Development of a Security-Focused Multi-Channel Communication Protocol and Associated Quality of Secure Service (QoSS) Metrics

Paul M. Simon

DEVELOPMENT OF A SECURITY-FOCUSED MULTI-CHANNEL COMMUNICATION PROTOCOL AND ASSOCIATED QUALITY OF SECURE SERVICE (QoSS) METRICS

DISSERTATION

Paul M. Simon, Civilian

AFIT-ENG-DS-22-S-038

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-DS-22-S-038

DEVELOPMENT OF A SECURITY-FOCUSED MULTI-CHANNEL COMMUNICATION PROTOCOL AND ASSOCIATED QUALITY OF SECURE SERVICE (QOSS) METRICS

DISSERTATION

Presented to the Faculty
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 Doctor of Philosophy in Electrical Engineering

Paul M. Simon, B.S., M.S.
Civilian

September 2022

AFIT-ENG-DS-22-S-038

DEVELOPMENT OF A SECURITY-FOCUSED MULTI-CHANNEL COMMUNICATION PROTOCOL AND ASSOCIATED QUALITY OF SECURE SERVICE (QOSS) METRICS

DISSERTATION

Paul M. Simon, B.S., M.S.
Civilian

Committee Membership:

Scott R. Graham, Ph.D.
Chair

Robert F. Mills, Ph.D.
Member

Michael R. Grimaila, Ph.D., CISM, CISSP
Member

AFIT-ENG-DS-22-S-038

Abstract

The threat of eavesdropping, and the challenge of recognizing and correcting for corrupted or suppressed information in communication systems is a consistent challenge. Effectively managing protection mechanisms requires an ability to accurately gauge the likelihood or severity of a threat, and adapt the security features available in a system to mitigate the threat. This research focuses on the design and development of a security-focused communication protocol at the session-layer based on a re-prioritized communication architecture model and associated metrics. From a probabilistic model that considers data leakage and data corruption as surrogates for breaches of confidentiality and integrity, a set of metrics allows the direct and repeatable quantification of the security available in single- or multi-channel networks. The quantification of security is based directly upon the probabilities that adversarial listeners and malicious disruptors are able to gain access to or change the original message. Fragmenting data across multiple channels demonstrates potential improvements to confidentiality, while duplication improves the integrity of the data against disruptions. Finally, the model and metrics are exercised in simulation. The ultimate goal is to minimize the amount of useful information available to adversaries.

AFIT-ENG-DS-22-S-038

I dedicate my dissertation work to teachers everywhere, old or young, formal or informal, intentional or unexpected. Indeed, there is always something new to learn. With knowledge comes power, and when knowledge is shared, so too is the power. If only more people were open to teaching, as well as learning, unconditionally, our collective power would be unlimited.

Acknowledgements

I am most grateful to my research advisor, Dr. Scott Graham, for his guidance and patience. His broad perspective on the subject and attention to detail helped point me in the right direction during this long and winding road that was my research journey.

To my research committee, Dr. Robert Mills and Dr. Michael Grimaila, who have been valuable mentors during my time as a Ph.D student. Interactions may have been limited, but that directly reflects the power of their influence.

To my children, whose smiles and hugs consistently provided all the motivation one could ever need.

To my astonishing wife, whose love and support (and occasional ridicule) kept me going through the most challenging of times. Without her constant encouragement, this research would not have been possible.

To my parents, whose unconditional love encouraged this lifetime of curiosity and exploration. They taught me to never give up.

Paul M. Simon

Table of Contents

Page

List of Figures

Figure Page

List of Tables

List of Acronyms

2FA	Two Factor Authentication
AL	Average Channel Loading
ASCII	American Standard Code for Information Interchange
BMC	Baseband Management Controller
CIA	Confidentiality-Integrity-Availability
CRC	Cyclic Redundancy Check
DDoS	Distributed Denial of Service
DF	Duplication Factor
DoS	Denial of Service
DTMF	Dual Tone Multiple Frequency
IP	Internet Protocol
IT	Information Technology
MIMO	Multiple-Input Multiple-Output
MTU	Maximum Transmission Unit
NED	NEtwork Description
OFDM	Orthogonal Frequency Division Multiplexing
OSI	Open Systems Interconnection
PLC	Programmable Logic Controller
PSMT	Perfectly Secure Message Transmission
PSTN	Public Switched Telephone Network
QoS	Quality of Service
QoSS	Quality of Secure Service
RAID	Redundant Array of Inexpensive Disks
SCADA	Supervisory Control and Data Acquisition

SNR	Signal to Noise Ratio
SS7	Signaling System Number 7
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol

DEVELOPMENT OF A SECURITY-FOCUSED MULTI-CHANNEL COMMUNICATION PROTOCOL AND ASSOCIATED QUALITY OF SECURE SERVICE (QOSS) METRICS

I. Introduction

1.1 Overview

The mitigation of operational risk when using communication systems is a critically important issue due to threats such as eavesdropping and the challenge of recognizing and correcting for corrupted or suppressed information. Hand-written notes were easily read or tampered with. To counteract that, written notes were sealed in scrolls or envelopes, first with wax, then with glue. Other methods of protecting written notes were also developed, for example the Caesar cipher. The advent of analog and digital communications transmitted via wire or radio frequency increased speed and changed the manner of eavesdropping or corrupting messages. Nonetheless, the threat of eavesdropping and corrupted or suppressed messages continues to exist in communication systems. In addition to noise, these systems may be affected by any combination of three malicious attack vectors: Denial of Service (DoS), data injection, or eavesdropping. As technology evolves, new forms of protection mechanisms are created, but may not always be applied to their fullest potential. Effectively managing these protection mechanisms requires an ability to accurately gauge the likelihood or severity of the threat, and adapt the security features available in the system to meet and exceed the threat. The intent of this research effort is the design of a security-focused communication protocol based on a re-prioritized communication architecture model and associated metrics. The model and metrics is exercised in a simulation environment, illuminating the path forward to complete the design and implementation. Ultimately, this security-focused protocol will improve or maintain a high-level of the traditional Information Technology (IT) qualities of confidentiality, integrity, and availability for that communication system.

1.2 Problem Statement

All communication system managers want their communication systems to be trustworthy. All users of modern communications systems want their messages to be free from eavesdropping or disruption. Therefore, anyone who manages or uses communications systems would be interested in a method to provably improve security. In broad terms, this effort strives to improve the security of communications systems, specifically, the confidentiality and integrity of messages (at the cost of complexity and redundancy). Communication system managers typically focus only on system availability. Confidentiality is typically achieved through some form of encryption. Integrity is pursued through the tools of availability, specifically by increasing signal power to maximize Signal to Noise Ratio (SNR), or with error detection and forward error correction codes. These techniques, although valuable, may be inadequate against a sophisticated and motivated adversary.

The intent to improve overall security is built upon existing models and protocols and possibly rearranging architectural components. To demonstrate improved security, reliable and repeatable metrics for security are needed for an accurate assessment of proposed improvements. The primary goal is to defend against adversarial listeners or malicious disruptors.

While various technological improvements have helped, the challenge of maintaining the security of messages persists. The threat of eavesdroppers and the possibility of maliciously corrupted or suppressed information are valid concerns, with no clear manner for consistently and reliably quantifying security claims in communications systems. Absent a clear set of metrics, any conversation about security ends up being an apples to oranges comparison. Therefore, in an effort to provide provable improvement to the security of communications systems, a set of metrics that quantifies the security in an objective and repeatable manner must be developed in conjunction with an architecture that embraces and interacts with those metrics. The architecture may then be improved where the metrics show deficiencies, and the metrics may be used to demonstrate the iterative improvements to the architecture. A novel contribution from this research is the development of a set of such security metrics. The architectural models and protocols build off of or use existing technologies as inspiration. The resulting metrics provide an objective view into quantifying security in repeatable terms.

1.3 Research Objectives

Understanding the challenges in measuring and maintaining the security of communications systems, the research questions motivating this work are:

- To what degree can we quantitatively determine the available or existing security of a communication network in an reliable and repeatable manner?
- What are the shortfalls of metrics?
- What are the essential elements of a communication protocol that enforces or ensures a desired or required level of security, and where it logically resides in the communication stack?
- What are the benefits of dynamically adjusting network configurations based on feedback mechanisms?

These questions will serve as a guide to research, develop, and analyze a security-focused session layer communication protocol.

1.4 Operational Motivation

Operational motivations for implementing a new communication architecture and protocol include the desire to actively adjust the security of communications in a dynamic threat environment. It may just be ensuring a baseline level of security in and around a normal environment, but might be used to better understand the security in a contested environment. Another operational motivation is to ensure that messages are received without any errors or interference of any kind, as is being able to switch security policy on-the-fly when environmental characteristics change. Finally, the ability to quantify security metrics in a reliable and repeatable manner will allow network managers and end-users to know the level of security available within their systems and to be able to accurately compare the performance across multiple systems.

1.4.1 Protection Against Known Attack Vectors

Every day, new vulnerabilities affecting computer software, protocols, networks, and infrastructure are published in the MITRE Corporation's Cyber Vulnerability Enumeration (CVE) database [2]. Many of these vulnerabilities are caused by hand-offs between protocols, poorly designed software or operating systems, or unexpected gaps in network security measures. Despite the root cause of the vulnerabilities, they provide attack vectors or entry points to malicious actors. The broad categories of attack vectors include DoS attacks, data injection, or eavesdropping. A DoS attack may involve cutting a wire or overpowering a particular frequency (jamming). A data injection, or spoofing attack, involves an adversary sending fabricated data that takes the place of actual data. Finally, and the most difficult to discover, is an eavesdropping attack, which involves an adversary intercepting and extracting useful information from the communication system. Unfortunately, many current communication security mechanisms are applied after initial design of the system is complete. These hastily applied or bolted-on security mechanisms tend to be inadequate. Built-in security mechanisms tend to fare much better. Therefore, a security-focused protocol designed to directly address these broad classes of attack vectors running on a dynamically adjustable architecture is expected to thwart these specific malicious actors.

1.4.2 Protection Against Future Attack Vectors

Knowing and preparing for existing attack vectors is relatively easy. Preparing for future attack vectors is exceptionally difficult and requires consideration of the state of the possible. Specific versions of the known categories of attack vectors are difficult to speculate, but they still may be exploited. However, based on the current technology, wireless and near-field interfaces may be more commonly exploited as they become more prevalent. Operationally, to be able to improve or maintain the security requires considering future attack vectors, as does the ability to quantify performance. Existing protection techniques and metrics may not fully capture unanticipated attacks, but the foundation will exist. Having the ability to grow and modify, the protocol and metrics will be able to adjust to unexpected infiltration.

1.5 Technical Motivation

The technical motivation behind implementing a security-focused multi-channel communication protocol is the ability to monitor the metrics, compare them to the desired or specified security posture, and to potentially make improvements to the performance or handling of the data. The technical motivation also encourages technology to be adapted and modified for broader acceptance.

One approach to improve a system is to incorporate redundancy such that failure of one portion of the system does not cause complete failure. Of course, that comes at a cost. There is also the possibility that redundancy also negatively affects other attributes of the system.

Having better performance everywhere is the ultimate goal. In this instance, the improvement is through redundancy, although being able to accurately measure current performance and pivot to an improved posture is equally valid. Unfortunately, the environment or adversarial interference can get in the way. Therefore, better should never be the enemy of perfect. Some improvement may be all that is needed to meet the desired or required performance goals.

1.5.1 Conceptual Gaps

The conceptual gaps in this research are areas where new research will be required in order to create a good solution to the research problem. They are associated with research risk. The primary conceptual gap for this research, which accurately reflects one of the primary risks, is the development of a series of metrics meant to quantify and improve security. This gap corresponds to understanding the degree to which we can accurately and reliably measure the available or existing security in a communication system. Unlike other metrics used in network architectures, e.g., data or bit rate, jitter, bandwidth, transmission frequency, or power, security does not have a direct measurable aspect. A series of security metrics could be misleading or completely wrong. Therefore, with thorough research and analysis into possible surrogate characteristics, a set of metrics may provide insight into the performance of networks in real-world scenarios.

1.5.2 Technical Gaps

Technical gaps in this research are areas where a technical solution would be needed to perform the research. This includes developing the architecture, designing the protocol, specifying the feedback mechanism, and programming the simulation environment. These are all feasible design and development tasks which would actively generate or consume data quantified by the metrics. The architecture would adjust configuration based on the metrics. The protocol would generate and consume data about the performance of packets transmitted through the architecture. The feedback mechanism would deliver the metrics from one end of the architecture to the other. Finally, the simulation environment would demonstrate them working together. How all these building blocks fit together would be determined throughout the research.

1.6 Hypothesis

The hypothesis is that, with thorough development and refinement, this security-focused session layer protocol can increase the security of communication systems as reflected by the corresponding metrics. This hypothesis is evaluated using the set of security metrics, observing the degree to which the security can be improved in a reliable and repeatable manner.

1.7 Approach

To answer the research questions, the development of a secure communications protocol follows a three-step development process:

1. develop a model communication architecture that utilizes the security protocol;
2. develop a set of metrics used to quantify the security in the communication

architecture; and

3. develop a simulation environment to exercise the architecture and capture the metrics.

As the protocol is exercised within the simulation environment, the model and the metrics are updated as necessary to ensure the protocol is as robust as possible.

1.8 Organization

The remainder of this document is organized as follows. Chapter II provides relevant background information on topics utilized for this research to include the security triad, attack vectors, and relevant technologies used as inspiration. Chapter III provides the methodology for developing a model architecture, developing metrics, and developing a simulation environment. Chapter IV presents the simulation results, analysis, and use cases. Chapter V provides the research summary and conclusion, as well as a brief discussion on potential future work.

II. Background and Related Work

2.1 Overview

Generally speaking, there are a plethora of extant communication architectures and protocols. Digital and analog communications devices of all shapes and sizes are readily available for nearly all applications. For most of these devices, architectures, and protocols, the offered levels of security is fixed, comparatively poor, or non-existent. On the other hand, it is also possible to improve the security of communication systems using a rearranged combination of the technologies and tools that exist and are distributed throughout other applications. For a security-focused communication protocol, technical inspiration can be gleaned from a review of the concepts of security and various architectures, protocols, and applied security approaches, in particular the adoption of multiple communication channels between end points. Individually, these existing approaches may have created isolated secure technologies, but viewed from a broader lens, they may be the answer to designing a security-focused multi-channel communications architecture, data handling protocol, and related metrics.

2.2 The Security Triad

Three characteristics are generally accepted to define the security of a traditional IT system: confidentiality, integrity, and availability. Together, these are often referred to as the IT security triad [3]. Additional sources also add authentication, non-repudiation, and code or data validation to the triad to form a dual triad [4], although that perspective is not as widely accepted. The aspects of confidentiality, integrity, and availability can also be viewed as physical layer characteristics, whereas the aspects of authentication, non-repudiation, and validation can be viewed as higher-layer, possibly even application layer, characteristics [5]. The focus of this research is on confidentiality, integrity, and availability. While each characteristic can be considered alone, they are often interrelated in how they impact operational risk so it is judicious to consider them in parallel.

Confidentiality is the aspect of a network that prevents messages from being understood by unauthorized users. Confidentiality is achieved by either preventing an eavesdropper from receiving a message, or by preventing the decoding of messages. This aspect alludes to possible adjustment of various characteristics, including the use of authentication and encryption as ways to limit access to the data. More exotic methods of ensuring data confidentiality include adjusting the SNR between a transmitter and receiver, changing the broadcast frequency or signal bandwidth, or using different encoding techniques for the data [6]. By reducing the transmission power, the SNR can be reduced to a level that the eavesdropper is not able to receive. Different encoding techniques may limit the eavesdropper's ability to receive the data or properly decode the message.

Integrity is a measure of accuracy and consistency of data sent through a network, e.g., the data received from the network is exactly the same data transmitted by the sender. Data integrity requires security measures to ensure the data has not changed in transit, or is only changed by authorized parties. In some systems, encryption is used to lock the data to a key, and in other systems, access control or authentication can be used to prevent unauthorized tampering of the data. Similarly, data validation algorithms can be used to verify uncorrupted transmission, such as checksums or hashes to perform error detection or correction. As they are intended to detect/correct noise induced errors, those are typically transmitted with the data, using the same path, and would therefore be easily modified by a deliberate attacker. Other methods of ensuring data integrity are increasing transmission power, thus increasing SNR, or decreasing the broadcast data rate, and thus reducing the received bit errors. These create a direct trade-off between confidentiality and integrity. Directional transmissions can reduce the need for significantly higher SNR, and could prevent data exfiltration, although it does nothing to stop malicious data injection or jamming. Another method is adding encryption, but that obviates the need for both transmitter and receiver to have the same encryption/decryption keys and algorithms and may add unacceptable latency in performing the encryption or decryption process. More thorough error checking and error correction capabilities can also be used. Rudimentary metrics for integrity include bit error rate and number of corrupted packets.

Availability is the reliable access by authorized users to the data when it is needed or expected. This alludes to the ability to take actions to mitigate jamming or a DoS attack. One method to ensure that the receiver has access to data when it is expected is to change channel state conditions: increase the SNR of the channel by increasing the broadcast power, use directional antennas with higher gain, or decrease the distance between transmitter and receiver. Of course, higher SNR increases the probability that the eavesdropper will intercept the transmissions, highlighting again the interplay between the elements of the security triad. Note that these approaches do not prevent jamming of the channel. Encryption can be used to make sure that only authorized users have access to the data if it is received, but that again requires that both transmitter and receiver have the proper keys and that the encryption algorithm does not detrimentally affect data latency. Rudimentary metrics for availability are number of dropped packets and system down-time, although other more complex measurements exist for wired systems [7].

