities, and ought to be considered in future research into the development of multi-channel communications.

4.3.6 Test Cases

A total of 12 transmission scenarios were conducted: three and ten channels; CRC and RS Code MACs; and $SDF = \{0, 1, 2\}$. Each scenario was executed for each of the three test files, giving a total of 36 tests. As previously discussed, there was no need to conduct any replicate cases. The only stochastic effect in these test cases are the specific amounts of message data transmitted on a given channel. Each channel should carry a probabalistically equal amount of data, equal to:

$$L_{ch} = \frac{1 + \mathrm{SDF}}{n}$$

However, the minor differences in channel load caused by the splitting mechanism does not affect the actual security services of a given configuration, or the ability of the system to recover from adversarial action. The test scenarios will allow us to demonstrate that the proof of concept system works as designed. *Eve* can conduct

each of the attacks; however, attack success is solely dependent on the transmission setup. We can analytically determine whether *Eve* is successful for each of the 36 test cases, for any of the three attack types. For this reason, adversarial action is not considered as a specific test case.

The three files were selected to simulate different transmission lengths. "Ulysses", which is 275 Bytes presents a small transmission, which is smaller than the MTU for a TCP packet. The other two files demonstrate the performance of the system over a sequence of several TCP packets. The traditional MTU for a TCP transmission is 1500 bytes. Thus, by transmitting $196.83KB$ for the Maroon Bells image, and $183.86KB$ for "Hamlet", it simulates the transmission of a message which would require several TCP fragments.

The selection of the number of channels reflects the benefits of a multi-channel communication scenario. Previous work utilized a two-channel system, and is described in [3]. However, the main drawback of that work was that with two channels, the user must determine which attack vector to mitigate, and cannot address all attack vectors simultaneously. It can address jamming by duplicating the entire message's contents on both channels, it can only address an interception attack by sending partial data on each channel; and it cannot provide real-time indications regarding a MITM attack. By running the system with three channels, this allows simultaneous resilience to all attack vectors by varying the amount of duplication. Thus, three channels was the first true demonstration of the system. The upper bound was set at ten channels. Theoretically, each "channel" in the channel set would be unique, whether in frequency or medium. Examples of these types of channels include wired links between destinations, 2.4GHz and 5.0GHz wireless frequencies, and Bluetooth. Thus, by setting the upper limit to 10, this includes a channel "slot" for all major communication technologies, while still providing meaningful information on the

performance of the system.

The choice of MAC reflects two independent functions: error detection and error correction. Error detection can be used when the amount of overhead needs to be constrained or pre-defined, when the cost of a re-transmission is minimal, or when the likelihood of an integrity attack is minimal. The use of the CRC defines the amount of overhead based on the number of channels utilized, while providing significant error detection. The use of a RS code allows the user to correct a defined number of transmission errors. The $(255, 245)$ RS code was selected, which requires 10 bytes of overhead for every 245 bytes of data. However, this ECC can correct errors of up to 5 bytes (in each 245 bytes of data). The overall transmission will be larger because the required overhead is a percentage of the message length, but it prevents the need to retransmit information given transmission errors on the specified channel.

The final experimental choice was varying the SDFs. The choice of SDF values was made specifically for the three channel setup, and then replicated with ten channels. With a $SDF = 0$, the system demonstrates the transmission of a single copy of the message, which mimics current transmissions. However, by fragmenting that copy of the message across the available channels, the aforementioned security services can be provided. Increasing the SDF to 1 provides the minimal amount of duplication to recover from a single channel degradation, whether adversarial or otherwise. And finally, $SDF = 2$ would indicate the full transmission of the message on each available channel for three channels, and would provide resilience to two channel degradation for the ten channel case.

4.4 Summary

This chapter presented the design decisions made in the creation of a proof of concept design. This system works to demonstrate the capabilities of a multi-channel

communication system. It also illustrates the areas where future network designers will have the most flexibility in creating a successful multi-channel communication scheme: the selection of the splitting mechanism, and the methods for error detection and correction being the most pertinent. There were several abstractions that were taken, such as not actually transmitting information across a network using several simultaneous connections. However, the use of internet connections would not give any additional benefit as the proof of concept gives no guarantee of scalable efficiency as it is currently implemented.

V. Observations and Analysis

5.1 Proof of Concept Results

This chapter will describe the insights gained through the proof of concept system, and analyze those insights with respect to the effectiveness and uses of a multi-channel protocol.