2.3 Security Measurement

It is very difficult to measure the total security of an communication system. Some authors present various Quality of Service (QoS)-derived security metrics, such as traffic filtration metrics, key length, and device authentication [8]. Other authors suggest measuring the vulnerability of a system, which is the theoretical inverse of measuring security, and is equally difficult [9]. The difficulty is particularly true with measuring the confidentiality of an IT system. Typical measurements are not widely accepted or do not exist, and theoretical limits assume knowledge about eavesdroppers or their capabilities to intercept messages. Some assumptions include that the eavesdropper is in range of the message, has the proper equipment to detect and receive the message, and that if the eavesdropper is able to intercept any part of a message, the entire message has been compromised. As a method of quantifying confidentiality, the broadcast secrecy capacity is a theoretical measure of the successful transmission of confidential data given a threshold secrecy outage constraint [10, 11, 12], although these are exotic and complex calculations strictly applied to wireless channels. In simple terms, if an eavesdropper observes an SNR that allows the detection of a transmitted signal, and the eavesdropper is capable of decoding the message at a given data rate, then the eavesdropper has successfully exceeded the secrecy outage constraint [13]. The safest, though perhaps most fatalistic, assumption is that the eavesdropper is highly capable and knows as much about the systems as the defender.

2.4 Attack Vectors

Three general attack vectors on communication systems are eavesdropping, data injection, or system disruption. In a sense, a system disruption such as a DoS or Distributed Denial of Service (DDoS) attack, may be considered a specific form of a data injection attack such that all actual data is blocked from the receiver by the overloading of the network in some manner. As stated earlier, availability is a specific set of objective performance metrics, or QoS, including, for example, data or bit rate, jitter, bandwidth, transmission frequency, or power. These metrics omit any consideration of malicious interference. The fact that malicious interference affects whether a message successfully travels between a transmitter and a receiver is more about message integrity as opposed to system availability. It is true that a denial of service attack is making the network connections *not* available, but the inherent capabilities of the network, the systemic availability, have not changed. Using this framework, all malicious attacks on communication networks may be collected under the umbrella of attacks on confidentiality or integrity, whereas availability reflects strictly the physical attributes of the combined transmitter, channel, and receiver system.

This basic architectural model and the associated attack vectors have matured to the point that the various actors in the model are given well understood and generally accepted names. Alice and Bob represent a transmitter and receiver pair, or points A and B in the communication network. Eve represents the eavesdropper, the malicious actor that wants to intercept communications between Alice and Bob. It is nearly impossible to know what Eve's capabilities or intent are, and is equally challenging to know when Eve is actively intercepting messages. Mallory is the malicious actor that wants to inject false messages, thus disrupting correct communications. Finally, Dave is the malicious actor that wants to completely disrupt and cut off communications between Alice and Bob. Actions by both Mallory and Dave are easy to discover. Figure 1 shows how these characters fit into the existent architectural model. The basis for communication security is that Eve, Mallory, and Dave must be stopped, or their actions must be delayed as long as possible.

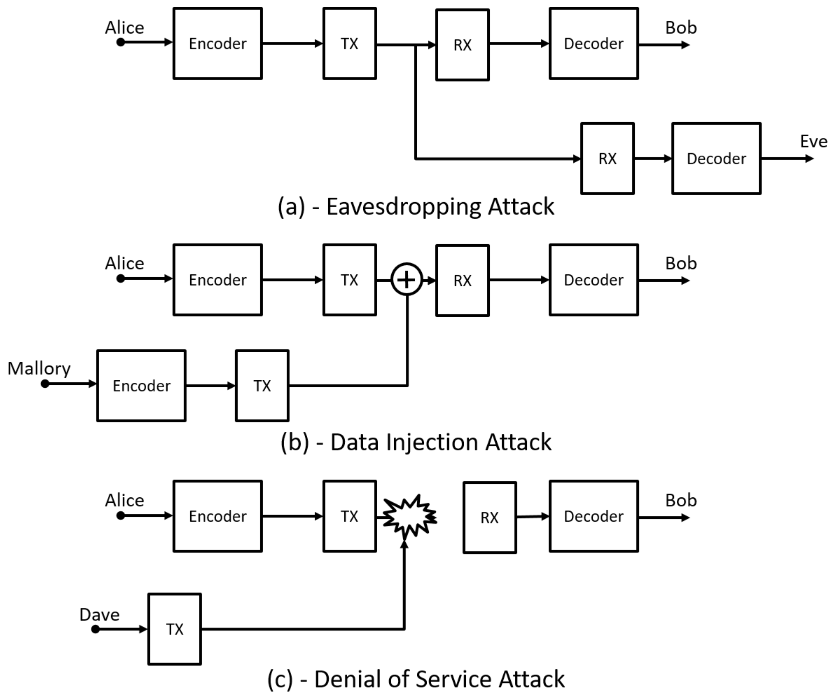


Figure 1. Traditional communication architectures subject to various attack vectors

2.5 OSI Model

Network services for devices and connections are generally standardized with the purpose of encouraging interoperability. To aid in understanding how the services interact, the Open Systems Interconnection (OSI) model assigns services to various levels, or layers, of a communication stack, and is depicted in Figure 2. The model divides network communications into seven segments for various levels of protocol implementation. Protocols may transcend layers, for example Transmission Control Protocol (TCP)/Internet Protocol (IP), however each layer represents a functional abstraction where a layer provides services to the layer above and receives services from the layer below. For all communication systems, protocols fit into the OSI model in some manner, but it may not be clear exactly where [14]. The Session layer, in particular, controls the connections, also referred to as sessions, between computers. Specifically, it is intended to establish, manage, or terminate connections between devices or applications. This layer is typically ignored by modern networking applications because TCP/IP is capable of performing many of the session control activities.

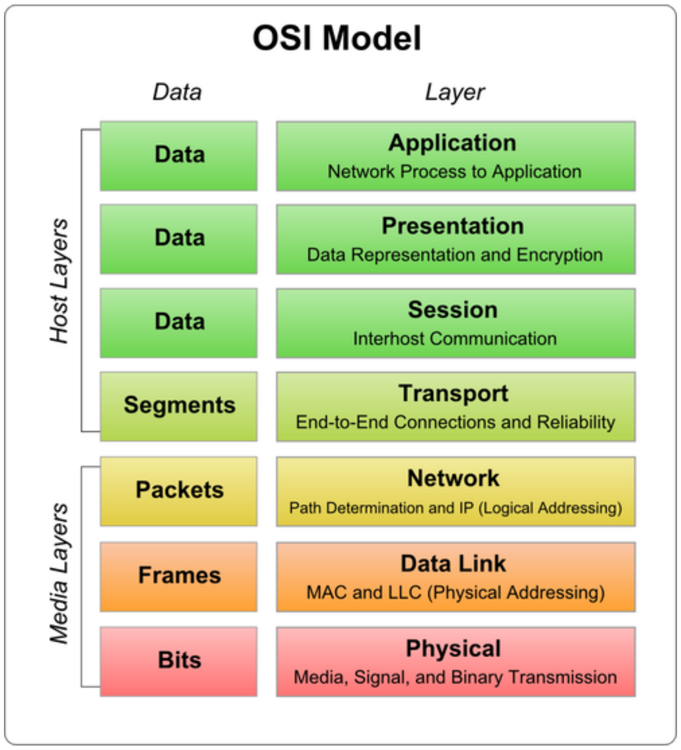

OSI Model
Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Physical Addressing)
Bits
Physical
Media, Signal, and Binary Transmission

Figure 2. Open System Interconnect (OSI) reference model highlighting the 7 layers associated with network communications [1].

Network security enforcement normally takes place at the data link and network layers, where devices are either granted or denied network access [1, 15, 16]. Little consideration is given to the session layer as being able to perform security functions above the transport layer. One exception is Transport Layer Security (TLS) which, in theory, operates at the application layer while interfacing with the lower layers. Furthermore, attempting to develop a new network architecture at the data link or physical layer ignores all the mature development inherent within existing technologies. With the ubiquity of TCP/IP, which spans the transport and network layers, if a protocol is not backwards compatible with TCP/IP, then it likely will not be readily used. Finally, any protocol that performs data handling should also interact predictably and transparently with the application layer.

2.6 Related and Applied Technologies

While the following technologies are used for other applications, they also serve as inspiration for solving challenges implementing multiple communication channels or security mechanisms or how these techniques and technologies are used to solve other existing problems.

2.6.1 Signalling System 7

Many applications in control systems maintain separate channels for data and control. For example, in Signaling System Number 7 (SS7)[1], the signaling path is separate and distinct from the voice channels that carry the telephone conversation. The predecessors to the SS7 performed these functions in-band with the audio path and were subject to exploitation. Having different channels, at different frequencies and differing bandwidths, allows for greater flexibility and more secure communications between network assets without the need to rely upon the availability or limitations of analog voice channels[2].

2.6.2 RAID

An example of data-at-rest performance improvement using multiple channels is found in the Redundant Array of Inexpensive Disks (RAID) architecture. Devel-

[1]SS7 was developed in 1975 as a set of protocols used to perform various network control and management functions, including call set up and tear down, in United States-based Public Switched Telephone Network (PSTN) communication connections.

[2]In reality, these two channels are not entirely separated. The Dual Tone Multiple Frequency (DTMF) digits dialed by a caller begin within the voice channel, but are recognized by the control channel and are an example of the signaling messages, including dialing a phone number, entering control functions like call-forwarding, or advanced billing information [17, 18].

oped in 1987, RAID demonstrated that by utilizing redundancy, an array of hard disk drives could be more reliable than any one disk drive while allowing greater data throughput[3]. Despite significant overhead, the ability to survive disk failures has made it very attractive in critical server environments and common in desktop systems.

2.6.3 Other Methods of Data Fragmentation

Similar to RAID systems, other methods of splitting data across multiple media or data connections channels exist. These methods have been developed to prevent unauthorized access to the information. In [19], the authors developed a detailed analysis of multi-channel communications with respect to maintaining privacy. In [20], the authors perform a review of fragmentation and duplication for cloud-based systems. Fragmentation for distributed storage systems is described in [21]. In [22], the authors describe self-adaptive fragmentation for large files to reduce overhead. Details about using the Trivium cipher[4] with TLS to distribute data across channels is described in [24].

2.6.4 Two-Factor Authentication

An abstract form of multi-channel communication is Two Factor Authentication (2FA), a subset of multi-factor authentication. This is an authentication methodology that requires a user to present two or more different forms of evidence to confirm the user's identity via separate delivery paths. Examples are presenting a password to a web page and having a text message sent to the phone number on file or presenting

[3]In a RAID array, data is split, known as striping, across various disks so that if one disk should fail, the data may be fully recovered despite not having all the original blocks of data. Various combinations of nested RAID levels may be used to reduce the vulnerabilities of simultaneous disk failures. The data may also be encrypted before or after striping, or both, as a manner of increasing confidentiality.

[4]Trivium is a synchronous stream cipher, a symmetric key cipher where plain text digits are combined one at a time with a key. It was designed to balance security, speed, and flexibility [23].

a key card and PIN to gain entry to a secured facility[5] [25]. By using multiple authentication factors sent via divergent paths, the likelihood that both messages are intercepted decreases. Even if a malicious actor intercepts one factor, full authentication by the malicious actor cannot occur without intercepting the other. Numerous other forms of 2FA also exist [26].

2.6.5 Bluetooth

In Bluetooth, adaptive frequency hopping spread spectrum is used to accomplish two different objectives. The first is to avoid unintentional jamming by other persistent signals or other Bluetooth devices and to avoid jamming them in return. The second is to make it more challenging for eavesdroppers by allowing only synchronized receivers to access the transmitted data [27]. The sequence of frequencies is unknown to an eavesdropper, and since Bluetooth performs 1600 hops per second, the entire spectrum of 79 channels would need to be monitored [28]. If this level of security is inadequate, Bluetooth also allows encryption at the link layer.

2.6.6 MIMO

There are some similarities between the proposed fragmentation of data across multiple communication channels and Multiple-Input Multiple-Output (MIMO) wireless communication architectures as defined by various standards [29]. MIMO exploits multi-path propagation within the radio frequency spectral medium to enhance the performance of a single data signal. The channels referred to in Section 3.4 can be considered logical channels, as opposed to their physical implementations, because the architectural model resides at a higher layer of the networking stack. With that, the communications architecture may utilize any possible physical medium for con-

[5]The user's identity is verified by using a combination of two or more factors: something they know, something they have, or something they are.

nectivity to the fragment and spread the data across the multiple connections as a technique to increase the security of the transmissions.

2.6.7 Multipath TCP and Multipath QUIC

Multipath TCP (MPTCP) is defined by the Internet Engineering Task Force's (IETF) proposed specification RFC 8684 [30] as a method to allow TCP to use multiple paths simultaneously for a single transport connection. This has the direct benefit of increasing redundancy and maximizing throughput. This method, described as reverse multiplexing and championed by Apple, stripes data from a single TCP connection across multiple subflows, each of which may take a different path through the network. The primary challenge with MPTCP is, when data is striped across connections, the sequence numbers in the TCP headers are either overwritten by middleboxes or gaps are left in the sequence numbers, triggering errors in the middleboxes [31]. Multipath QUIC, a draft IETF standard, is an extension of the QUIC protocol to enable simultaneous usage of multiple paths for a single connection. QUIC initially stood for "Quick UDP Internet Connections", however the name is no longer used as an acronym. Currently, QUIC is a general-purpose transport layer network protocol designed by Google which only allows communication in encrypted form. While features of TCP, User Datagram Protocol (UDP), and TLS appear in QUIC, QUIC is a completely different method of data transport. Bandwidth aggregation and seamless network handovers are direct benefits of using Multipath QUIC [32]. Both MPTCP and Multipath QUIC are used to increase overall bandwidth and throughput in modern cell phones and mobile devices on wireless networks.

2.6.8 Other Technologies

Using multiple channels can improve the performance of data-in-transit in diverse ways. A straightforward example is directly increasing the data rate, such that additional channels provide more bandwidth, e.g., channel bonding within IEEE 802.11 [33, 34, 35, 36]. Another example is frequency hopping through multiple channels, which is one of several techniques known as spread spectrum and which provides resistance to various forms of interference, noise, and jamming. Frequency hopping is a technique where the carrier rapidly switches among many frequency channels in a sequence known to both the transmitter and receiver. Spread spectrum is a technique in which a signal is transmitted on a bandwidth considerably larger than the frequency content of the original information [37]. This technique is currently used in Bluetooth, and such transmission diversity is also a key element of 5G wireless [38, 39].

2.7 Published Articles

The following subsections contain summaries of previous contributions that build on the related technologies and provide a foundation to the development of a security-focused communication protocol.

2.7.1 Securing Data in Transit Using Tunable Two Channel Communication

Two threat scenarios are considered in which an attacker has gained access to only one of two communication channels for the purpose of either eavesdropping or modifying the data-in-transit. Given these threats, five data splitting methods are presented that will enable the communicators to utilize the concept of two-channel communication to detect and adapt to the attacks in order to provide some level of data confidentiality or data integrity. Additionally, a simple proof-of-concept packet structure is introduced to facilitate transmission of the data across both channels in accordance with the data-splitting methods presented [40].

2.7.2 Multi-Channel Security Through Data Fragmentation

A multi-channel communications framework is developed. The architecture achieves security through distributing messages and their authentication codes across the available channels at the bit level. The system uses key exchange mechanisms to perform initialization functions, along with error detection and correction mechanisms. Tunable parameters are also provided, for example the number of channels, the duplication factor, and the number of fragments per message. This framework provides resilience against confidentiality and integrity attacks without relying on encryption [24].

2.7.3 Efficient Distribution of Fragmented Sensor Data for Obfuscation

In an effort to protect unattended sensors in contested environments from detection, exploitation, and DoS attacks, a mechanism for attack mitigation was developed. The protection mechanism encrypts, slices up, and distributes redundant file fragments throughout the sensor network. The improvements detailed in this document are to the custom data-splitting protocol. Improved security and scalability are claimed through the use of exact routing to avoid the adversarial collection and reconstruction of entire fragments [41].

2.7.4 Perfectly Secure Message Transmission

By abstracting away the network infrastructure and concentrating on solving the Secret Message Transmission problem, the authors present an algorithm with no complexity theoretic assumptions. They define the adversarial listener and disruptor, and using them, obtain upper and lower bounds for the secrecy capabilities. The authors propose that the algorithms simultaneously achieve the three goals of perfect secrecy, perfect resiliency, and worst case time linear in the diameter of the network. A final consideration is given to whether the adversarial listeners and adversarial disruptors cooperate, and how this may affect their algorithm [42].

2.7.5 A Model-Based Analysis of Tunability in Privacy Services

Typical security services do not allow trade-offs between performance and security during run-time. The security configuration is programmed during setup or installation, and remains static during operations. By having various security configurations available at or during run-time, the trade-off between security and performance can be tuned during operation. Three examples are evaluated demonstrating potential improvements to security while highlighting current limitations of system flexibility [43].

2.7.6 Quantification of Integrity

This article decomposes what data integrity means. As a starting point, the authors present that a program or system can have trusted and untrusted input and output, for authorized users and attackers, respectively. From that definition, the concepts of data contamination, leakage, and suppression are explored and quantified using conditional probabilities. Finally, various protection mechanisms are overlaid on the calculations to demonstrate how the quantification changes [44].

2.7.7 Probabilistic Bounds on Information Leakage

One approach to quantifying leakage, specifically quantitative information flow, is to measure how much of a secret an adversary can intercept by monitoring processes. The risk of secret information being discovered is calculated as an intermediate step toward quantifying information leakage within a set of bounds. The vulnerability is the probability the secret information is captured. Various techniques are used to improve the precision of the statistical analysis. Experimental results suggest that the approach can efficiently scale to larger data sets [45].

2.7.8 The Eavesdropping and Jamming Dilemma in Multi-Channel Communications

Using a zero-sum game-theoretic approach, the choice of whether an adversary jams or eavesdrops on a multi-channel communication system is explored. The communication system is assumed to be a broadcast wireless network separated into n channels, similar to different subcarriers in Orthogonal Frequency Division Multiplexing (OFDM) systems. The adversary can either eavesdrop or interfere with the communications; they cannot do both. The equilibrium strategies are presented from the adversary's prospective based on the various attack models and system conditions [46].

2.7.9 Combining Fragmentation and Encryption to Protect Privacy in Data Storage

The confidentiality constraints for data collections, e.g., lists of names, characteristics, and identifying information, may be maintained by a combination of encryption and fragmentation. Encryption is performed using known cryptographic algorithms. Fragmentation involves splitting sets of attributes so the attributes are not visible together. Various degrees and heuristic approaches to minimize fragmentation are presented, as are the degree of attribute affinity. Experimental results are presented to demonstrate the correctness and complexity of a given data collection query [47].

2.7.10 An Introduction to Spread Spectrum

At the most basic level, digital communications transmit pulses that correspond to the binary digits, or bits, contained within the message. The bandwidth is the range of frequencies contained within the original signal. Spread spectrum is a method by which frequency signal bandwidths used in transmission exceed the minimum bandwidth required to transmit the digital signal. The use of spread spectrum demonstrates resistance to noise jamming while also improving the processing gain at the receiver. Frequency hopping, the quasi-random hopping from frequency to frequency of the transmitted signal on a symbol-by-symbol basis, is one method of ensuring the sequential rather than instantaneous spreading of the spectrum [37].

2.7.11 Private Broadcasting Over Independent Parallel Channels

Using a coding scheme based on a superposition technique, the privacy of broadcast messages is explored. Two confidential messages are broadcast over independent parallel subchannels. Both special cases and general cases are demonstrated to maintain the privacy of two separate messages across the members of various groups of receivers. Ultimately, this highlights the usefulness of coding schemes in maintaining individual or group privacy in broadcast messages [48].

2.8 Summary

Using these existing technologies and background as inspiration and supporting infrastructure, this research effort will focus on detailing the development of a multi-channel communication architecture and protocol. The multi-channel architecture and supporting session layer protocol is expected to provide one possible solution to the complex challenge of ensuring security in communications. The areas which will receive the most focus are the development of the architectural model, supporting metrics, and the ability of the simulation environment to demonstrate survivability and data reconstruction in the presence of disruption and injection attacks while obfuscating what is intercepted by an eavesdropper.

III. Methodology

3.1 Overview

To date, there is no clear method of quantifying the overall security of a communication system, nor is there a comprehensive method of ensuring all aspects of security in communication systems. In an effort to demonstrate quantifiable and provably secure communications, one solution is to develop a security-focused communication protocol based on various existing architectural building blocks and technologies. Those technologies are detailed in Chapter II, Section 2.6. The development of this secure communications protocol involves three interrelated elements:

1. A model architecture demonstrating proposed components of a secure communications system on which the new protocol suite could be deployed;
2. A set of metrics to quantify the security characteristics between communication end-points employing the protocol; and
3. A simulation environment based on the architectural model and metrics to exercise the various functions of the protocol.

As the protocol is exercised within the simulation environment, the model and the metrics can be updated to improve robustness of the protocol.

3.2 Developing an Architecture

The traditional communication model is essentially a point-to-point architecture. Multiple hops may exist between the transmitter and receiver, relaying information and data and providing network flexibility to route around inefficiencies. As noted in Chapter II, Section 2.4, the traditional communication architecture is subject to possible adversarial influence, notably by eavesdropping, data injection, or denial of service. These are generalized as depicted in Figure 1, found in Section 2.4.

Informed by the various technologies detailed in Chapter II, Section 2.6, and patterning the architectural model after existing models that feature quantification methods [49, 50] and models that fragment data across multiple heterogeneous channels [41, 47, 22, 51], the concept of improving security through the use of a multi-channel communication architecture emerges.

3.2.1 Multiple Channels

Although a communication network typically uses only one network channel between two given nodes, the possibility exists to utilize multiple paths between nodes. Some existing communication protocols already allow for multiple parallel heterogeneous channels to exist between transmitter and receiver. One example is SS7 [17, 18], a protocol used in analog telephone networks, wherein the control channel is separated from the data channel. Another example uses two independent parallel channels to limit the amount of information leakage if either are intercepted [48]. RAID systems [52] allow the ability to fragment and stripe data across multiple storage units (disks) primarily for reliability and performance purposes, but that capability could also be employed for security purposes. TCP/IP networks allow for fragmentation of data into packets that comply with the Maximum Transmission Unit (MTU) of the network, although those strategies are strictly to optimize the performance of a single TCP/IP connection [53].

Inspired by [42], Figure 3 shows a simplified, arbitrary network with eight individual channels, any of which may be used to transport data. A message sent through the network may traverse any of the available channels, including those accessible to the set of adversarial listeners, A_L, or one of those influenced by the set of adversarial disruptors, A_D. This conceptual network serves as the basis for the proposed security-focused architectural model.

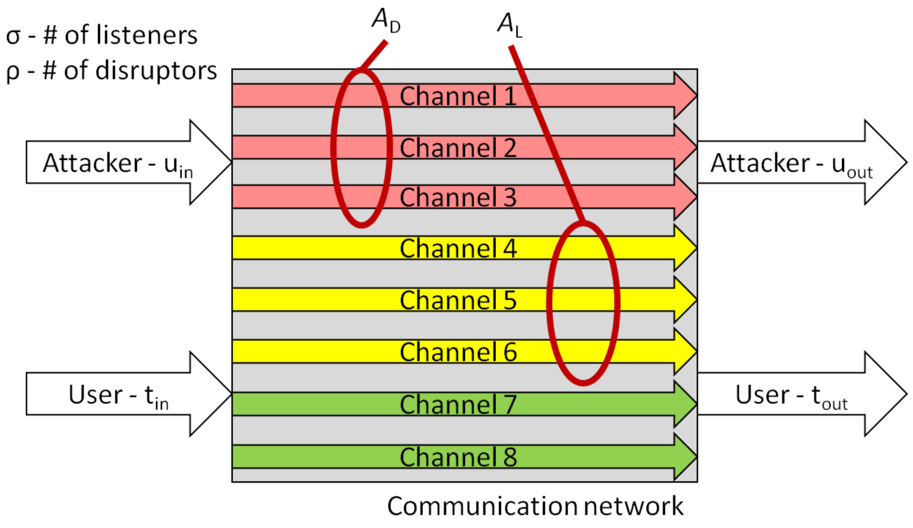


Figure 3. A network configuration with multiple possible channels.

To coordinate the sending of data across multiple channels, an apparatus is needed to make decisions on how to split the message across the available channels. In this work, that function is referred to as a *dissembler*. It will also be used for various maintenance actions such as set-up of the multiple channels, connection maintenance while transmitting data, and tear down of the connections when data transmissions end. Because this function involves orchestration of individuals channels, but serves to provide communication to an application layer, this work assumes that the *dissembler* resides in the Session layer of the OSI model. On the receiver side, an apparatus called an *assembler* will work in concert with the dissembler, also operating at the session layer, to reconstruct the message from the various channels. By comparison, employing a *dissembler-assembler* pair to manage connections does not rely on the outside application of security controls, e.g., access controls, configuration management, interfaces application controls, contingency plans, or business process application controls, among others, to all of the devices within the communication network [54]. Rather, the analysis of the system performance and probabilistic aspects of the network being attacked are monitored and maintained only near the end-points, at the *dissembler-assembler* pair. Expanding the basic network shown in Figure 3 to include the *dissembler-assembler* pair yields the initial architectural model show in Figure 4. In this basic model, multiple channels may be used to transmit the data between the endpoints. A typical employment would also have bi-directional transmissions, so each channel has both transmitters and receivers at the end points.

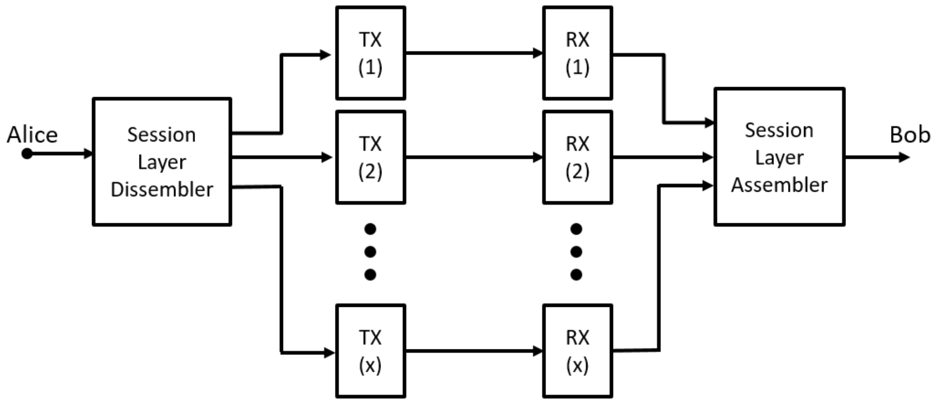


Figure 4. A basic architectural model communication network.

Modern networking also employs the use of intermediate nodes to provide flexible relaying and routing across broad networks. In a complex architecture that features multiple channels and multiple intermediate hops, similar to the one shown in Figure 5, data is routed over the multiple channels and is relayed by the multiple intermediate nodes, as necessary. In this architecture, a listener may be aware that multiple channels exist between the endpoints, but limited information about the additional channels is available at the intermediate node. Additionally, the intermediate nodes (if any) simply appear to have one input and one output. There may be one or more listeners, and they may or may not be working cooperatively. Further, even if a listener gains access to more than one channel, the challenge of reassembling the data remains. If some form of data injection or disruption were to occur on this type of network, the malicious injected data could possibly be discovered through data validation checks enabling data to be rerouted around the disruption. This model architecture is used to define the security-focused communication protocol.

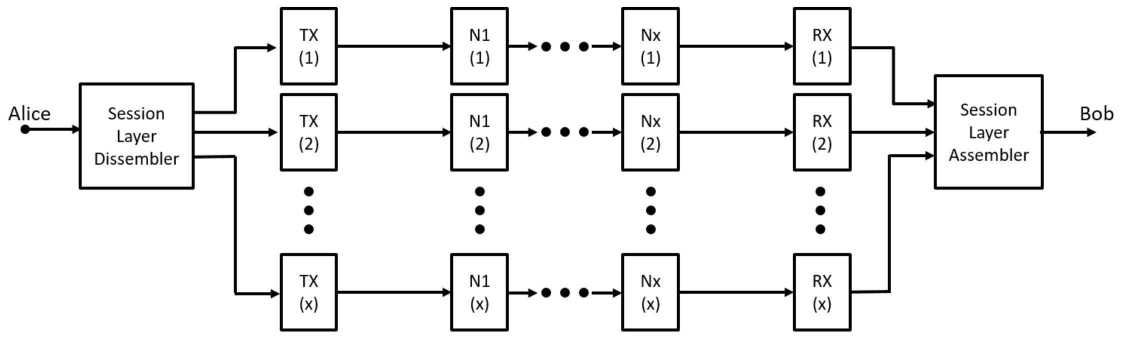


Figure 5. The security-focused architectural model.

Having an architecture featuring multiple channels introduces the possibility of transmitting the data in different ways, including various combinations of duplication and fragmentation of the data.

3.2.2 Duplicating and Fragmenting Design Choices

One method of using multiple channels is to duplicate all data over all the available channels. This would be helpful for ensuring message arrival, and can also protect against tampering with messages during transmission because there are multiple copies to validate the received data against. However, duplicating all the data over the multiple channels provides an eavesdropper additional opportunities to intercept the data. Therefore, fully duplicating data across all available channels can be considered one of several "corner cases" that may have only limited operational appeal.

Another method of employing multiple channels is inspired by RAID systems and involves splitting messages across the multiple channels to prevent unauthorized access to the information, while harnessing the available increase in data throughput.

There are numerous variations of fragmenting and distributing messages. The simplest method of distributing data is to use a round-robin technique across channels without any duplication, similar to RAID 0. Following this approach, the first packet on the first channel is filled, then the first packet on the second channel is filled, and so on until the first packet on all channels are filled, and the process starts again. Variations of this approach could include progressively smaller fragments, ultimately distributing individual bits of the message across the various available channels. These techniques increases security both by requiring an eavesdropper to have access to all the channels and by obfuscating the reconstruction of the message.

A hybrid approach of blending duplication and fragmentation is also feasible, akin to RAID 1+0 configurations. Depending on the size of the fragments and the number of channels, individual fragments may be duplicated across multiple channels, or on all channels. This technique allows for broad flexibility in ensuring minimal disruption while also ensuring minimal potential interception based on desired performance. The size of the message fragments, the number of duplicated fragments, and the order the various fragments are arranged on the available channels all represent design choices and characteristics that can be fine-tuned for desired performance. Those design choices will require varying amounts of overhead processing to receive and buffer the data packets at the assembler, to compare the duplicated portions, check for correct or tampered data, and verify all expected data is received, while also properly reassembling the fragmented data. Therein lies the trade-off between security and complexity.

3.2.3 Examples of Fragmentation

As one example of simple message fragmentation, message M is eight 8-bit American Standard Code for Information Interchange (ASCII) characters, for a total of 64 bits of data [55]. For this example, there are $n = 2$ channels, no intermediate nodes, and no data is duplicated[1] across the two channels. Assuming that M is split into two, four, or eight equally sized fragments, k, then the total number of bits per fragment is 32, 16, or 8 bits, respectively. These fragments maintain the byte-wise alignment of the ASCII characters. As there are two channels, each channel transmits one, two, or four fragments. With more fragments, there are more opportunities for rudimentary scrambling. The mechanism to reassemble the original message correctly is assumed to be contained in the receiver, referred to as the assembler in Figure 4. Therefore, in the examples shown in Figure 6, if an eavesdropper has access to only one of the channels, then the eavesdropper only intercepts at most half of the original message[2].

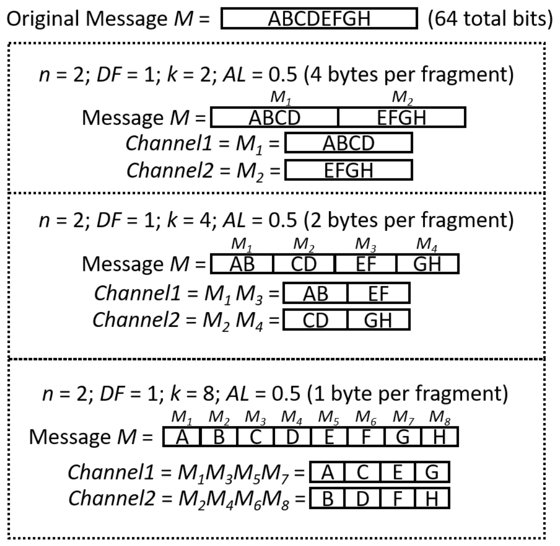

Original Message M = ABCDEFGH (64 total bits)
n = 2; DF = 1; k = 2; AL = 0.5 (4 bytes per fragment)
Message M = ABCD EFGH
Channel1 = M1 = ABCD
Channel2 = M2 = EFGH
n = 2; DF = 1; k = 4; AL = 0.5 (2 bytes per fragment)
Message M = AB CD EF GH
Channel1 = M1 M3 = AB EF
Channel2 = M2 M4 = CD GH
n = 2; DF = 1; k = 8; AL = 0.5 (1 byte per fragment)
Message M = A B C D E F G H
Channel1 = M1M3M5M7 = A C E G
Channel2 = M2M4M6M8 = B D F H

Figure 6. Byte-wise Fragmentation of Original Message Across Two Channels.

Figure 7 and Figure 8 continue the example, showing the 64-bit message M split into smaller fragments, such that each fragment is 4 bits or 1 bit, respectively [55]. If the sixteen 4-bit fragments were distributed across two channels, an eavesdropper would again only have access to half of the total message, although the bits appear

[1]The number of times the content of a message is duplicated across channels is referred to as the Duplication Factor (DF) and will be defined in Chapter III, Section 3.4.2. In this case, $DF = 1$

[2]The Average Loading (AL), or the average amount of the original message M that appears on any one channel, will be discussed at length in Chapter III, Section 3.4.2. In this case, $AL = 0.5$

scrambled if an 8-bit ASCII-encoding is assumed. It is interesting to note that in Figure 7, the lightly scrambled data on Channel 1 appears to be a series of ASCII letter "D", while meaningful information is transmitted on Channel 2. If the 64 one-bit fragments shown in Figure 8 are distributed across two channels, the amount of data on each channel remains half of the original message. In this case, the re-assembly protocol must interleave the two channels as they are received, increasing the complexity of the receiver. The scrambled bits on each channel do not appear to be in any pattern, especially if they are assumed to be ASCII-encoded text. As for the possibility of malicious injection or disruption on one of the two channels, if half of the data is either overwritten or completely destroyed, that impacts the entire original message M. In both of these cases, every byte of the original message is compromised. Therefore, even though message fragmentation shows promise in thwarting eavesdroppers, additional measures are needed to prevent malicious data injection or disruptions.