5.1.1 Overhead Data for Transmission

As previously discussed, the amount of overhead incurred through the test cases can be pre-calculated. For any system utilizing the CRC-32, there are 32 bits of overhead for each copy of the message (indicated by the SDF), and 32 bits of overhead for each channel's CRC.

$$\text{Overhead} = 4 \cdot (SDF + 1) + (4 \cdot n)$$

For scenarios using the RS Code, there are 4 bytes of overhead for the original message (CRC), and then each channel incurs a $\frac{10}{245} = 4.08\%$ overhead relative to the amount of data carried on the channel. Because each channel carries an equal amount of data (probabilistically), the total system will incur 4.08% overhead for each total copy of the message transmitted, as well as a 100% overhead for each additional copy of the message.

$$\text{Overhead} = 4 + (\text{SDF} + 1) \cdot (|M| + 4) \cdot \frac{10}{245} + |M| \cdot \text{SDF}$$

The above two equations give the amount of overhead in bytes. To convert it into a percentage, one must simply divide the result by $|M|$. The predicted results are shown in Tables 2-4.

ECC	**Number of Channels**	**SDF**		
		0	**1**	**2**
CRC	3	5.82%	107.27%	208.73%
	10	16.00%	117.45%	218.91%
RS	3	12.36%	113.82%	226.18%
	10	37.82%	139.27%	240.73%

Table 2. Ulysses Predicted Overhead

ECC	**Number of Channels**	**SDF**		
		0	**1**	**2**
CRC	3	0.01%	100.01%	200.01%
	10	0.02%	100.02%	200.02%
RS	3	4.08%	108.17%	212.25%
	10	4.09%	108.18%	212.26%

Table 3. Hamlet Predicted Overhead

ECC	**Number of Channels**	**SDF**		
		0	**1**	**2**
CRC	3	0.01%	100.01%	200.01%
	10	0.02%	100.02%	200.02%
RS	3	4.08%	108.17%	212.25%
	10	4.09%	108.18%	212.26%

Table 4. Maroon Bells Predicted Overhead

For each of the three test files, simulations were performed for all possible configurations of the following options: error detection (CRC) and error correction (RS Code); 3 channels and 10 channels; and SDF = 0, 1, and 2. The measured overhead from each test case, relative to the message size is, is shown in Tables 5-7 below.

These results are an affirmation that the predicted calculations in the amount of overhead are reliable and that the model functions as desired. In summary, the difference between the predicted overhead and the measured overhead was averaged

ECC	Number of Channels	SDF		
		0	1	2
CRC	3	6.18%	108.00%	208.73%
	10	17.45%	119.27%	220.73%
RS	3	12.73%	114.55%	226.18%
	10	39.27%	141.09%	242.55%

Table 5. Ulysses Overhead Percentages

ECC	Number of Channels	SDF		
		0	1	2
CRC	3	0.01%	100.01%	200.01%
	10	0.02%	100.03%	200.03%
RS	3	4.09%	108.17%	212.26%
	10	4.10%	108.19%	212.28%

Table 6. Hamlet Overhead Percentages

ECC	Number of Channels	SDF		
		0	1	2
CRC	3	0.01%	100.01%	200.01%
	10	0.02%	100.03%	200.03%
RS	3	4.09%	108.17%	212.26%
	10	4.12%	108.19%	212.27%

Table 7. Maroon Bells Overhead Percentages

across each of the test cases (average across the test files for a given transmission configuration), giving the results in Table 8.

ECC	Number of Channels	SDF		
		0	1	2
CRC	3	1.00	4.33	5.33
	10	3.67	7.33	9.67
RS	3	10.43	7.19	8.29
	10	25.33	22.42	21.52

Table 8. Difference Between Predicted and Measured Overhead (Bytes)

The maximum value was a 25 byte difference in total overhead, across 10 channels. This was most likely padding errors from the simulated transmission when they were written into text files. An analysis of the overhead percentage reveals a fundamental

trade-off for the end-user. When the transmission is small, relative to the given MACs, such as for "Ulysses" (shown in 2 and 5), the overhead required for MAC represents a significant percentage of the total transmission. This occurs because the CRC occupies 4 bytes for the message, and 4 bytes for each channel utilized. With no duplication, that incurred overhead is large relative to the 275 bytes of data. However, when transmissions are large, such as for "Hamlet" and "Maroon Bells", the overhead percentage for the MAC is small. When using the CRC, there are 4 bytes of overhead for the message's CRC, and an additional 4 bytes of overhead for each channel. However, when using the RS Codes, there is a fixed overhead, relative to the size of the message. The selected (255,245) RS Code indicates that there are 10 bytes of overhead for every 245 bytes of data, or a 4.08% overhead cost. The end user can choose how many bytes to allocate for error correction based on their given security needs. One can also see the effect on changing the SDF. An increase in the SDF causes the overhead to increase by 100% for each integer step of SDF because this refers to the total number of additional copies of the message which are transmitted in a given communication session. The increase in message data additionally increases the overhead of the RS Code transmissions. However, the change in SDF causes no additional change in the number of overhead bytes required for error detection.

These results support that the design decisions in this protocol give network designers the flexibility and tradespace necessary to tailor a multi-channel communication system to meet security requirements. If an end user requires minimal overhead, they can simply chose to utilize error detection, making the total incurred overhead small. However, if communication integrity is the main concern, a RS Code might indicate a better selection. This would allow the end user to select the percentage to devote to overhead, and receive a corresponding guarantee of integrity, given that a $2 \cdot t$ length RS Code can correct up to t errors. The proof of concept design illustrates

the design trade-offs that the end users can make, specifically through the analysis of the required overhead. However, this does not address the overhead required to establish, maintain, and tear down the unique network connections.

5.1.2 Attack Effectiveness

RFC 3552 clarifies the terminology for writing the security considerations for future RFCs. Additionally, it specifies the following attack environment [12]:

> We assume that the attacker has nearly complete control of the communications channel over which the end-systems communicate. This means that the attacker can read any PDU (Protocol Data Unit) on the network and undetectably remove, change, or inject forged packets onto the wire. This includes being able to generate packets that appear to be from a trusted machine. Thus, even if the end-system with which you wish to communicate is itself secure, the Internet environment provides no assurance that packets which claim to be from that system in fact are.

This section will detail the effectiveness of this protocol to mitigate the three major attacks: Interception, Jamming, and Man in the Middle, occurring on a single channel of communication. As stated above, this assumes that the attacker can access the Protocol Data Unit (PDU) of any compromised channel, if desired.

5.1.2.1 Interception

The first type of attack is an interception, or eavesdropping attack. An adversary performing this attack only intercepts the portion of the message carried on the given channel. However, the adversary does not know which bits they have intercepted, nor how many bits are missing from the complete message. This reduces the usable information that can be gained significantly. *Eve* was only able to successfully decode the message when the $SDF = n - 1$, indicating that the entire message was sent on each channel. When there was no duplication used, *Eve* only recovered partial

information. This is due to the splitting mechanism employed, and resulted in a brute force search over the missing bits, as well as the proper arrangement of the intercepted information. However, if the sender uses any duplication, and *Eve* intercepts every channel, there exists the potential to decode the message by leveraging the matching values of duplicated bits. If the user believes there is a risk for every channel to be intercepted, additional entropy might need to be added to the message transmission to prevent a successful interception.

5.1.2.2 Jamming

A jamming, or DoS attack seeks to undermine the availability of a given message. *Eve* was able to destroy the message for all scenarios which had no duplication because there was no redundant information with which to re-combine the entire message. Because *Bob* would know which channel's information was lost, as well as the data type and structure of the data, there may exist methods with which he could recover the jammed information. However, the ability to perform this recovery is outside the scope of this research. As long as the number of jammed channels was less than or equal to the SDF, *Bob* was able to successfully receive and decode the message with jamming attacks.

5.1.2.3 Man in the Middle

The most sophisticated attack is the MITM or injection. This presumes that the adversary successfully modifies the information of a given compromised channel, including its channel MAC, so that the channel's information is accepted at the destination. The goal of a MITM attack is to successfully inject erroneous data at the destination, while simultaneously intercepting the original information. The resilience against the interception aspect of the MITM attack is identical to the aforementioned

interception attack, and will not be addressed here. If *Eve* only desires to intercept information, it is not a MITM attack. Similarly, if *Eve* only wants to prevent receipt of the message entirely, it is simply jamming attack. Thus, one must assume that the adversary desires the destination to accept its injection. Traditionally, if an adversary can conduct a MITM attack, the victim only recognizes the attack when its erroneous data causes a downstream effect, or if the receiving application detects an anomaly. A multi-channel protocol can prevent this vulnerability entirely.

The main mechanism for this protection stems from the distribution of the message's MAC across the channels, and the duplication of information. *Bob* will accept the modified channel's contents because *Eve* modified the channel-specific MAC. However, the message's MAC was split at the bit level, and then distributed across the set of channels. *Eve* does not know which bit(s) of the affected channel represent that MAC. If the entire message is not on the vulnerable channel, there is no way for *Eve* to properly modify the entire MAC because some portion resides on an unaffected channel. When *Bob* attempts to re-assemble the message from the discrete portions carried on each channel, the changes will result in a failed message MAC.

Eve conducted two types of MITM attacks: a "noisy" attack, where she did not modify the channel's MAC, and a more sophisticated "silent" attack, where she recalculated the channel MAC after injecting errors/modifying information. For the "noisy" attack, *Bob* was able to detect the attack through the channel's MAC; and if using a RS Code, potentially correct the errors depending on the size of the RS Code and the amount of errors injected by *Eve*. The "silent" attack caused *Bob* to accept each channel's transmission. However, the distributed message MAC indicated the presence of erroneous information during the recombination process. At worst, the MITM attack becomes a jamming, or DoS attack because it prevents the message from successful receipt, which occurs when *Eve* compromises $x > SDF$ channels.

However, when *Eve* compromises $x \leq SDF$ channels, *Bob* was able to successfully detect the attack, identify the compromised channel(s), and recover the message without requiring retransmission by comparing the duplicate fragments to identify which channels had been modified.

5.1.2.4 Perfect Detection

Dolev et al. posited the existence of a communication system which can be both perfectly secure and perfectly resilient. However, it relied on a multi-round approach which allowed the sender and receiver to identify vulnerable communication links, and remove those links from the set of available channels. Thus, their system was only resilient **if** there were enough secure channels.