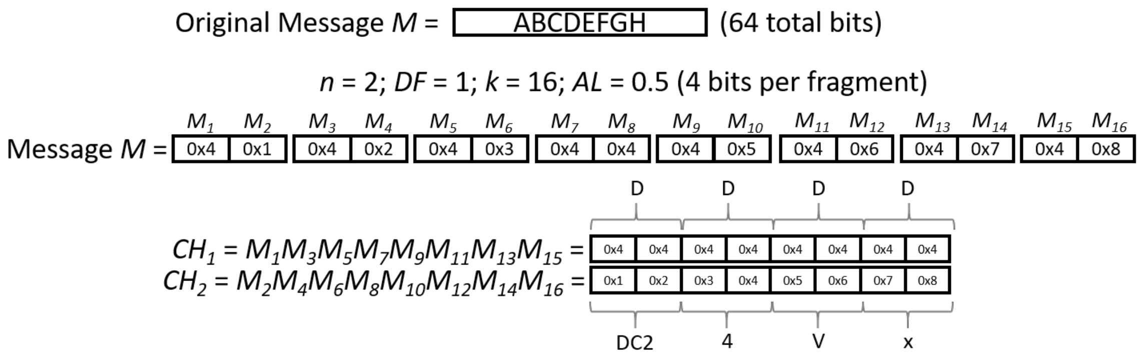
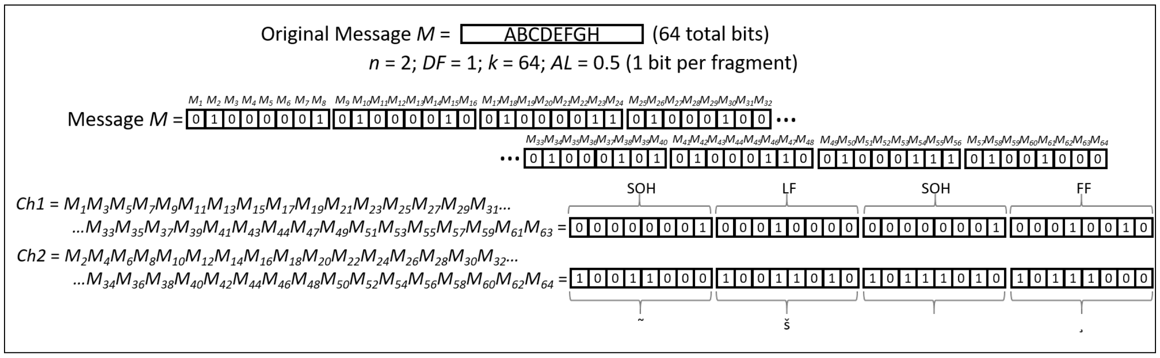

Original Message M = ABCDEFGH (64 total bits)
n = 2; DF = 1; k = 16; AL = 0.5 (4 bits per fragment)
M1 M2 M3 M4 M5 M6 M7 M8 M9 M10 M11 M12 M13 M14 M15 M16
Message M = 0x4 0x1 0x4 0x2 0x4 0x3 0x4 0x4 0x4 0x5 0x4 0x6 0x4 0x7 0x4 0x8
D D D D
CH1 = M1M3M5M7M9M11M13M15 = 0x4 0x4 0x4 0x4 0x4 0x4 0x4 0x4
CH2 = M2M4M6M8M10M12M14M16 = 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8
DC2 4 V x

Figure 7. 4-Bit Fragmentation of Original Message Across Two Channels.

As an additional set of examples, figures 9-11 show a similar system utilizing three parallel channels [55]. To avoid uneven division, message M is twelve 8-bit ASCII-encoded characters, for a total of 96 bits. For these examples, there are $n = 3$ channels, no intermediate nodes, and no data is duplicated[3]. Figure 9 shows the

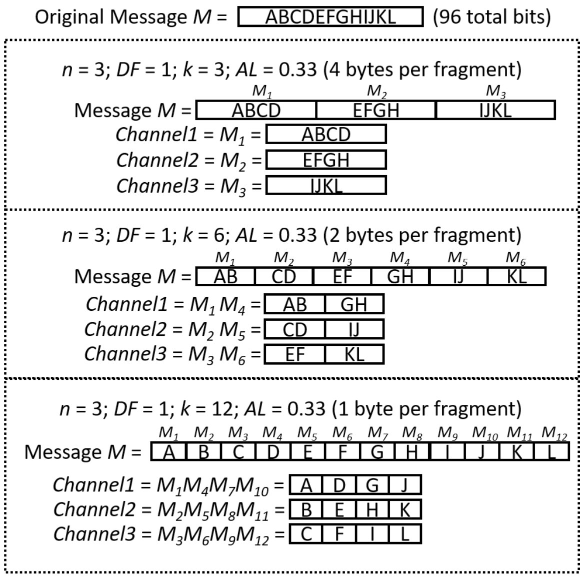
[3]Again, the Duplication Factor, $DF = 1$



Original Message M = ABCDEFGH (64 total bits)
n = 2; DF = 1; k = 64; AL = 0.5 (1 bit per fragment)
Message M =
Ch1 = $M_1M_3M_5M_7M_9M_{11}M_{13}M_{15}M_{17}M_{19}M_{21}M_{23}M_{25}M_{27}M_{29}M_{31}$… …$M_{33}M_{35}M_{37}M_{39}M_{41}M_{43}M_{45}M_{47}M_{49}M_{51}M_{53}M_{55}M_{57}M_{59}M_{61}M_{63}$ =
Ch2 = $M_2M_4M_6M_8M_{10}M_{12}M_{14}M_{16}M_{18}M_{20}M_{22}M_{24}M_{26}M_{28}M_{30}M_{32}$… …$M_{34}M_{36}M_{38}M_{40}M_{42}M_{44}M_{46}M_{48}M_{50}M_{52}M_{54}M_{56}M_{58}M_{60}M_{62}M_{64}$ =
SOH
LF
SOH
FF
~
š

Figure 8. 1-Bit Fragmentation of Original Message Across Two Channels.

examples with M split into two, four, or eight fragments, k, for a total number of 32, 16, or 8 bits per fragment, respectively. These fragments maintain the byte-wise alignment of the ASCII characters. Each of the channels transmits one, two, or four fragments. Because there are three channels, if an eavesdropper accesses one channel, they intercept at most one third of the original message[4]. However, a key difference between this example and that of Figure 6 is that an eavesdropper which captures two of the three channels only receives two thirds of the original message, and there may be no clear manner of reassembly.

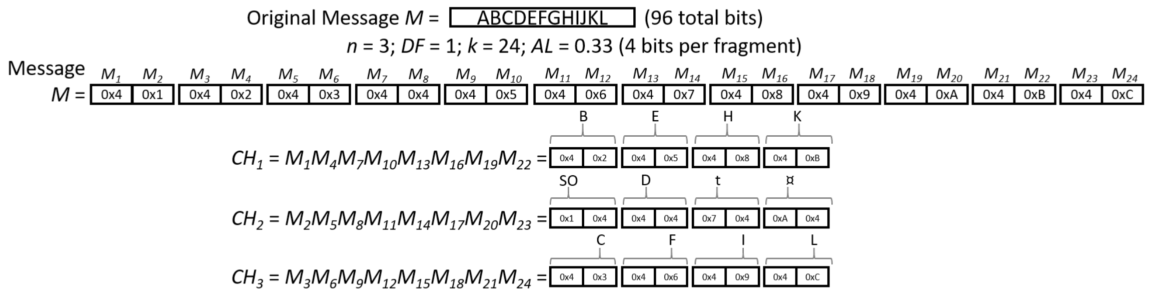


Original Message M = ABCDEFGHIJKL (96 total bits)
n = 3; DF = 1; k = 3; AL = 0.33 (4 bytes per fragment)
Message M = ABCD | EFGH | IJKL
Channel1 = M_1 = ABCD
Channel2 = M_2 = EFGH
Channel3 = M_3 = IJKL
n = 3; DF = 1; k = 6; AL = 0.33 (2 bytes per fragment)
Message M = AB | CD | EF | GH | IJ | KL
Channel1 = $M_1 M_4$ = AB | GH
Channel2 = $M_2 M_5$ = CD | IJ
Channel3 = $M_3 M_6$ = EF | KL
n = 3; DF = 1; k = 12; AL = 0.33 (1 byte per fragment)
Message M = A | B | C | D | E | F | G | H | I | J | K | L
Channel1 = $M_1M_4M_7M_{10}$ = A | D | G | J
Channel2 = $M_2M_5M_8M_{11}$ = B | E | H | K
Channel3 = $M_3M_6M_9M_{12}$ = C | F | I | L

Figure 9. Byte-wise Fragmentation of Original Message Across Three Channels.

[4]In this case, the Average Loading $AL = 0.33$

The final portion of these examples has the 96-bit message M split into smaller fragments, as shown in figures 10 and 11, such that each fragment is four bits or one bit, respectively. An eavesdropper would again have access to one third of the total message bits if a single channel were intercepted. An unexpected alignment in the ASCII-encoded text occurs between channels 2 and 3 of the 1-byte fragment example (Figure 9) and channels 1 and 3 of the 4-bit fragment example (Figure 10). In both examples, Channel 3 provides the same information, although context for correct reassembly does not exist. This appears to be a coincidental anomaly and not indicative of a broader pattern because the original message M is an ordered string of ASCII-encode characters. If the original message were not an ordered string of characters, or was any other variation of encoded data, the intercepted data would not appear in that manner. Furthermore, the assumption here was that the intercepted data is ASCII text. An adversary may not choose to make those assumptions, which highlights the difficulty in correctly reconstructing and decoding the (potentially incomplete) intercepted data.

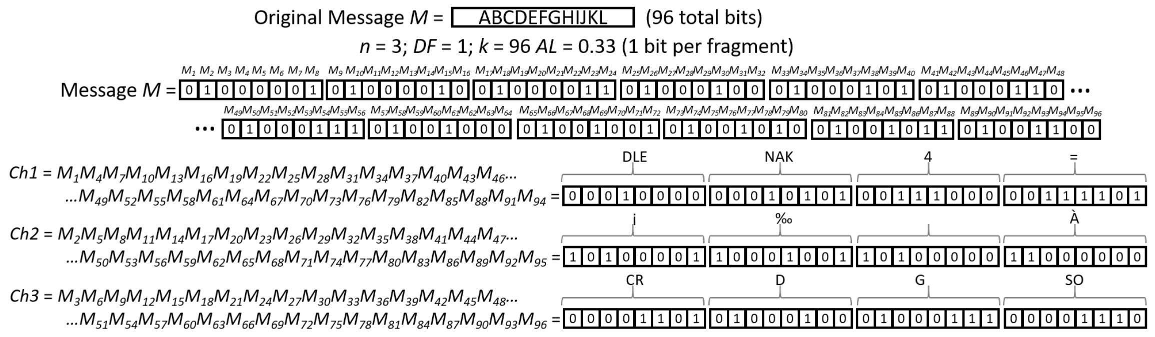


Original Message M = ABCDEFGHIJKL (96 total bits)
n = 3; DF = 1; k = 24; AL = 0.33 (4 bits per fragment)
Message
M₁ M₂ M₃ M₄ M₅ M₆ M₇ M₈ M₉ M₁₀ M₁₁ M₁₂ M₁₃ M₁₄ M₁₅ M₁₆ M₁₇ M₁₈ M₁₉ M₂₀ M₂₁ M₂₂ M₂₃ M₂₄
M = 0x4 0x1 0x4 0x2 0x4 0x3 0x4 0x4 0x4 0x5 0x4 0x6 0x4 0x7 0x4 0x8 0x4 0x9 0x4 0xA 0x4 0xB 0x4 0xC
B E H K
CH₁ = M₁M₄M₇M₁₀M₁₃M₁₆M₁₉M₂₂ = 0x4 0x2 0x4 0x5 0x4 0x8 0x4 0xB
SO D t ¤
CH₂ = M₂M₅M₈M₁₁M₁₄M₁₇M₂₀M₂₃ = 0x1 0x4 0x4 0x4 0x7 0x4 0xA 0x4
C F I L
CH₃ = M₃M₆M₉M₁₂M₁₅M₁₈M₂₁M₂₄ = 0x4 0x3 0x4 0x6 0x4 0x9 0x4 0xC

Figure 10. 4-Bit Fragmentation of Original Message Across Three Channels.

If the 96 one-bit fragments shown in Figure 11 are distributed across three channels, the amount of data on each channel remains one third of the original message. The scrambled bits on each channel do not appear to be in any pattern, especially when compared to the examples shown in figures 9 and 10. However, this again is based on the assumption that an adversarial eavesdropper will view the data as

ASCII-encoded text. If an adversary intercepts two of three channels in this example, the information received increases to two-thirds of the original message.



Original Message M = ABCDEFGHIJKL (96 total bits)
n = 3; DF = 1; k = 96 AL = 0.33 (1 bit per fragment)
Message M =
Ch1 = M1M4M7M10M13M16M19M22M25M28M31M34M37M40M43M46... ...M49M52M55M58M61M64M67M70M73M76M79M82M85M88M91M94 =
Ch2 = M2M5M8M11M14M17M20M23M26M29M32M35M38M41M44M47... ...M50M53M56M59M62M65M68M71M74M77M80M83M86M89M92M95 =
Ch3 = M3M6M9M12M15M18M21M24M27M30M33M36M39M42M45M48... ...M51M54M57M60M63M66M69M72M75M78M81M84M87M90M93M96 =
DLE
NAK
4
=
¡
‰
À
CR
D
G
SO

Figure 11. 1-Bit Fragmentation of Original Message Across Three Channels.

As for the possibility of malicious injection or disruption on one of the three channels, if a third of the data is either overwritten or completely destroyed, the entire original message M is impacted severely. This is especially true for the 1-bit sized fragments, such that if every third bit of the original message is missing, reconstructing and decoding the message becomes much more challenging. In the case of 4-bit fragments, one out of every three bytes of the original message is compromised in some way, whereas in the case of the 1-bit fragments, every byte of the original message is compromised. Therefore, as was the case with the 2 channel example, even though message fragmentation shows promise in thwarting eavesdroppers, additional measures are needed to prevent malicious data injection or disruptions.

This reorganized communication system model requires additional supporting components, in particular a robust data-handling protocol. The communication protocol requires standards to dictate how data is fragmented or duplicated across the multiple channels. Various metrics are also needed to evaluate the performance of the protocol and evaluate the difference between this and other communication systems.

3.3 Protocol Requirements

The developmental and performance requirements needed to guide the design process and provide a foundation for the security-focused session layer communication protocol include the following abilities:

- initialize the dissembler and assembler pair, and set up the system to reflect initial security requirements;
- dynamically increase or decrease the number of channels utilized by the protocol;
- operate at the session layer;
- obfuscate the number of channels;
- scatter or shuffle (in a non-cryptographic manner) the data across the number of channels;
- dynamically increase, or decrease, the amount of duplication of data on the various channels;
- detect an integrity attack, jamming, or data injection on (at least) one channel in a multi-channel configuration;
- reconstruct data from the various channels despite potential lost or changed data;
- provide real-time feedback between the dissembler-assembler pair so that performance adjustments may be made.

As each of these requirements focus on a specific set of functions in the protocol, additional characteristics and considerations are provided for each of the requirements.

3.3.1 Initialization process

The requirement for the dissembler-assembler initialization mechanism is necessary because at the start of a transaction, the desired architecture, e.g., the number of channels, and performance metrics, e.g., the duplication or fragmentation of the original message, are unknown to the receiver. In all but the most simple one-to-one direct connections, the only information available to a transmitter about a receiver is the IP address of the receiver. The transmitter and receiver must perform additional steps to set up the session-layer connection over one or several channels. Additionally, they must agree on the size of messages and the method of transmitting data, whether it is fragmented or duplicated in any manner. This could be accomplished simply as a set of hand-shaking messages sent from the transmitter to the receiver requesting a secure connection, and the receiver responding with a list of compatible capabilities and policies. However, an eavesdropper may already be listening to the initialization request and may be able to intercept those configuration settings.

One method of performing the hand-shaking is via encrypted communications, but that would require both transmitter and receiver to have the same set of encryption-decryption keys and be prepared for the encrypted communications. Therefore, the hand-shaking process could borrow from existing key-exchange processes, such as TLS [56] or the post-quantum computing key-exchange mechanism CRYSTALS-Kyber [57], to set up the initial connection. Once the initial verification of capabilities is completed, then the dissembler-assembler pair can negotiate further capabilities to include number of channels and methods of distributing the data, similar in fashion to the process USB devices perform for capability negotiation. This control infrastructure could be maintained as an encrypted channel throughout the entire set of transactions, thus providing a secure, readily available method of changing configurations in real time, much in the same manner that SS7 is used in legacy telephone networks. Additionally, if this control channel were be disconnected or fail in any other manner, a separate policy on the regular channels could be provisioned to trigger a renewed key exchange, a renegotiation process, and finally a resynchronization of the control and data channels. This process should provide adequate protections during the entire set-up, maintenance, and tear-down proceedings which the protocol must manage.

3.3.2 Adjust the number of channels

The ability to dynamically adjust the number of channels between the dissembler and assembler during a session is an extension of the initialization process, which may be needed during operation, either in response to cues from performance metrics or proactively as a rolling protection mechanism. Having the ability to adjust the number of active channels is imperative for rapid recovery from message loss or data injection attacks. It is assumed that as data is transmitted to the assembler, the assembler monitors key performance parameters. If a channel appears to have stalled, then that channel may be pruned from the session. Or, if there is suspicion that channels are being intercepted or are actively being corrupted, then the assembler may request another channel be initiated, and data begin to be distributed across the larger set of channels, thus further scrambling the data. At that point, the process of adding or removing channels becomes very similar to that of the initialization process and simply part of the renegotiation and resynchronization management performed by the protocol.