In practice, there is no way to guarantee enough channels to provide perfect resilience to an end-user. But what if there was a related metric - perfect detection - which guarantees the sender and the receiver will detect adversarial action. This term applies only to the MITM or injection or jamming[1] attacks because, for most channel mediums, there is no reliable mechanism to detect whether or not an adversary has eavesdropped on a transmission unless the eavesdropping affects the signal itself, which would then be considered an injection attack. Provided *Bob* was expecting to receive a message, a jamming attack is always detectable - the absence of a transmission on a given channel, or a transmission arrives in such a condition that it is completely unusable. However, successful MITM or injection attacks can easily go undetected.

The proposed multi-channel protocol provides this detection guarantee, even if the adversary has compromised the entire set of channels in use. An adversary cannot conduct a successful MITM attack, under any of the transmission scenarios, because

[1]There is no way to distinguish between channel degradation due to non-adversarial means and an attack, but they have identical attacks and would most likely require the same form of response

the system distributes the message MAC across the available channels. When the receiver attempts to re-combine the message, and it fails its integrity check, the receiver can verify the fragments on different channels to determine which fragments had been modified. In the best case, the receiver will report which channel was compromised, and recover the original message **without retransmission**. In the worst case, consider an adversary which modifies the contents of every available channel. Without the splitting mechanism key, this adversary still has no knowledge of how the message bits were fragmented, to include the message MAC. Thus, any changes they make will be detected when the receiver recombines the message. If each channel is modified, the effects will be no worse than a jamming adversary on each channel.

5.1.2.5 Attack Summary

This section described the system's resilience to each of the main attack vectors. By looking at the conditions for a "successful" attack, a pattern emerges. Let x be the number of channels affected by Eve. If $x < SDF$ channels have been affected, the receiver will successfully detect any attack without the need for a retransmission. For a MITM attack affecting $x < SDF$ channels, the receiver can report which channels have been affected. Additionally, if $x \geq SDF$ channels have been affected, the transmission will fail. However, the system will know that its transmission has failed. This presents a meaningful increase in the guarantees of confidentiality, integrity, and availability for the end user.

5.2 Proof of Concept Insights

5.2.1 Implementation Challenges

The first, and most glaring challenge in the implementation of a multi-channel communication architecture is how to define the communication channels between the

sender and the receiver. We addressed this challenge by utilizing a pre-defined channel set; however, this limits the system's flexibility. Similarly, given that the splitting mechanism relies upon the communicated session parameters, the communication of those parameters must use existing key exchange mechanisms over one or multiple channels in the channel set, or be communicated out of band. How to communicate these parameters is a design choice for the end user based on their specific security needs. There must also be dynamic operation/channel configuration to address the possibility that a given channel can become unresponsive or experience degraded performance. These primary challenges would likely be resolved as protocols such as TLS develop support for multi-channel systems.

The system must also address the issue of error detection and correction. Given a cryptographically secure splitting mechanism, if a bit is lost (and has not been recovered using forward error correction), the joining mechanism will be unable to piece together the final message without knowing which particular bit was lost. Thus, there must be error detection and/or correction utilized for each channel's information. Our system design relies on the use of CRC and RS code systems for error detection and correction, respectively.

Another challenge that needs to be addressed is how each particular component addresses the security configuration of the system. There exist relationships between the resilience of a given security scheme, the overhead required to implement that scheme, and an inverse relationship between integrity and confidentiality. As the amount of overhead increases, and more information is duplicated, the system achieves better integrity and confidentiality. The user does not directly specify a confidentiality and integrity requirement, but rather attains those services through the selection of error correction and the amount of duplication.

5.2.2 Timing and Storage Complexity

The operation of the designed system is broken into the following parts: system initialization and key exchange, fragmenting and sending, transit, and finally receipt/recombination.

The most computationally intensive portion of the entire system is the key exchange, which utilizes the DHKE mechanism. The key exchange's security relies on the discrete logarithm problem, and is discussed in detail in [25]. *Yakymenko et al.* in [26] proves that the temporal complexity of modular exponentiation, required for the computation of a DHKE key, is $O(b \cdot \ln^3 b)$, where b is the size of the modulus p from the DHKE in bits.

To fragment and send the message, it requires $\log_2(n)$ operations to generate a fragment for a message bit, and that must be performed $|M| = m$ times. As the system processes each byte, it computes the error correcting code for the message, requiring $O(1)$ operations. After fragmenting the entire message, each channel computes its own, per-channel ECC, which occurs in $O(\frac{m}{8}) = O(m)$ time. Thus, this portion of the system operates in $O(m \cdot \log_2(n))$ time. However, in practice the number of channels will always be much less than the size of the message in bits, so this becomes $O(m)$. This also accounts for the inclusion of the message and channel error correcting codes The transmission of the message is not included in timing analysis because it depends purely on the transmission time and end-to-end delay of a communication link/interface.

Receiving and recombining the message operates similarly to sending, with the exception of when the system must recover an adversarially modified message. If each channel transmits securely, the recombination is $O(m \cdot \log_2(n)) = O(m)$. However, the recombination of a modified transmission depends on the allowed attempts. The system can attempt to recover errors on x channels, where $0 \leq x \leq n - 1$, because

there must be at least one correct transmission. Each recombination required m operations, and in the worst case it would require the full nested structure. Thus, the recombination would occur in $O(m^x)$.

The storage needs of a system in operation depends primarily on how much duplication the user requires. We selected the Trivium cipher purely because its implementation is small in hardware, requiring only 180 bits of state. The operation of this system requires the ability to recombine the message from some of its parts (if there are errors), which indicates the requirement to store each channel's contents in a buffer. Because each channel could potentially carry the entire message, these buffers would need to be $\frac{m \cdot n}{8}$ bytes. Again considering the fact that $m >> n$, the storage requirements in practice are $O(m)$, and thus reasonable for use.

5.3 Discussion

5.3.1 Implementation in Existing Architecture

One of the main questions regarding this system is how it would interact with the existing TCP/Internet Protocol (IP). There is an argument that could be made for including it as a form of a session layer protocol because it relies on several available channels, which would each consist of their own TCP/UDP connection. However, the transport layer may also be the natural place for this idea to reside. Consider that TLS was specifically designed to be flexible to allow for a changing cyber security paradigm. This protocol could provide the catalyst for the development of multi-channel communication systems, which can achieve security even if attacks outpace the development of suitable encryption schemes. Thus, there is a reasonable likelihood of TLS or another protocol adopting this approach while developing the support for a multi-channel security system.

The proposed system demonstrates a solution that would work with a protocol

that allows for the creation and synchronization of multiple channels. Given a set of channels between the client and the server, this system would be able to function as a cipher suite for TLS. Relying on existing nomenclature, the user would simply need to specify the method of key exchange, such as the following examples

- TLS_RSA - public and private keys
- TLS_DH - Diffie-Hellman
- TLS_DHE - ephemeral Diffie-Hellman
- TLS_ECDH - elliptic-curve Diffie-Hellman

The user also needs to specify which MAC to use. TLS currently utilizes Hash-based Message Authentication Codes (HMACs) for stream cipher use, where HMACs are a special form of MAC that provide for both message integrity and authenticity. These are currently denoted as follows

- HMAC-MD5
- HMAC-SHA1
- HMAC-SHA256/384

To match the existing TLS nomenclature, the MACs we utilized would be denoted as either CRC-32 or RS. This only leaves the session parameters for the fragmentation factor and duplication level, based on the number of channels. Thus, the client can offer the following items:

- FF-X - The # of fragments the message will be broken into
- DF-X - The duplication level for the session

By modifying the proposed multi-channel system as a cipher suite for a future version of TLS, the following would be used to communicate the session parameters, where $X \leq n$, and specified by the client at system initialization.

- TLS_DH_CRC-32_FF-X_DF-X
- TLS_DH_RS_FF-X_DF-X

The concept of a "channel" has been intentionally vague throughout this work. For optimal security, each channel would consist of physically distinct communication mediums. This would require an adversary to address each channel's physical characteristics individually, making attacks significantly more challenging. Each physical medium requires different transmission capabilities, and offers different performance with respect to transmission time and other delays (I.e. a fiber optic connection has significantly different performance than a wireless, 2.4GHz transmission). The same security services could be provided by instantiating unique TCP connections for each channel. While each transmission would stem from the same transmitter, each connection will take different routes to the destination, and have a unique port on the destination device. If an adversary did not know specifically which ports contain the transmission, this still represents a non-trivial task to gain any meaningful information. As previously discussed, even if the adversary has access to each channel, re-assembling the fragmented message into meaningful data presents a computational challenge for the adversary, maintaining the security requirements for the end user. This would allow systems to utilize the security provided by multi-channel data fragmentation, even if they only have access to a single channel.

5.3.2 Implementation in Critical Infrastructure

The encryption/authentication methods utilized in this proof of concept system would not meet the current security requirements for widespread implementation

in critical infrastructure communications. However, it demonstrates several of the concepts that need to be thought through and solved for its implementation. This solution demonstrates the benefits to utilizing multiple channels, specifically when the data is split at the bit level in a non-predictable/pseudorandom way. Assuming there exists an adversary that can break the encryption of a given channel, they do not know which bits they have decrypted, nor how those bits fit into the overall message. Thus, an adversary would require the interception/compromise of several channels in the channel set to gain any meaningful information. This supports the confidentiality requirement specified in [12] for network communications. Similarly, because the message's MAC is distributed across channels, the system can detect changes to a particular channel, even if the adversary modifies the information in such a way that it passes the channel specific MAC. This supports the integrity requirement specified in [12].