3.3.3 Session Layer operation

Based on the overview of the OSI model presented in Chapter II, Section 2.5, in conjunction with the requirements for initialization process and controlling the number of channels, it seems clear that a multi-channel protocol should reside at the session layer. This protocol handles the set-up, management, and take-down of multiple channels which is consistent with the intent of the session layer. Application layer programs and apps generate data which is encapsulated by the session layer protocol, providing information about the session policy and data alignment. The session layer protocol then initiates the multiple TCP/IP connections, enforcing policy requirements. As the packets move down the stack, each layer then adds its own header information, and the packets are transmitted through the network. All the benefits of the mature lower layer protocols, i.e. transport, network, data link, and physical layers, remain intact and available to the session layer protocol. Furthermore, the use of multiple channels and fragmentation would be transparent to the application layer functions. By functioning within the session layer, the protocol is able to use all of the services of the lower layers, and also minimizes the need to redesign applications.

3.3.4 Obfuscating the Number of Channels

As a way to obfuscate the operational architecture in use during a session, at no point should there be any indication to an eavesdropper exactly how many channels are being employed between the dissembler and assembler. This would limit potential information leakage to an eavesdropper about the configuration of the session. However, periodic messages should be exchanged, perhaps on an encrypted control channel, to confirm that the dissembler and assembler are synchronized, and these messages may resemble the hand-shaking messages sent during the initialization process.

3.3.5 Distributing Fragments

The possible variations for fragmenting within a multi-channel communication protocol are extensive. To limit the scope of the specific implementation presented in this work, a limited set of policies dictating basic configurations for data fragmentation were developed. Some of the preset configurations are detailed in Section 3.2.2. However, as described in the initialization requirement, the protocol must also manage switching between degrees of fragmentation and methods of distributing the fragments across the various channels. Additionally, the protocol must have a method of adding non-standard degrees of fragmentation and non-standard methods of distributing the fragments. As this protocol is intended to be available for all to use, the requirements and preset configurations would be readily available to everyone which means any adversary would also have access to the requirements and preset configurations. Having a means to update or overhaul the configurations via a secure connection adds to the tunable security of the protocol. This exchange of new capabilities is similar to and can be modeled after the initial negotiation process to add various policies or change the manner that feedback is incorporated in the renegotiation and resynchronization steps.

3.3.6 Duplicating Data

Similar to the distribution of message fragments, the protocol must be equipped with a preset number of policies that dictate basic duplication configurations. Some of the preset configurations are detailed in Section 3.2.2. The protocol would also need to manage switching between amounts of duplication and methods of distributing the duplicated fragments across the available channels. As with fragmentation, the protocol must have a method of adding non-standard approaches to duplicating and distributing fragments. The exchange of new capabilities is just as necessary for duplication as it was described for fragmentation. A complete set of possible duplication policies is beyond the scope of this work, but a representative set were discussed in Section 3.2.2.

3.3.7 Detecting Integrity Attacks

To improve the security and reliability of communications, it would be helpful for the protocol to have the ability to detect and survive various attacks on data integrity, in particular jamming or injection attacks. In general, noise is a natural phenomena and is present in all communication media to some degree. In some cases, maliciously generated noise may be induced onto communication channels. Solutions to some of that type of noise are readily available, from the use of Cyclic Redundancy Check (CRC) codes for error detection to error-correcting codes, such as Reed-Solomon [58] or Viterbi [59]. However, when data integrity is further challenged by jamming or malicious data injection attacks, the use and management of duplicate copies of the data sent across multiple channels becomes an attractive solution. Consider the case that fragments of a message are duplicated, i.e. each fragment is transmitted twice, over a communication architecture featuring three discrete channels. If one of those channels were to be actively jammed, there are still two channels that can relay the portions of the message to the receiver. Ideally, the receiver would also have the ability to recognize that one of those three channels is either receiving bad data or no data at all. Various checksums can be included in the message structure, similar to a CRC or the parity block used by RAID 5 configurations, such that the multiple checksums can cross-check the data on each channel. Thus, with a combination of data fragmentation and duplication across multiple channels and the cross-checking checksums, the protocol can actively detect and survive various integrity attacks. Finally, the initialization process can also be triggered when an attack is detected, causing the system to instantiate a new channel or change the method of fragmentation or duplication of data.

A serious challenge to detecting data integrity attacks comes when architectures utilize only one or two channels. If the one and only channel is being tampered with, then there is little that can be done to recover[5]. Even two channels presents a challenge with detecting an integrity attack, especially with a sophisticated attacker. A savvy adversary will be able to craft injected data to appear plausible, which then will cause an inability to determine which is the corrupted channel. In some cases, even a three channel architecture may not provide enough protection from sophisticated attacks. In the case of a three channel system, it is possible that an adversary can access two of the three channels, overwhelming the correction algorithm. Worse, with properly formed data and checksums on the exploited channel and noise errors on the other channels, the malicious data may appear more correct. It is also possible to have adversarial interference on one of the channels and environmental interference (or simply a bad connection) on a second channel. In these cases, there may not be enough checking or redundancy to counteract any interference, but there may be enough checking to identify the bad channel. At least with three channels, the degree of surviveability begins to increase.

3.3.8 Reconstructing Data

The protocol must properly recover and reconstruct the original message after fragmentation and transmission across multiple channels, despite modified or missing bits. The assembler must be able to receive all packets from the available channels and extract the important data and reassemble the original message, including checksum verification of the data on each channel. If data is found to be incorrect, or it is

[5]This is ignoring extreme temporal redundancy approaches such that messages or message fragments are sent repeatedly during a finite span of time over the same channel(s).

unclear what the correctly reconstructed message should be, the initialization process can be triggered, and the protocol behavior can be renegotiated and re-synchronized.

Reconstructing the original message with possible modified or missing data is generally not possible in architectures that utilize one or two channels. This limitation is similar to the limitations of detecting integrity attacks with only one or two channels. If only channel is being used to transmit data, and that data is lost or modified, there is little that can be done to reconstruct the correct original message. Even with two channels, data that is lost or modified can represent as much as half the original message. That portion of the data can be made to appear correct by a sophisticated adversary, thus calling into question which portion of the data is actually correct. A three channel architecture may provide enough redundancy to reconstruct data, but a few cases exist where sophisticated attacks will render the data unusable. An adversary can again potentially access two of the three channels, overwhelming the correction algorithms. Environmental interference on one channel can again coincide with adversarial interference on another channel. Again, with three channels, the degree of surviveability begins to increase.

3.3.9 Feedback

Real-time feedback between the dissembler-assembler pair is absolutely critical to the proper functionality of the security-focused protocol. All of the features and requirements detailed herein need some form of feedback from the assembler to the dissembler to close the loop on reporting any malicious or incorrect behaviors. The feedback messages can be similar to TCP/IP re-transmission requests sent over the multiple channel architecture, or perhaps custom renegotiation or resynchronization requests. The format of the feedback messages is not as important as the existence the feedback messages. Formatting can be simple or detailed based on the needs of the particular end-user, in so far as the end-user may have more strict security requirements. If that is the case, the format of the feedback messages can be renegotiated during the initialization process. As the session layer protocol employs the features of TCP/IP, the feedback messages create a two-way, multi-channel system.

The possibility does exist for a malicious actor to inject false feedback messages, or to block feedback messages altogether. Based on the level of security required, these feedback messages on the data channels can also be fragmented and distributed across the available communication channels. Handling feedback message in this manner effectively reverses the dissembler-assembler pair direction, such that the dissembler becomes the assembler, and the assembler becomes the dissembler. As is the case with the other requirements, this represents one possible solution to the need for feedback management within the security-focused protocol.

3.4 Developing Security Focused Metrics

With the architectural model conceptualized and requirements for the protocol defined, the next step in the development of the protocol suite is to derive a set of metrics to evaluate and compare desired performance characteristics and provide a foundation for the feedback mechanisms. As these metrics provide a baseline for comparing the security available in communication systems, they are collectively referred to as the Quality of Secure Service (QoSS).

To demonstrate security improvements in different network models, clear and repeatable methods of quantifying the security are essential. These metrics must describe the available security or the current security posture in a consistent manner. Ideally, these metrics aid in tuning the system to increase security, without impacting flexibility. Quantifying security remains a challenge. By analogy, consider computer performance metrics. To quantify the performance of a computer, it is generally accepted that "the only consistent and reliable measure of performance is the execution time of real programs, and that all proposed alternatives to time as the metric or to real programs as the items measured have eventually led to misleading claims or even mistakes in computer design" [52]. With such a clear metric, it is unsurprising that computer performance has steadily improved over several decades.

In this section, an approach to quantifying security is presented, based upon elements of prior security work, but tailored to the architectural model of multi-channel communication. According to Lundin [43], an equation to describe the tunable security for a communication system could be

$$TS : T \times Env \to R \tag{1}$$

where TS is the tunable security, which may be dynamically adjusted based on the user security requirements. The transmitter capabilities[6] are represented by T, the environmental conditions are represented by Env, and the overall system security requirements are represented by R. The goal is to map the tunable security services to the system security requirements. To achieve this, the tunable security services must first be decomposed into the constituent parts, such as the available number of channels, the use or disuse of encryption, and the amount of fragmentation across the network. In many cases, the environmental conditions are directly reflected in the traditional QoS measurements available from the service provider.

This initial version of the Quality of Secure Service (QoSS) metrics is a static snap-shot, reflecting the system security at one point in time. The multiplication operator in Equation (1) does not adequately address the numerous non-linear relationships between system capability and environmental aspects. Instead, Quality of Secure Service (QoSS) metrics captures those factors as an array of features or

[6]Wired and wireless networks have different characteristic values based on the specific technologies and protocols used.

values and then relates the transmitter capabilities and the environmental description to the Confidentiality-Integrity-Availability (CIA) triad, where confidentiality, C, and integrity, I, replace the transmitter capabilities, and availability, A, replaces the environmental descriptors.

Security measures are typically subjective. To achieve objectivity, we substitute measurements of confidentiality and integrity with the probability of each, designated as $P(C)$ and $P(I)$, respectively, as discussed in subsequent sections. Although it is unconventional to consider a DoS attack as impacting data integrity[7], doing so has the added benefit of collecting all adversarial influences into the metrics for confidentiality and integrity, leaving only the system and network capabilities to be considered as availability. The resulting QoSS equation is

$$QoSS : [P(C), P(I), A] \rightarrow \textit{Security Requirements} \tag{2}$$

representing a snapshot of QoSS metrics mapped to the set of security requirements. If the array of metrics does not directly map to the security requirements, then the QoSS for that network is inadequate, and the system must be redesigned. The array of metrics also provides a foundation to perform one-to-one comparisons between two networks.

3.4.1 Single Channel Metrics

Numerous researchers have attempted to quantify confidentiality with varying success [60, 61]. Confidentiality is the aspect of a network that protects against unauthorized message receipt, i.e., preventing an eavesdropper from either receiving or decoding messages. One approach to quantifying confidentiality is to redefine it as a probability so that

[7]This point is explored at length in Chapter II, Section 2.4.

$$P(C) = 1 - P(l) \tag{3}$$

where $P(C)$ is the probability of confidentiality and $P(l)$ is the probability of leakage. Leakage refers to an untrusted listener having access to an "information flow from secret inputs to public outputs" [44]. Inspired by Perfectly Secure Message Transmission (PSMT) [42], the set of all adversarial listeners, A_L, maps to a set of wires (channels), σ, that the listeners have access to; if one of the members of A_L has access to the information, then the probability of leakage exists.

For leakage to occur, a listener must intercept the message, decrypt it (where applicable), and then decode the data contained in the message. The probability of interception, $P(int)$, quantifies the probability that a listener with channel access will receive the message. The probability of decryption, $P(dcr)$, quantifies the probability that the adversary will decrypt it[8]. Finally, the probability of decoding, $P(dco)$, quantifies the probability that an adversary will decode the message[9] [62].

Consider the relationship between the probabilities of interception, decryption, and decoding. For data leakage to occur, an adversary must be able to achieve all three actions, i.e., decryption is irrelevant if the adversary is unable to receive any messages. Conversely, receiving every transmission ever sent is irrelevant if an adversary is unable to decrypt or decode the messages. The logical binary relationship of how $P(l)$ relates to $P(int)$, $P(dcr)$, and $P(dco)$ is captured in Table 1. The proposed equation to describe $P(l)$ in terms of $P(int)$, $P(dcr)$, and $P(dco)$ is

$$P(l) = P(int) \times P(dcr) \times P(dco). \tag{4}$$

Quantifying integrity is equally challenging. Integrity is a measure of the consis-

[8]This value reflects the quality of the encryption used, be it no encryption, a simple ROT13 algorithm, or a sophisticated encryption algorithm.

[9]This value highlights the differences in binary strings, and if the adversary has the ability to recognize those differences. For example, an adversary with a .mp3 file who mistakenly believes it is a .txt file, will not be able to derive useful information from that particular file.

Table 1. Logical binary relationship for the probability of leakage.

P(int)	***P(dcr)***	***P(dco)***	***P(l)***
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

tency, accuracy, and trustworthiness of data. Integrity implies that data has not been changed by unauthorized users in transit. One method of quantifying integrity is the "prevention of unauthorized modification of information" [44]. Under this assumption, unauthorized modification is *corruption*, resulting in

$$P(I) = 1 - P(c) \tag{5}$$

where $P(I)$ is the probability of integrity and $P(c)$ is the probability of corruption. Corruption here captures any damage to integrity yielding "two distinct measures of corruption ... named *contamination* and *suppression*" [44]. Contamination may arise from adversarial action, *injection*, or non-adversarial input, *noise*. Further, an adversary may carefully inject portions of false data (a spoofing attack), inject massive amounts of false data to disable communications (the traditional DoS attack), or overtly jam a message with a false signal (traditional RF jamming).

In this work, a choice was made to classify DoS attacks as being an attack on the integrity of the data or message, not as an attack on the availability of the network. Again, inspired by PSMT [42], the set of all adversarial disruptors, A_D, maps to a number of wires, ρ, that the disruptors have access to; if one of the members of A_D has access to the information, then the probability of corruption exists.

Corruption can be further decomposed into three components: noise, data suppression, and data injection. The probability of noise occurring in a message, $P(n)$, is the probability that a message will be adversely affected by noise. Noise is a natural phenomenon that happens regardless of the transmitter's capability. The probability of suppression, $P(s)$, quantifies the probability that an adversary will suppress or jam the message, thus, preventing the receiver from obtaining the message[10]. Finally, the probability of injection, $P(inj)$, quantifies the probability that an adversary will inject false data into the message. $P(inj)$ requires the ability to insert malicious data into a data stream, a much more sophisticated activity than that of jamming[11]. For the most part, noise is a natural phenomenon[12], it is consistently present and may influence $P(s)$ and $P(inj)$. Noise works cooperatively with $P(s)$ since both cause the receiver to incorrectly receive the intended message [62]. Based on these probabilities, the logical binary relationship for $P(c)$ is shown in Table 2 and reflected as

$$1 - P(c) = \big(1 - P(n)\big) \times \big(1 - P(s)\big) \times \big(1 - P(inj)\big). \tag{6}$$

Equation (6) does not adequately capture the behavior of the system. Noise may be detrimental to data injection, making the injected data unusable. Due to the interaction between $P(n)$ and $P(inj)$, namely that noise affects both intended and malicious transmissions, a more comprehensive equation is

$$\begin{aligned} P(c) =& \Big(\big(1 - P(n)\big) \times P(inj)\Big) + \big(P(n) + P(s)\big) - \big(P(n) \times P(s)\big) \\ &- \Big(\big(1 - P(n)\big) \times P(inj) \times \big(P(n) + P(s)\big)\Big). \end{aligned} \tag{7}$$

[10]To clarify, $P(s)$ is the active jamming by an adversary as quantified at the receiver, whereas availability is quantified by the transmitter's capabilities.

[11]As the adversarial intent of suppression is counter to that of injection, it is unlikely, although not impossible, to have high $P(s)$ and high $P(inj)$. This would be akin to an adversary steering a receiving channel to a compromised channel by jamming the intended channel. Neither of these speaks directly to the intent of an adversary but rather to the requirements and built-in capabilities of the transmitter and receiver.

[12]Noise may also be generated by an adversary, as an intermediate form of corruption leading to either data injection or jamming, which is why it is still considered potentially malicious, similar to a DoS attack

While less elegant than Equation (6), Equation (7) provides realistic results that account for all probabilities between 0 and 1 for each of the factors.

Table 2. Logical binary relationship for the probability of corruption.

$P(n)$	$P(s)$	$P(inj)$	$P(c)$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Methods exist for assessing and improving the performance of a system based on QoS measures [8]. The QoSS metrics used to describe availability are already conveyed in standard QoS metrics. This is reflected as $A = QoS$, where QoS is the set of metrics that include cost, jitter, latency, bandwidth, and bit rate, which already provide a repeatable method of measuring availability.

3.4.2 Multiple Channel Metrics

The relationship between listeners, disruptors, and the total number of needed channels is described by PSMT, which "abstract[s] away the network entirely and concentrate[s] on solving the Secure Message Transmission Problem" for a single transmitter and receiver pair [42]. Additional articles explore multi-channel architectures [63, 19], while others strive to prove the general case and optimize the statistical reliability and secrecy [64, 65].