The framework can thus be applied to existing critical infrastructure communication, where the primary concerns are message integrity and confidentiality, as discussed in Section 2.6. By fragmenting that data across multiple channels for infrastructure communications, the required adversarial action to break the system increases considerably, because they need to intercept/modify multiple channels; and even with several channels at their disposal, still need to break the data splitting mechanism at the endpoints. If an end user determines that the Trivium cipher does not provide an adequate level of security to the splitting mechanism, it can be replaced with a more secure alternative. The use of encryption can be applied to the the original message, the individual channels, or both, depending on the user's tolerance for overhead and timing.

5.4 Summary

This section outlined the insights into the operation of a multi-channel communication system through the use of a proof of concept system. The proof of concept system did not utilize networked transmissions, but simulated those transmissions by using unique files to represent the source, destination, and channel transmissions. This demonstrated the amount of incurred overhead for either error detection or error correction modes, as well as the effect on the number of channels and the number of unique message fragments. The selection between error detection and error correction gives the user the choice between a constant overhead cost and an overhead cost relative to the size of the transmitted message. Additionally, the use of duplication demonstrated resilience to multiple attack vectors. As long as the entire message is not transmitted on an individual channel, this system can provide security against interception/eavesdropping attacks; however, if every channel can be simultaneously intercepted, additional entropy might be needed to provide the same guarantees for communication security.

VI. Conclusion

6.1 Overview

This thesis investigated the security of a multi-channel communication system which relies on data fragmentation at the bit level to provide security services to the end user. It presented terminology to describe multi-channel communications, and applied it to a theoretical system. A proof of concept system was prototyped to demonstrate its operation. This proof of concept system allowed analysis of the implementation challenges related to building such a system, and provided possible solutions. Areas where the user can make design decisions based on their specific use case were identified, as well as the trade-offs between the cost incurred and the services gained. This demonstrated the tunable nature of a multi-channel communication protocol, and the resilience it provides to adversarial action. Finally, current security systems were evaluated to determine where this architecture would fit within existing network paradigms.

6.2 Summary

Chapter II of this thesis laid the groundwork for the proposed architecture. Among the topics that were detailed are the CIA Triad, emphasizing the traditional security needs which the proposed architecture must maintain in order to be adopted for widespread use. The next topic was the TLS protocol, which is widely used to secure network transmissions. Throughout its lifecycle, the architects of TLS have demonstrated a willingness to adapt, as evidenced by its pattern of updates to meet changing cryptographic and security needs. It is a natural extension, then, that such a protocol can adapt to support multi-channel communications. Correspondingly, the TLS protocol's use of cipher suites presents a natural fit for the proposed security

protocol. Chapter II addressed the three main types of network attacks: interception/eavesdropping, jamming, and MITM. Several mechanisms, such as the DHKE algorithm, the use of LFSRs, and the error detection and correction provided by CRC and RS codes were described for future use in the proof of concept design. The final section of pertinent background information involved the regulatory standards requiring network security for all aspects of the CIA Triad.

Chapter III established the terminology and theory for a multi-channel communication protocol. This theory was applied to a hypothetical multi-channel system to establish such a system's benefits and uses. The primary mechanism for this protocol is connecting each data bit to a corresponding fragment identifier, and then assigning these fragments to specific channels. This allows the communication system to distribute the message contents, as well as its MAC across the available channels. By distributing that information across the available channels, such a system provides resilience for the user against the three main attack vectors. However, this resilience comes at the cost of additional transmitted information in the form of MACs and duplicated data.

Chapter IV described the development of the proof of concept system. Specifically, each step in the proof of concept's operation is outlined from session initialization through message receipt. It describes some of the specific design choices to improve the proof of concept's efficiency This establishes the design decisions made for the system, and highlights the flexibility a multi-channel communication system could provide the end user.

Chapter V analyzed the operation of the proof of concept system. It details the overhead incurred through the design decisions, as well as the associated resilience to adversarial action. This highlights the design trade-offs available to the network designer, illustrating the tunability of such a system to meet specific needs. Following

the design trade-offs, the lessons learned through the proof of concept design are described in two specific areas. The first area is the challenges in this protocol's implementation. These challenges must be solved prior to widespread adoption of such a system. Most of these challenges were abstracted in the proof of concept system, such as the use of pre-established "channels", and not negotiating a specific channel set upon initialization of a communication session. Analysis of the timing and storage complexity of the proof of concept system suggested that the basic operation is realistic relative to the size of the message's length. Suggestions for how this system could be implemented in current architectures were discussed, specifically as a cipher suite for a future version of TLS which supports multi-channel communications, as well as a specific sector - critical infrastructure - which could benefit from the use of a multi-channel communication system. Chapter V concludes with a discussion of the future work required to bring this architecture to fruition.

6.3 Research Contributions

This work classifies a novel communication architecture which can achieve security without relying on traditional encryption methods. Instead, it relies on bit level fragmentation and distribution of data across multiple, unique communication channels. This system can provide simultaneous resilience to jamming, eavesdropping, and MITM attacks. This system duplicates message data to prevent a single channel from becoming a single point of failure (jamming). It fragments message data at the bit level using a cryptographically secure splitting mechanism to prevent any channel from carrying meaningful data (eavesdropping). And finally, by both fragmenting and duplicating information across the set of channels, it can detect sophisticated MITM attacks, report the affected channel(s), and potentially recover the message without requiring retransmission.

This thesis also proposed a specific network layer where this architecture could be reasonably implemented within current security systems, facilitating the adoption of this protocol with minimal changes to existing infrastructure. The most appealing candidate for this implementation is the popular TLS protocol, which operates over TCP connections. Finally, this work evaluated and identified several specific developments/challenges that must be solved in order to field a multi-channel communication system.

6.4 Future Work

To determine if this proposed system is truly viable, it must be tested in an environment where performance can be measured relative to other secure network paradigms. Future work in this area should therefore include full system development, both as part of the existing TLS protocol, and perhaps as a custom hardware solution, and verify that the communication sessions can be established, maintained, and used effectively with normal internet traffic/communications. This would address the buffering/timing delay challenges. Those challenges have been comprehensively analyzed for traditional network communications, but would need modification to apply to this framework, as the total network traffic increases in a non-linear fashion as the number of channels increases.

A major benefit of a multi-channel system is the security services gained through its use. However, the resilience to attacks was based purely on analysis of the information present on a given channel. The effectiveness of an eavesdropping attack specifically was outside of the scope of this thesis. Future work would need to conduct a security analysis of the multi-channel system for each major type of attack. This evaluation would rely on information theory to quantify how much information an adversary gains by eavesdropping a channel carrying non-related, non-continuous

portions of the original message. Additionally, this work would generate relationships between the amount of incurred overhead, and the levels of Confidentiality, Integrity, and Availability provided. It would also allow for more direct comparison between a multi-channel system and traditional security models to allow network architects to determine if the multi-channel system is worth the additional investment. Finally, this would allow a user to define the security of their system given user-specific run-time decisions.

6.5 Conclusion

This thesis contends that security protocols should develop support for multi-channel communications because of the security services multiple channels provide. The proposed multi-channel communication protocol relies on data fragmentation at the bit level to secure its transmissions, and demonstrates the resilience gained through its use. A proof of concept system provided insights into valuable services, such as the ability to detect and defeat a MITM attack, without re-transmission, and report which channel was compromised. It illustrated both the system's resilience to adversarial action, and the system's tunable nature. This system utilized existing mechanisms for the key exchange that communicated initialization information; the splitting mechanism that mapped data to a specific channel; and the ECCs that provided error detection and correction. This suggests that a multi-channel protocol can be created from existing systems. The proof of concept also illuminated several of the remaining challenges for such a protocol to achieve widespread implementation, such as how to securely determine the channel set prior to a communication session's initialization.

This work serves as a starting point for security architects to develop a secure, multi-channel communication system, highlighting several key challenges. This frame-

work would fit into future implementations of TLS or similar protocols developed to support multi-channel communications. By increasing the number of channels utilized, a communication system can be tailored to the specific needs of an end-user, gaining a corresponding increase to all aspects of the CIA triad, even in the presence of adversarial action. Ultimately, a multi-channel communication system promises to increase the standard required for an attacker to gain meaningful information (Confidentiality), to modify the transmitted information (Integrity), or to prevent the customer from its use (Availability).

Bibliography

1. J. Nechvatal, E. Barker, L. Bassham, W. Burr, M. Dworkin, J. Foti, and E. Roback, "Report on the Development of the Advanced Encryption Standard (AES)," *Journal of Research of the National Institute of Standards and Technology*, vol. 106, no. 3, pp. 511–577, 2001.

2. K. Moriarty, B. Kaliski, J. Jonsson, and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2," Internet Requests for Comments, RFC 8017, November 2016.

3. C. Wolfe, S. Graham, R. Mills, S. Nykl, and P. Simon, "Securing Data in Power-Limited Sensor Networks Using Two-Channel Communications," in *Critical Infrastructure Protection*, vol. 12, 2018, pp. 81–90.

4. T. Liston and E. Skoudis, *Counter Hack Reloaded: A Step-by-Step Guide to Computer Attacks and Effective Defenses*, 2nd ed. Prentice Hall, 2005.

5. C. Paar and J. Pelzl, *Understanding Cryptography.* Springer, 2010.

6. S. Lindskog, A. Brunstrom, R. Lundin, and Z. Faigl, "A Conceptual Model of Tunable Security Services," in *International Symposium on Wireless Communication Systems*, 2006, pp. 530 – 534.

7. D. Dolev, C. Dwork, O. Waarts, and M. Yung, "Perfectly Secure Message Transmission," *Journal of the ACM*, vol. 40, no. 1, pp. 17–47, 1993.