In this architectural model, σ represents the number of wires (channels) between the transmitter and receiver available to the adversarial listener set, A_L, and ρ is the number of channels between the transmitter and receiver available to the adversarial disruptor set, A_D[13]. These sets of listeners and disruptors are shown in Figure 3, complete with eight possible data channels. Communication is two-way between the transmitter and receiver and, following PSMT, the number of channels that must exist between transmitter and receiver is given by

$$n \geq max\{\sigma + \rho + 1;\ 2\rho + 1\}. \tag{8}$$

This equation establishes how many channels must be used to maintain secure and reliable communication. If a channel is unavailable, then it must not be counted as part of n. If we assume the number of channels accessible to a listener or disruptor, then we can arrive at a specific quantification of n. For example, when $n = 8$ and $\sigma = 3$, the probability that any one channel of the eight could be listened to is 0.375. The probability of leakage [62] for each channel within a multi-channel architecture becomes

$$P(l) = \frac{P(int) \cdot P(dcr) \cdot P(dco) \cdot \sigma}{n}. \tag{9}$$

Similarly, the probability of corruption [62] for each channel within the multi-channel architecture becomes

$$P(c) = \frac{\left(\left(1-P(n)\right)P(inj)+\left(P(n)+P(s)\right)-\left(P(n)P(s)\right)-\left(1-P(n)\right)\left(P(n)+P(s)\right)P(inj)\right)\rho}{n}. \tag{10}$$

Therefore, the more channels there are in a network, the lower the probability of adversarial interference of the data[14]. This, then, follows the premise of PSMT: to have more channels than the combined set of listeners and disruptors $A_L \cup A_D$.

[13]In the general case, A_L or A_D may be subsets of or intersect with each other; i.e., $A_L \subseteq A_D$, or $A_D \subseteq A_L$ or $A_L \cap A_D$.

[14]One potential implication is that each channel may carry both a portion of the data and be used as a method to check for errors on the others channels, which will be explored in Chapter IV

In the same manner that multiple channels with some amount of duplicated data may thwart adversarial interference, message fragmentation may also thwart eavesdropping. As described in Section 3.2.2, message fragmentation is the splitting of data across the available channels, effectively parallelizing the data. Fragmentation describes how many portions the original message is divided into. Various methods of fragmentation are possible, including uniform or non-uniform fragmentation from 1-bit to the total m-bits in message M. Published research exists on particular approaches to fragmentation [66, 41]; for this work, we assume that fragments are of equal size across the available network connections. If C_n is the set of n channels, and F_M is the set of k fragments of $1 \leq |f_i| \leq m$-bits of the message M, then

$$F_M = \{f_{(M,1)}, f_{(M,2)}, f_{(M,3)}, \ldots, f_{(M,k)}\} \tag{11}$$

$$f_{(M,i)} \subseteq M \ for \ 1 \leq i \leq k \tag{12}$$

where each fragment is unique. The Channel Load, L, is the percentage of M on a particular channel j, such that

$$L_{(j,M)} = \frac{\sum_{i=1}^{n} |f_i| \ for \ f_i \in C_{(j,M)}}{|M|} \tag{13}$$

and the Average Channel Loading (AL) for the set of channels is

$$AL_M = \frac{\sum_{i=1}^{n} L_{(j,M)}}{n}. \tag{14}$$

For example, $F_M = \{f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8\}$ is the set of eight fragments of message M on a network that has $n = 8$ channels, and each channel transmits two fragments. Therefore, $AL = 0.25$. Message fragmentation also allows for duplicating data across channels. The Duplication Factor (DF) measures the average number of times a given fragment is transmitted, indicating the network redundancy. The

DF may increase as compensatory tuning for known adversarial interactions. For the previous example, $DF = 2$, since each fragment is sent across two channels and, thus, duplicated twice. For these calculations of DF and AL, the fragment sizes are uniform.

The AL and DF directly affect $P(C)$ and $P(I)$. Of the constituent parts of $P(C)$, $P(int)$ is only affected by DF in aggregation across all channels because the probability of interception of a single channel is not necessarily improved by duplication or fragmentation. However, $P(int)$ may be increased by the message M being duplicated across multiple channels, offering an adversary more opportunities to intercept portions of the message.

Therefore, DF is only multiplied by $P(int)$ when averaging all the channels into a composite probability of leakage. For the constituent parts of $P(I)$, duplication directly affects $P(s)$ because sending fragments multiple times decreases the probability of lost data through suppression. $P(n)$ and $P(inj)$ are not directly influenced by duplication. Thus, $P(s)$ is divided by DF for each channel, giving

$$P(c) = \frac{\left(\left(1-P(n)\right)P(inj)+\left(P(n)+\frac{P(s)}{DF}\right)-\left(P(n)\frac{P(s)}{DF}\right)-\left(1-P(n)\right)\left(P(n)+\frac{P(s)}{DF}\right)P(inj)\right)\rho}{n}. \tag{15}$$

Fragmentation does not necessarily increase or decrease $P(s)$ except that it allows for duplication. However, fragmentation does directly affect $P(inj)$ since each fragment sent needs to be modified by the adversary in order to have malicious data accepted at the receiver. Thus, $P(inj)$ is multiplied by AL for each channel, giving

$$P(c) = \frac{\left(\left(1-P(n)\right)P(inj)AL+\left(P(n)+\frac{P(s)}{DF}\right)-\left(P(n)\frac{P(s)}{DF}\right)-\left(1-P(n)\right)\left(P(n)+\frac{P(s)}{DF}\right)P(inj)AL\right)\rho}{n}. \tag{16}$$

Applying the PSMT and decomposing the network into constituent channels yields

$$QoSS : \begin{bmatrix} P_1(C), & P_1(I), & QoS_1 \\ P_2(C), & P_2(I), & QoS_2 \\ \vdots & \vdots & \vdots \\ P_n(C), & P_n(I), & QoS_n \end{bmatrix} \rightarrow Sec\ Reqs, \tag{17}$$

which highlights that each channel has its own characteristics. From the end-user perspective, only the aggregated QoSS for the entire network is apparent. With insight into each channel's QoSS, an analyst may suggest a different quantity of channels, different fragmentation or duplication, or a different encoding or encryption algorithm if adversarial actors attempt to influence communications.

3.4.3 Reliability and the Multi-Channel, Multi-Hop Metrics

Reliability is defined as the "ability of an item to perform a required function, under given environmental and operational conditions and for a stated period of time" [67]. For the purposes of the QoSS metrics, the item is a communication network, and the required function is to transmit data in a secure manner. Additionally, the definition of security is the "availability of performance with respect to prevention of deliberate hostile actions", where availability of performance includes reliability performance, maintainability performance, and maintenance support performance. With that, the reliability function is the "probability that the item survives" over a specific period of time [67]. For the concept of unconditional reliability to be possible, both reliability and security are necessary. The message must be correct and must not be intercepted; in other words, both the disruptor and the listener must fail to achieve their objectives [68].

Calculation of the reliability of a system varies, depending on whether the system is constructed in series, parallel, or is a hybrid, containing a combination of series and parallel items. Series systems have items, elements, or devices connected in series, and the entire system fails if any of the individual elements fail. The reliability of such a system is calculated as the product of each element's reliability in the series, or specifically:

$$R_{series} = R_1 \times R_2 \times R_3 \times \cdots \times R_n = \prod_{j=1}^{n} R_j, \tag{18}$$

where R represents the reliability of each element. The practical implication is that the reliability of a series system is always lower than the reliability of any of its components. The reliability of a parallel system represents the reliability of mutually independent elements such that all must fail for the entire system to fail. The reliability of a parallel system is therefore calculated as:

$$R_{parallel} = 1 - \prod_{j=1}^{n} (1 - R_j), \tag{19}$$

where, again, R represents the reliability of each element [69]. In contrast to a series system, the reliability of a parallel system is always higher than the reliability of any of its components. Hybrid systems will, of course, vary, depending on their construction.

Applying the definitions for reliability to the QoSS metrics, a failure is any event where data fails to be securely transmitted across some portion of a network. Any failure will affect the confidentiality, integrity, or availability of the network, both individually and collectively. As the probability of failure is complimentary to the reliability, then, by extrapolation, the probability of confidentiality and the probability of integrity can both be viewed as measures of reliability in addition to the typical availability metrics.

Iteratively, each hop in a series may be calculated and then, using typical reliability calculations for a system of components may be multiplied to yield the reliability of the system. To obtain the reliability of the larger system, the probability of interception, decryption, decoding, noise, suppression, and injection may all be multiplied by each other and the full QoSS may be calculated in the same manner. Each hop has the similar cumulative effect of reducing the reliability of the network in terms of security because an adversarial listener or disruptor has additional opportunities to infiltrate. A naive approach would simply average the values together, but that overlooks the cumulative effect of multiple hops. The architecture presented here makes the assumption that message packets are reformed and retransmitted at each hop. As such, signal degradation is only a concern on a per-hop basis. Admittedly, this view is overly conservative in that the possibility of a listener obtaining a message at one or more hops may be counted multiple times.

To compute an overall QoSS value of a system composed of multi-hop and parallel logical channels, it is necessary to consider the effects of the fragmentation and duplication of messages on the QoSS metrics.

Consider a simple point-to-point system with one channel and perfect, unbreakable encryption. What if that system were to instead have two parallel channels, one with perfect encryption, and the other with no encryption? How should the summary QoSS values then be computed? If the data were fragmented into equally-sized portions, and an adversarial listener has access to both of those channels, then they would have unencrypted access to exactly half of the data. However, equal sized portions is not the only way to split the data. A “single fragment could represent a single bit of a message while the maximally-sized fragment could contain the entire message” [24]. Therefore, the amount of data accessible to an adversary is based upon the proportions of data that are transmitted across the two parallel channels, and would be represented as a weighted average.

The previous example assumed that the message was split evenly and without any duplication. If only a single copy of each bit from a given message is sent, either physically via multiple channels or temporally via repeated messages, then there is zero duplication. Conversely, “full duplication indicates that the entire message is sent across each of the available channels” [24], which would lead, in the previous example, to the adversary having full access to the message via the unencrypted channel.

If the data is unevenly or non-uniformly split among the channels, in a similar manner to RAID architectures [52], or data are duplicated across the channels, then the amount of data accessible to the adversary would be a weighted average based upon the proportional splitting and the DF of the channel(s), which in turn causes the broader QoSS metrics to utilize a weighted average across the available channels.

To maintain simplicity in the example cases, data is assumed to be split equally and without duplication across parallel links. However, the architectural model and QoSS metrics can support more elaborate combinations when needed.

When reducing a complex equation in algebra, specific rules dictate which operations are solved first, thus simplifying to a more easily-solvable equation. The same is true when reducing the complexity of a communications network into an equivalent single-channel, point-to-point network. Through similar decomposition, it is possible to analyze a complex communication network that utilizes multiple intermediary nodes and multiple channels, similar to the example network link nodes A and D as shown in Figure 12, and to quickly compare the QoSS metrics to another network. When data are split across multiple channels, the end-user may remain agnostic to how the data is handled between the end-points within the larger, end-to-end network, in the same manner as an end-user is agnostic to multiple intermediary hops in current network architectures.

The initial step in reducing complexity is to assess how the two end-points are connected, if there are multiple intermediary nodes, and whether the network utilizes multiple channels. In the reduction and simplification process, the primary goal is to derive an equivalent point-to-point network by reducing the number of hops or multi-channel connections between end-points thus leading to an equivalent point-to-point link [70]. To begin, the QoSS metrics, specifically $P(C)$ and $P(I)$, must be calculated for all intermediary connections between nodes. This is accomplished by applying Equations (9) and (16) and solving for $P(C)$ and $P(I)$, respectively. Using the example of Figure 12, this would require calculating $P(C)$ and $P(I)$ for the eight individual links.

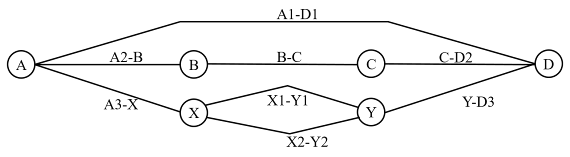


Figure 12. A Notional Complex, Hybrid Multi-Channel, Multi-hop Architecture.

For the next step, if the link between two nodes contains multiple channels, for example, between nodes **X** and **Y** in Figure 12, then the respective QoSS metrics for those two links are averaged. To be precise, a weighted average of the links is applied based on AL and DF as described above. If, on the other hand, the link between two nodes contains multiple hops, for example, link $\langle$**A2-B, B-C, C-D2**$\rangle$ in Figure 12, then the respective QoSS metrics for those three links are multiplied together, as detailed by Equation (18). It is not possible to apply Equation (18) if there are multiple channels between the end-points, thus collapsing multi-channel connections into an equivalent single channel takes precedence, after which Equation (18) may be applied. This is the process for finding the QoSS equivalent for nodes [$\mathbf{A} > \mathbf{X} > \mathbf{Y} > \mathbf{D}$] in Figure 12. Repeat steps 1 and 2 as necessary to further reduce the system in Figure 12 to one, point-to-point connection. By following these steps, the point-to-point equivalent QoSS for a complex network may be derived. For example, once the two links between nodes **X** and **Y** have been collapsed into one equivalent QoSS value by weighted average, then the series of three links of $\langle$**A3-X, X-Y, Y-D3**$\rangle$ may be multiplied together per (18), at which point the system may be fully reduced to one point-to-point link.

3.5 Metrics Join the Architectural Model

The following case studies will provide detail as to how to calculate the probability of confidentiality and the probability of integrity for various architectural configurations. The five example architectural models are presented to highlight the initial metrics estimates and are intended to be refined as systemic understanding is increased. For simplicity, the probabilities used in the following examples are discrete values; however, any value between 0 and 1 is possible. In developing the QoSS metrics, estimating the intermediate values is a challenge. As a starting point, 0 may be used for a network that has absolutely no encryption, 0.5 may be used for a system that has minimal or sub-standard encryption, and 1 may be used for a system that employs strong encryption. Several intermediary steps may be necessary to reach the equivalent single-channel, point-to-point network, which provides the most direct method of comparison.

Incremental changes may be employed as desired or as needed after a baseline understanding is developed, much like understanding the incremental difference between AES-128 and AES-256, or the difference between DES, triple-DES, and AES. The primary goal of the initial metrics development is to apply estimates for each of the constituent elements as implied by [45]. Further refinement of those estimates may be applied after more thorough system analyses, e.g., after running a simulation several times to reveal possible emergent behaviors.

During the early stages of analysis, the difference between a probability of 0.76 and 0.77 remains undefined and the numbers tend to be more arbitrary. This serves to assign a starting point for analysis, thus, establishing a baseline. Given the five example architectural models that follow and some initial probabilistic estimates for the various characteristics, the QoSS metric calculations are applied. Each case has a realistic configuration that allows for one-to-one comparison.

3.5.1 Single-Channel, Point-to-Point Network

The first architectural example utilizes a single point-to-point communication channel to provide a realistic baseline and a starting point for developing the QoSS metrics [62, 70]. With $n = 1$, there is assumed to be $\sigma = 1$ possible listener, and $\rho = 1$ possible disruptor. Because the message cannot be split $AL = 1$ and $DF = 1$ since, for this exmple, the message is only sent once. This example is depicted in Figure 13, and Table 3 shows the notional probabilities for this network that has no encryption, standard data encoding, and a moderate probability of interception because it uses a standard broadcast frequency and a moderately strong broadcast signal, which also results in a low probability of noise. Since there is only one channel and one point-to-point connection, there is only one set of QoSS metric characteristics.

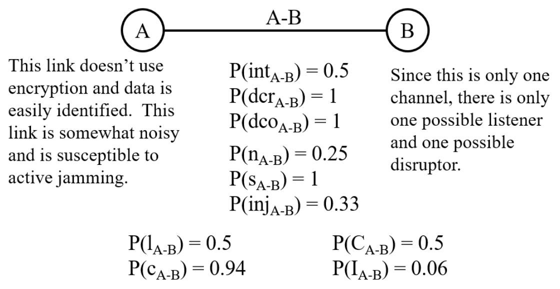

A
A-B
B
This link doesn't use encryption and data is easily identified. This link is somewhat noisy and is susceptible to active jamming.
$P(int_{A\text{-}B}) = 0.5$
$P(dcr_{A\text{-}B}) = 1$
$P(dco_{A\text{-}B}) = 1$
$P(n_{A\text{-}B}) = 0.25$
$P(s_{A\text{-}B}) = 1$
$P(inj_{A\text{-}B}) = 0.33$
Since this is only one channel, there is only one possible listener and one possible disruptor.
$P(l_{A\text{-}B}) = 0.5$
$P(c_{A\text{-}B}) = 0.94$
$P(C_{A\text{-}B}) = 0.5$
$P(I_{A\text{-}B}) = 0.06$

Figure 13. A Single-Channel, Point-to-Point Network and QoSS Metrics.