8. T. Dierks and C. Allen, "The TLS Protocol Version 1.0," Internet Requests for Comments, RFC 2246, January 1999.

9. R. Barnes, M. Thomson, A. Pironti, and A. Langley, "Deprecating Secure Sockets Layer Version 3.0," Internet Requests for Comments, RFC 7568, June 2015.

10. T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," Internet Requests for Comments, RFC 4346, April 2006.

11. ——, "The Transport Layer Security (TLS) Protocol Version 1.2," Internet Requests for Comments, RFC 5246, August 2008.

12. E. Rescorla and B. Korver, "Guidelines for Writing RFC Text on Security Considerations," Internet Requests for Comments, BCP 72, July 2003.

13. E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," Internet Requests for Comments, RFC 8446, August 2018.

14. T. Maxino and P. Koopman, "The Effectiveness of Checksums for Embedded Control Networks," *IEEE Transactions on Dependable and Secure Computing*, vol. 6, no. 1, pp. 59–72, jan 2009. [Online]. Available: http://ieeexplore.ieee.org/document/4358707/

15. P. Koopman and T. Chakravarty, "Cyclic redundancy code (CRC) polynomial selection for embedded networks," pp. 145–154, 2004.

16. P. Koopman, "Best CRC Polynomials." [Online]. Available: https://users.ece.cmu.edu/~koopman/crc/index.html

17. G. Solomon and I. S. Reed, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960. [Online]. Available: https://doi.org/10.1137/0108018

18. M. Riley and I. Richardson, "Reed-Solomon Codes." [Online]. Available: https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html [Accessed: 4 October 2019]

19. C. De Canniere and P. Bart, "Trivium Specifications," *eSTREAM, ECRYPT Stream Cipher Project*, 2006.

20. D. Rama, K. Reddy, D. Hemalatha, and A. Mubeen, "A Study on Quality of Service for Computer Networks," Tech. Rep., 2014. [Online]. Available: www.iosrjournals.org

21. Q. Duan, "Modeling and analysis of end-to-end Quality of Service provisioning in virtualization-based future internet," in *Proceedings - International Conference on Computer Communications and Networks, ICCCN*, 2010.

22. United States Mandatory Standards Subject to Enforcement. [Online]. Available: https://www.nerc.com/pa/stand/Pages/ReliabilityStandardsUnitedStates.aspx [Accessed: 2019-11-14]

23. S. Fries and R. Falk, "Ensuring Secure Communication in Critical Infrastructures," in *International Conference on Smart Grids, Green Communications, and IT Energy-aware Technologies*, June 2016, pp. 15–20.

24. S. D. Bao, M. Chen, and G. Z. Yang, "A Method of Signal Scrambling to Secure Data Storage for Healthcare Applications," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 6, pp. 1487–1494, 2017.

25. I. F. Blake and T. Garefalakis, "On the Complexity of the Discrete Logarithm and Diffie-Hellman Problems," *Journal of Complexity*, vol. 20, pp. 148–170, 2004.

26. I. Z. Yakymenko, M. M. Kasianchuk, S. V. Ivasiev, A. M. Melnyk, and Y. M. Nykolaichuk, "Realization of RSA Cryptographic Algorithm Based on Vector-Module Method of Modular Exponentiation," in *14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET*. IEEE, 2018, pp. 550–554.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

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 this 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 any 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)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
26–03–2020	Master's Thesis	Sept 2018 — Mar 2020

4. TITLE AND SUBTITLE

Multi-Channel Security through Data Fragmentation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Hayden, Micah J., 2d Lt, USAF

5d. PROJECT NUMBER

20G202A

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Air Force Institute of Technology
Graduate School of Engineering an Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-MS-20-M-026

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis presents a novel security system developed for a multi-channel communication architecture, which achieves security by distributing the message and its associated message authentication code across the available channels at the bit level, to support systems that require protection from confidentiality and integrity attacks without relying solely on traditional encryption. One contribution of the work is to establish some helpful terminology and present a basic theory for multi-channel communications. This proof of concept, focused on the splitting and recombination activities, operates by using existing key exchange mechanisms to establish system initialization information, and then splitting the message in fragments across each available channel. Splitting prevents the entirety of a given message from being transmitted across a single channel, and spreads the overall message authentication across the set of channels. This gives the end user the following unique service: the sender and receiver can identify a compromised channel, even in the presence of a sophisticated man in the middle attack wherein the adversary achieves fragment acceptance at the destination by altering the message's error detecting code. Under some conditions, the receiver can recover the original message without retransmission, despite these injected errors. This system would be a natural fit as a cipher suite for a future iteration of the Transport Layer Security protocol targeting support for multi-channel communication systems.

15. SUBJECT TERMS

Multi-Channel Communication, Data Security, Data Fragmentation, Communication Security

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Scott Graham, AFIT/ENG
U	U	U	UU	94	19b. TELEPHONE NUMBER *(include area code)* (937) 255-6565 x4581; scott.graham@afit.edu