Assuming a wireless communication system, we can assign a high probability of suppression under the assumption of an omni-directional receiver which is susceptible to jamming. The probability for injection is moderately high, though not as high as the probability of suppression, because injection is more challenging than suppression. Based on these constraints, the single-channel network has a high probability of

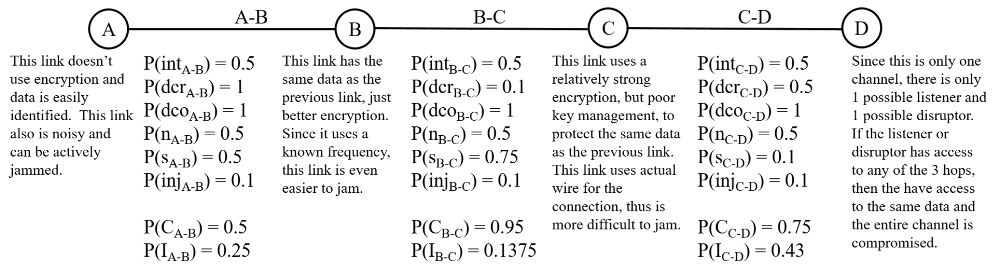
Table 3. Input and Output Values for a Single-Channel Architecture.

channel	$P(int)$	$P(dcr)$	$P(dco)$	$P(l)$	$P(C)$	$P(n)$	$P(s)$	$P(inj)$	$P(c)$	$P(I)$
1	0.5	1	1	0.5	0.5	0.25	1	0.33	0.9381	0.0619

leakage, with a corresponding probability of confidentiality. The probability of corruption is also very high, with a correspondingly low probability of integrity. These probabilities may be improved by using encryption and by using directional receivers or a wired connection. This first case is intended to be somewhat representative of a simple, yet realistic, communication network and its characteristics, and will also serve as a baseline comparison for the multi-hop and multi-channel networks

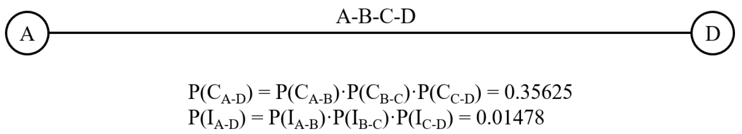
3.5.2 Single-Channel, Three-Hop Network

A more realistic example is a network that employs several intermediary nodes to forward data, as shown in Figure 14 [70]. For this example, two intermediary nodes relay the data, making it a four-node network, with three independent links between the end-point transmitter and receiver. Each section of the communication network, ⟨**A-B, B-C, C-D**⟩, may have different characteristics because they may traverse different interconnecting channels or they may travel different distances. Those characteristics are also shown in Figure 14, with notionally assigned values for illustration. To calculate the QoSS for each link between hops, the values may be handled in the same manner as the single channel in Figure 13. However, to calculate the equivalent single channel point-to-point network, as shown in Figure 15, the equivalent $P(C)$ and $P(I)$ are simply calculated by multiplying each of the component probabilities of the confidentiality or probability of integrity values, respectively. This final value of the equivalent single-channel, point-to-point network is shown in Figure 15.



A-B
B-C
C-D
A
B
C
D
This link doesn't use encryption and data is easily identified. This link also is noisy and can be actively jammed.
P(int_A-B) = 0.5
P(dcr_A-B) = 1
P(dco_A-B) = 1
P(n_A-B) = 0.5
P(s_A-B) = 0.5
P(inj_A-B) = 0.1
P(C_A-B) = 0.5
P(I_A-B) = 0.25
This link has the same data as the previous link, just better encryption. Since it uses a known frequency, this link is even easier to jam.
P(int_B-C) = 0.5
P(dcr_B-C) = 0.1
P(dco_B-C) = 1
P(n_B-C) = 0.5
P(s_B-C) = 0.75
P(inj_B-C) = 0.1
P(C_B-C) = 0.95
P(I_B-C) = 0.1375
This link uses a relatively strong encryption, but poor key management, to protect the same data as the previous link. This link uses actual wire for the connection, thus is more difficult to jam.
P(int_C-D) = 0.5
P(dcr_C-D) = 0.5
P(dco_C-D) = 1
P(n_C-D) = 0.5
P(s_C-D) = 0.1
P(inj_C-D) = 0.1
P(C_C-D) = 0.75
P(I_C-D) = 0.43
Since this is only one channel, there is only 1 possible listener and 1 possible disruptor. If the listener or disruptor has access to any of the 3 hops, then the have access to the same data and the entire channel is compromised.

Figure 14. A Single-Channel, Three-Hop Network and QoSS Metrics.



A-B-C-D
A
D
P(C_A-D) = P(C_A-B)·P(C_B-C)·P(C_C-D) = 0.35625
P(I_A-D) = P(I_A-B)·P(I_B-C)·P(I_C-D) = 0.01478

Figure 15. A Single-Channel, Point-to-Point Equivalent of a Single-Channel, Three-Hop Architecture.

3.5.3 Three-Channel, Point-to-Point Network

A third example network incorporates the initial application of multiple channels between the two end-points, as described by the theory of PSMT [42, 64, 65, 71] and is shown in Figure 16 [62, 70]. In this example, the communication network uses three discrete, heterogeneous channels to communicate between the transmitter and the receiver. For this example, $n = 3$, $\sigma = 1$ listeners, and $\rho = 1$ disruptors. One difference between the single channel case and this case featuring three channels is the AL. The original message is fragmented into three equal portions, f_1, f_2, and f_3, which are each transmitted twice as follows: $\{f_1, f_2\}$ on Channel 1, $\{f_2, f_3\}$ on Channel 2, and $\{f_3, f_1\}$ on Channel 3. For this case $AL = 0.66$, and $DF = 2$ (because each fragment is sent twice). Table 4 shows the calculated metrics for the network with various probability of interception and fixed values for probability of decryption and decoding. Additionally, Table 4 shows that the network has various probabilities of injection with fixed values for probability of noise and suppression.

The QoSS is similar to that of Figure 13 with some minor variations in the characteristics to highlight the possibility that the three channels cross different network

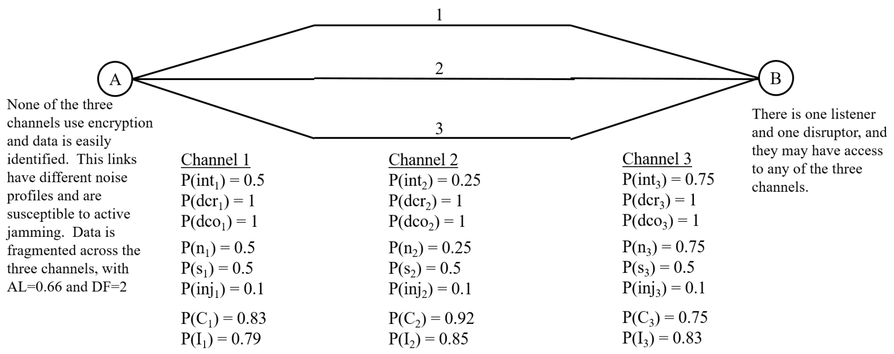


Figure 16. A Three-Channel, Point-to-Point Network with QoSS Metrics.

Table 4. Input and Output Values for a Three-Channel Architecture.

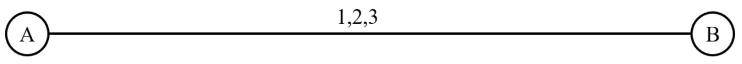
$channel\ (n)$	$P_n(int)$	$P_n(dcr)$	$P_n(dco)$	$P_n(l)$	$P_n(C)$	$P_n(n)$	$P_n(s)$	$P_n(inj)$	$P_n(c)$	$P_n(I)$
1	0.5	1	1	0.1667	0.8333	0.25	1	0.33	0.2219	0.7781
2	0.75	1	1	0.2500	0.7500	0.25	1	0.4	0.2248	0.7752
3	0.25	1	1	0.0833	0.9167	0.25	1	0.26	0.2191	0.7809
Avg	0.5	1	1	0.1667	0.8333	0.25	1	0.33	0.2219	0.7781

infrastructures. Channel 1 has identical input factors to the single-channel network as demonstrated in Table 3; however, the message is fragmented across multiple channels, which causes the probability of confidentiality and probability of integrity to increase, not only for Channel 1, but for each channel in the network[15]. The average probability of confidentiality is $P(C) = 0.83$ even without encryption, indicating that fragmenting data across the multiple channels improves the probability of confidentiality and over-all QoSS metrics, partially mitigating the lack of encryption.

The simplified equivalent of the three-channel system, shown in Figure 17 is represented as a single-channel network. The single-channel network helps illustrate the average QoSS metrics of $P(C) = 0.83$ and $P(I) = 0.79$. Compared to the single-channel point-to-point example in Figure 13 or the single-channel three-hop equivalent example in Figure 15, this system has a significantly improved probability of confidentiality and probability of integrity despite (and at the cost of) being more

[15]In a real communication architecture, all three channels would likely have more similar characteristics.

complex.



$P(C_{1,2,3}) = (P(C_1)+P(C_2)+P(C_3))/3 = 0.833$
$P(I_{1,2,3}) = (P(I_1)+P(I_2)+P(I_3))/3 = 0.789$

Figure 17. A Single-Channel, Point-to-Point Equivalent of the Three-Channel, Point-to-Point Network.

3.5.4 3-Channel, 3-Hop Network

The fourth example incorporates the previous two examples to create a more realistic architecture, to highlight the order needed to reduce the complexity, and to demonstrate the analysis process for the QoSS metrics [70]. The fourth example features three channels, two of which have two intermediate nodes and the third with only one intermediate node, thus creating a three-channel, multi-hop network, as shown in Figure 18. In the same manner in which the network shown in Figure 14 may utilize three different interconnecting channels for each of the intermediate links and the network shown in Figure 16 may utilize three different interconnecting paths or media for each channel, so too may Figure 18 utilize different interconnecting channels, which may feature different distances, architectures, or technologies, for each of the eight different, independent links.

In this example, and providing continuity with the example network shown in Figure 14, there is $\rho = 1$ listener and $\sigma = 1$ disruptor with access to the network, and they may have access to any of the channels. The data are fragmented across the three channels such that the AL, the average percentage of the message on any of the channels, is 0.66 and the DF, the average number of times a given fragment is transmitted, is 2. The QoSS metrics are again similar to all of the previous examples with some minor variations in the characteristics to highlight the possibility that each of the eight different, independent links feature different distances, architectures, or

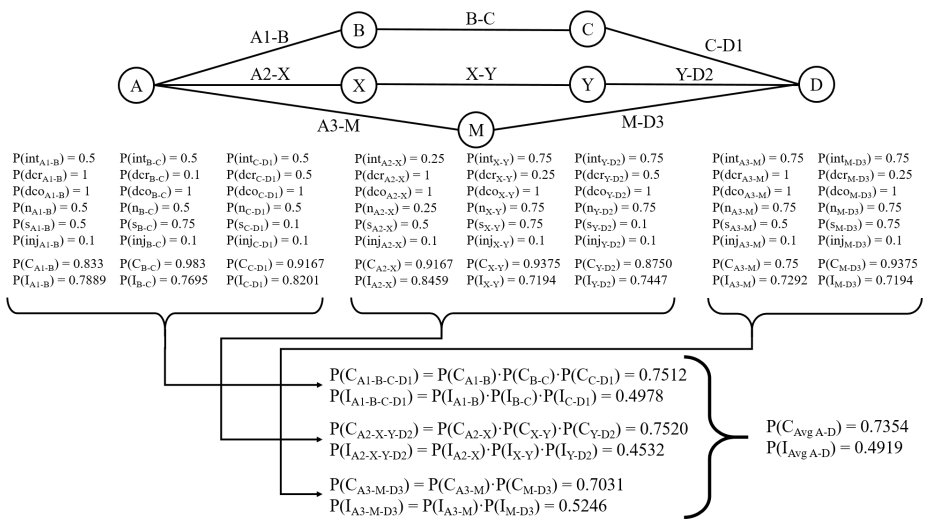

A1-B
B-C
C-D1
A2-X
X-Y
Y-D2
A3-M
M-D3
A
B
C
X
Y
M
D
$P(int_{A1-B}) = 0.5$ $P(dcr_{A1-B}) = 1$ $P(dco_{A1-B}) = 1$ $P(n_{A1-B}) = 0.5$ $P(s_{A1-B}) = 0.5$ $P(inj_{A1-B}) = 0.1$
$P(C_{A1-B}) = 0.833$ $P(I_{A1-B}) = 0.7889$
$P(int_{B-C}) = 0.5$ $P(dcr_{B-C}) = 0.1$ $P(dco_{B-C}) = 1$ $P(n_{B-C}) = 0.5$ $P(s_{B-C}) = 0.75$ $P(inj_{B-C}) = 0.1$
$P(C_{B-C}) = 0.983$ $P(I_{B-C}) = 0.7695$
$P(int_{C-D1}) = 0.5$ $P(dcr_{C-D1}) = 0.5$ $P(dco_{C-D1}) = 1$ $P(n_{C-D1}) = 0.5$ $P(s_{C-D1}) = 0.1$ $P(inj_{C-D1}) = 0.1$
$P(C_{C-D1}) = 0.9167$ $P(I_{C-D1}) = 0.8201$
$P(int_{A2-X}) = 0.25$ $P(dcr_{A2-X}) = 1$ $P(dco_{A2-X}) = 1$ $P(n_{A2-X}) = 0.25$ $P(s_{A2-X}) = 0.5$ $P(inj_{A2-X}) = 0.1$
$P(C_{A2-X}) = 0.9167$ $P(I_{A2-X}) = 0.8459$
$P(int_{X-Y}) = 0.75$ $P(dcr_{X-Y}) = 0.25$ $P(dco_{X-Y}) = 1$ $P(n_{X-Y}) = 0.75$ $P(s_{X-Y}) = 0.75$ $P(inj_{X-Y}) = 0.1$
$P(C_{X-Y}) = 0.9375$ $P(I_{X-Y}) = 0.7194$
$P(int_{Y-D2}) = 0.75$ $P(dcr_{Y-D2}) = 0.5$ $P(dco_{Y-D2}) = 1$ $P(n_{Y-D2}) = 0.75$ $P(s_{Y-D2}) = 0.1$ $P(inj_{Y-D2}) = 0.1$
$P(C_{Y-D2}) = 0.8750$ $P(I_{Y-D2}) = 0.7447$
$P(int_{A3-M}) = 0.75$ $P(dcr_{A3-M}) = 1$ $P(dco_{A3-M}) = 1$ $P(n_{A3-M}) = 0.75$ $P(s_{A3-M}) = 0.5$ $P(inj_{A3-M}) = 0.1$
$P(C_{A3-M}) = 0.75$ $P(I_{A3-M}) = 0.7292$
$P(int_{M-D3}) = 0.75$ $P(dcr_{M-D3}) = 0.25$ $P(dco_{M-D3}) = 1$ $P(n_{M-D3}) = 0.75$ $P(s_{M-D3}) = 0.75$ $P(inj_{M-D3}) = 0.1$
$P(C_{M-D3}) = 0.9375$ $P(I_{M-D3}) = 0.7194$
$P(C_{A1-B-C-D1}) = P(C_{A1-B})\cdot P(C_{B-C})\cdot P(C_{C-D1}) = 0.7512$
$P(I_{A1-B-C-D1}) = P(I_{A1-B})\cdot P(I_{B-C})\cdot P(I_{C-D1}) = 0.4978$
$P(C_{A2-X-Y-D2}) = P(C_{A2-X})\cdot P(C_{X-Y})\cdot P(C_{Y-D2}) = 0.7520$
$P(I_{A2-X-Y-D2}) = P(I_{A2-X})\cdot P(I_{X-Y})\cdot P(I_{Y-D2}) = 0.4532$
$P(C_{A3-M-D3}) = P(C_{A3-M})\cdot P(C_{M-D3}) = 0.7031$
$P(I_{A3-M-D3}) = P(I_{A3-M})\cdot P(I_{M-D3}) = 0.5246$
$P(C_{Avg\ A-D}) = 0.7354$
$P(I_{Avg\ A-D}) = 0.4919$

Figure 18. A Three-Channel, Three-Hop Network with QoSS Metrics and Equivalent Calculations.

technologies.

The first step, as detailed in Section 3.4.3, calculates the QoSS metrics for each independent link. The next step eliminates the intermediary nodes from each of the three channels. The final step consolidates the multi-channel architecture into one point-to-point connection. With the single point-to-point connection, it becomes trivial to compare the QoSS characteristics of Figure 18 to the more simple system architectures presented in Figures 13, 14, and 16 and their respective single-channel, point-to-point equivalents.

Based on the comparison of the equivalent QoSS metrics of Figure 18 to the QoSS metrics of Figures 13, 14, and 16, there are several details that must be highlighted. The example that has, after reduction, the highest probability of confidentiality and probability of integrity is Figure 16 primarily because it employs three channels that feature message fragmentation and duplication across the three available channels. It is observed that simply splitting the data across the available channels, with some duplication, provides a significant increase to the available security of the system, despite the number of potential listeners and disruptors. Conversely, the example that has, after reduction, the lowest probability of confidentiality and probability of integrity is Figure 14 because it employs a single channel with two intermediate nodes. The intermediate nodes provide opportunities for adversarial intervention in the transmission. Furthermore, since there is only one channel, there is no benefit of message fragmentation and duplication, and any listener or disruptor has just the one channel to target. As an adequate median example, Figure 18 demonstrates the improved QoSS metrics of utilizing multiple channels and message fragmentation and duplication across the three available channels while also demonstrating some of the potentially degraded QoSS because of the multiple intermediary nodes.

3.5.5 Eight-Channel Network

The fifth and final example presents a communication architecture with eight discrete, heterogeneous channels [62]. In this example, $n = 8$, $\sigma = 3$ listeners, and $\rho = 3$ disruptors. The original message is fragmented into eight equal portions, $\{f_1, f_2, ..., f_8\}$, of which $\{f_1, f_2\}$ are transmitted on Channel 1, $\{f_2, f_3\}$ on Channel 2, $\{f_3, f_4\}$ on Channel 3, and so on. Here, $AL = 0.25$ because each channel transmits exactly a quarter of the original message M, and $DF = 2$ because each fragment is sent twice. Table 5 shows the theorized input for the eight-channel network.

Of particular note, Table 5 has the same input as Table 4 for Channels 1 through 3, and other values for Channels 4 through 8, although with different results[16]. The only difference from the three-channel case is that, with eight channels, the message is fragmented across more channels, causing the confidentiality and integrity to increase. The average values for $P(int)$, $P(dcr)$, and $P(dco)$ are the same for the single-channel, three-channel, and eight-channel networks, although the average $P(l)$ and $P(C)$ are notably different.

[16]As in the Three-Channel example, a real communication system would likely have channels with similar characteristics.

Table 5. Input and Output Values for an Eight-Channel Architecture.

channel (n)	$P_n(int)$	$P_n(dcr)$	$P_n(dco)$	$P_n(l)$	$P_n(C)$	$P_n(n)$	$P_n(s)$	$P_n(inj)$	$P_n(c)$	$P_n(I)$
1	0.5	1	1	0.1875	0.8125	0.25	1	0.33	0.2402	0.7598
2	0.75	1	1	0.2813	0.7188	0.25	1	0.4	0.2414	0.7586
3	0.25	1	1	0.0938	0.9063	0.25	1	0.26	0.2389	0.7611
4	0.2	1	1	0.0750	0.9250	0.25	1	0.05	0.2353	0.7647
5	0.35	1	1	0.1313	0.8688	0.25	1	0.1	0.2361	0.7639
6	0.4	1	1	0.1500	0.8500	0.25	1	0.2	0.2379	0.7621
7	0.7	1	1	0.2625	0.7375	0.25	1	0.6	0.2449	0.7551
8	0.85	1	1	0.3188	0.6813	0.25	1	0.7	0.2467	0.7533
Avg	0.5	1	1	0.1875	0.8125	0.25	1	0.33	0.2402	0.7598

As expected, the single-channel architecture has the lowest calculated QoSS values. With a slightly higher percentage of listeners, the eight-channel network has a slightly higher $P(l)$ and correspondingly lower $P(C)$ than the three-channel network[17]. Similarly, the average values for $P(n)$, $P(s)$, and $P(inj)$ are the same for the single-channel, three-channel, and eight-channel networks, yet the $P(c)$ and $P(I)$ are significantly different.

3.6 Simulation Development to Demonstrate the Metrics

As the final step in defining the security-focused protocol, a representative simulation of communication architectures with varying complexity and featuring multiple heterogeneous channels and intermediary nodes was developed. This architecture is in contrast to separately routing critical information over secure channels or maintaining multiple other levels of security, as proposed in [72], which is similar to the multi-level security model presented in [73]. Typically, communication systems are modeled and analyzed using Markov chains because the mathematical models are tractable. When the real communications systems vary significantly from the assumptions required for such analysis, simulations can provide a meaningful way to

[17]Note the number of channels with respect to Equation (8) for this multi-channel space. For the eight-channel network, $n = 8$ even though seven channels would be sufficient based on $\sigma = 3$ listeners, $\rho = 3$ disruptors, and Equation (8).

obtain insight into the performance of such systems. Similarly, due to the difficulty in defining security, understanding security within an architecture similar to the one shown in Figure 5, or specifically Figure 12, for both authorized and unauthorized users becomes intractable. One goal of the simulation environment is to quantify confidentiality and integrity with an eye toward developing more secure communication architectures through the creation of a comprehensive protocol that splits data across multiple channels with varying amounts of cross-channel duplication, checksum, or CRC overhead. Messages must also be received correctly and discourage eavesdropping despite the possible malicious injections and recognizing which channel is being targeted. The other goal of the simulation is to develop understanding about integrity, thus preventing malicious injection or jamming, or at least easily recovering from it. This simulation represents one of many possible approaches to implementing and quantifying the security available within a communication network.

By simulating various architectures ranging from single point-to-point connections to multi-channel, multi-hop architectures, and by varying the data transmitted across those channels, it is expected to become clear that the probability of leakage and its constituent metrics have a complex relationship to system confidentiality. The simulations are expected to provide measurable evidence that a multi-channel architecture can limit the useful information an eavesdropper receives. The probabilities of interception and decoding, and the amount of fragmentation and duplication across channels should demonstrate useful emergent performance characteristics of confidentiality from both the authorized user's and the adversarial listener's perspective. It is also expected that the probability of corruption, specifically the probability of disruption and the probability of injection, will demonstrate useful insights about the integrity of data being transmitted across the system.

3.6.1 The Simulation Environment

The simulation environment used in this work was created using OMNeT++, the "extensible, modular, component-based C++ simulation library and framework" [74]. To build the communication architecture required developing five different modules using the NEtwork Description (NED) language and saved as .ned files. NED is the topology description language of OMNeT++, and it allows the declaration of simple modules which can then be connected and assembled into larger, complex compound modules [75]. The five modules created are the *source*, *dissembler*, *node*, *assembler*, and *sink*, effectively recreating the modules shown in Figure 5. Each of these .ned files perform specific functions within the larger simulation world. The *source* loads a text file into memory and initiates the transmission across the network. The *source* is the stand-in for Alice, or more appropriately, the application layer software that will generate data. The *dissembler* performs the various packetization processes. These processes include fragmentation and duplication of the original message per the desired policy, generating the checksums and CRC, and loading them into the various transmission packets with the policy and packet number. These packet are then pushed out into the remaining architecture. Again, based on policy, a varying number of intermediate nodes are instantiated to relay messages to the next module. The *nodes* function just like relay nodes, or routers, in modern networks, receiving data, processing it, and sending it on to the next destination. The *assembler* performs the complimentary processes to the *dissembler*. These processes include reassembly and validation of the received data. In cases where data is lost or modified, the *assembler* also attempts to recover that data using the various checksums and CRC. If errors are detected by the *assembler*, a re-transmission request or feedback messages are triggered and a log of the error is recorded. The reassembled message is then forwarded to the *sink* to be compared with the original message and generate the various metrics for final evaluation. The *sink* represents Bob, or the application layer software that will ultimately receive the data. Having the architecture decomposed into these five nodes provides sufficient flexibility, with the most complex functions contained in the *dissembler* and *assembler*.

The top-level .ini file initializes the entire architecture and provides all the connective logic that instantiates the five .ned files. It links together the *source* to the *dissembler*, the *assembler* to the *sink*, and based on the number of desired channels and intermediate hops, joins the *dissembler* to the *assembler* via the desired number of *nodes*, if any. It also provides the various policies, scenarios, and test characteristics for the various architectures.

Finally, the in-depth functions of each of the five modules are programmed in C++ and stored in .cc files, respectively. As stated above, the modules have their own specific functions and processes. As the .ini file initialize the architecture and links the .ned files, the .ned files initialize the respective .cc files, thus instantiating all of the lowest level tasks. Using the architecture in this way allowed for all of the lowest level processes to be completely customized at the bit level, which also abstracts away details at the highest level of the simulation. The bit-level functions do not need to be observed directly when exercising the individual simulations.

3.6.2 Policies and Scenarios

In broad terms, the policies are applied to the chosen architectural model and dictate how the messages are transmitted. Each policy defines the amount of fragmentation and duplication, if any, performed at the dissembler. In the single-channel models, the policies generally behave the same, since there is not an option for transmitting data fragments across multiple channels. However, fragmenting the data is still an option even in single-channel models, as those fragments may be duplicated.

The possible combinations with respect to the size of fragments, byte-wise to bit-wise, the method of distributing the fragments across available channels, and the number of times fragments are duplicated, are numerous. For this simulation, we have limited the policies to fragments of 8-bytes, 4-bytes, 2-bytes, 1-byte, 4-bit, and 1-bit. In general, duplication is limited to no duplication ($DF = 1$) or one duplicate copy sent ($DF = 2$), although one policy, an early developmental artifact, sends the same data across all channels ($DF = n$). The policies are detailed in Chapter IV, Section 3.6.6. Although it would certainly provide valuable insight, a thorough inquiry into the optimal size, distribution, and duplication of fragments across a set number of channels is beyond the scope of this research.

The previous scenarios represent all of the architectural models currently available in the simulation environment. A total of twenty-five scenarios exist, although more are possible. A maximum of $n = 5$ channels may be instantiated by the simulation environment. Beyond five channels, the added complexity of design did not appear to provide any additional insights about architecture. Modifying the simulation to have more than five channels is relatively trivial, although changing the communication policies to harness the additional channels would require significant coding. The same is true for the number of intermediate hops. A maximum of four nodes are available per channel, corresponding to a network architecture with a maximum five hops per channel. Additional nodes also did not appear to provide any additional insights into performance. Adding more nodes to the simulation is equally trivial, and does not require significant changes to the code either. Therefore, any architectural combination from $n = 1$ channel and zero nodes to $n = 1$ channel and four nodes to $n = 5$ channels and four nodes may be observed. Any further complexity of the architecture, for example channels splitting into additional channels between nodes, may be decomposed into one of these available scenarios for easy observation.

3.6.3 Experimental Test Cases

The basic test case is the transmission of a text string. For this exercise, a paragraph was desired to be recognizable, long enough to provide a reasonable test to the architecture, and an adequate variation of characters. The selected text is the first paragraph from J.R.R. Tolkien's book *The Hobbit*, and reads:

> In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort. It had a perfectly round door like a porthole, painted green, with a shiny yellow brass knob in the exact middle. The door opened on to a tube-shaped hall like a tunnel: a very comfortable tunnel without smoke, with panelled walls, and floors tiled and carpeted, provided with polished chairs, and lots and lots of pegs for hats and coats - the hobbit was fond of visitors. The tunnel wound on and on, going fairly but not quite straight into the side of the hill - The Hill, as all the people for many miles round called it - and many little round doors opened out of it, first on one side and then on another. [76]

The individual characters of the text, as they are loaded by the *source*, are encoded as 8-bit ASCII strings. The ASCII-encode text provides direct, easily observed insight regarding the integrity of the transmitted data. If one bit is changed, then the change is obvious because that particular character changes. In some cases, this changes the ASCII character from a letter or number to a control character or some other extended encoding character. The basic test case is also useful because the data that is intercepted by the eavesdropper can be identified easily. Based on the size of the data fragments, the intercepted data can be located to which portion of the message they came from.

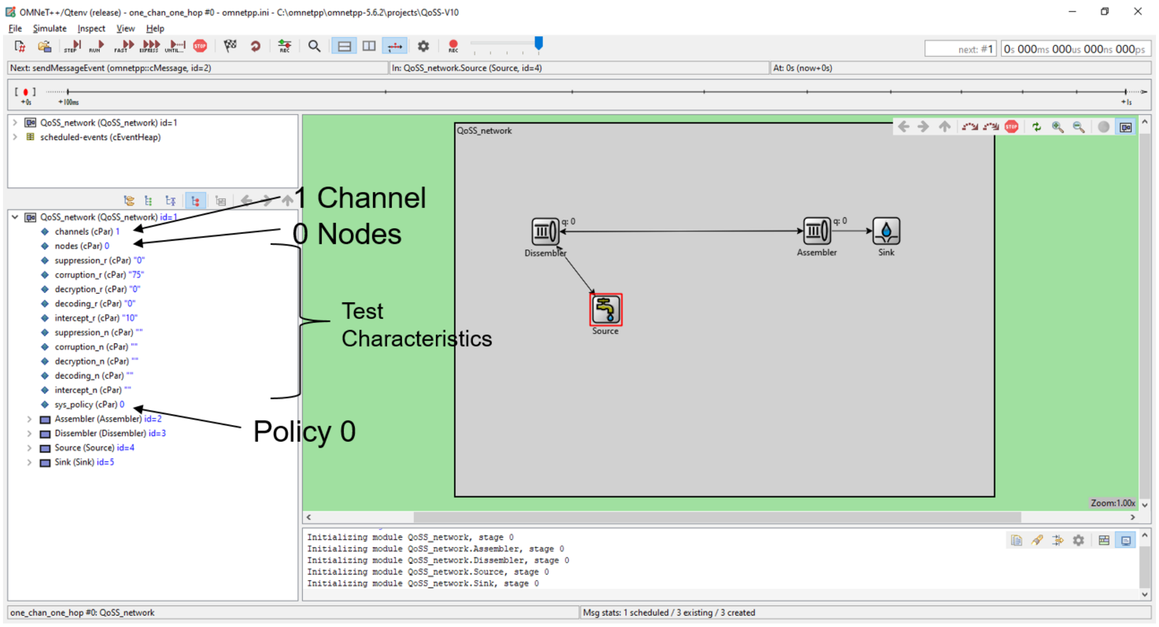
A secondary test case was considered. In this case, a small image file, either .bmp or .jpg, would be transmitted across the architecture. This test case would also easily provide insight about the integrity of the data being transmitted because any modified bits would ultimately change the image in some manner. The entire image may be corrupting completely if the corrupted bits were part of the header data. Otherwise, corrupted bits would be presented as different colored pixels or some other change that is easily observed. This test case does not provide as much insight about the confidentiality of the data. As the intercepted data is likely only a portion of the original image, it is difficult to recreate a standard image file from only a portion of the data. This case was ultimate not exercised.

The final set of variables to explore are the QoSS metrics, also referred to as test characteristics. These test characteristics are the exact numbers used in the calculations developed in Section 3.4.2 and demonstrated in Section 3.5. For example, the simulation environment presented in Figure 19 shows a single-channel, single-hop architectural model. The test characteristics are:

```
[Config one_chan_one_hop]
#network = one_chan_three_hop
network = QoSS_network
**.channels = 1
**.nodes = 0
**.decryption_r = ("0")
**.decoding_r = ("0")
**.intercept_r = ("10")
**.suppression_r = ("0")
**.corruption_r = ("75")
**.decryption_n = ("")
**.decoding_n = ("")
**.intercept_n = ("")
**.suppression_n = ("")
**.corruption_n = ("")
description = "One Channel with Zero Intermediate Nodes
   / One Hop"
seed-set = ${5}
```

where the probabilities of decryption, decoding, interception, suppression, and corruption can have test values applied to them, thus matching the QoSS performance metrics. The test characteristics are also separated as those for the *dissembler*, (_r, residual artifact when originally referred to as the *router*) and *nodes*, (_n). Since there are no intermediary nodes, the values next to the *node* test characteristics are null.

This model is also set up to exercise Policy 0.[18].



OMNeT++/Qtenv (release) - one_chan_one_hop #0 - omnetpp.ini - C:\omnetpp\omnetpp-5.6.2\projects\QoSS-V10

Next: sendMessageEvent (omnetpp::cMessage, id=2)
In: QoSS_network.Source (Source, id=4)
At: 0s (now+0s)

```
QoSS_network (QoSS_network) id=1
scheduled-events (cEventHeap)

QoSS_network (QoSS_network) id=1
    channels (cPar) 1
    nodes (cPar) 0
    suppression_r (cPar) "0"
    corruption_r (cPar) "75"
    decryption_r (cPar) "0"
    decoding_r (cPar) "0"
    intercept_r (cPar) "10"
    suppression_n (cPar) ""
    corruption_n (cPar) ""
    decryption_n (cPar) ""
    decoding_n (cPar) ""
    intercept_n (cPar) ""
    sys_policy (cPar) 0
  Assembler (Assembler) id=2
  Dissembler (Dissembler) id=3
  Source (Source) id=4
  Sink (Sink) id=5
```

```
Initializing module QoSS_network, stage 0
Initializing module QoSS_network.Assembler, stage 0
Initializing module QoSS_network.Dissembler, stage 0
Initializing module QoSS_network.Source, stage 0
Initializing module QoSS_network.Sink, stage 0
```

one_chan_one_hop #0: QoSS_network
Msg stats: 1 scheduled / 3 existing / 3 created

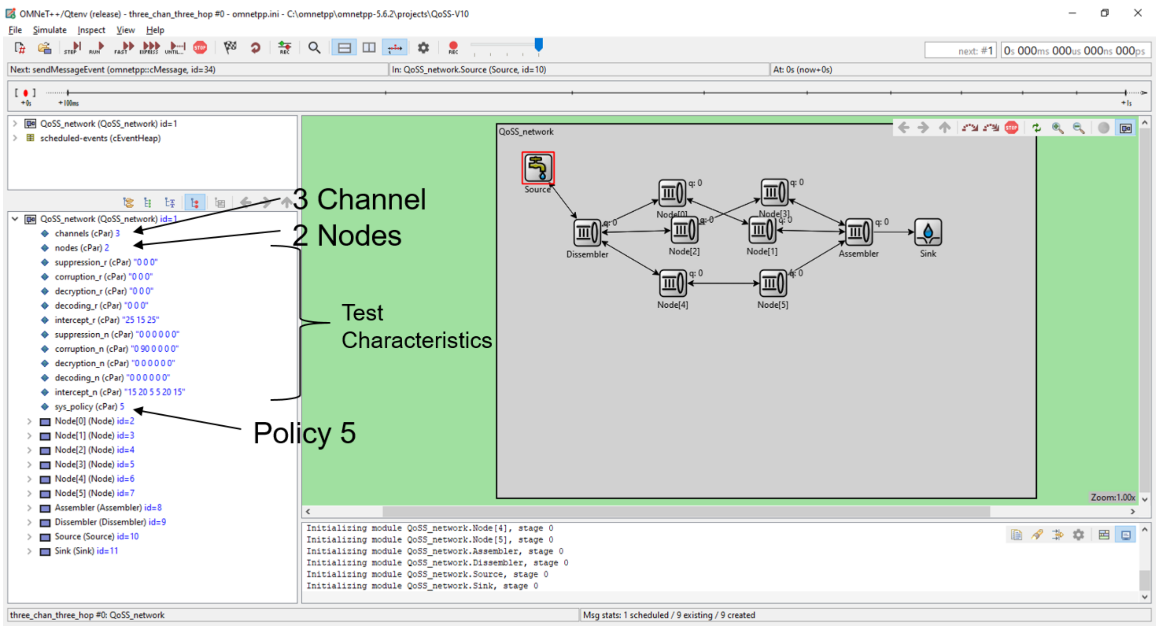
Figure 19. Single-channel, Single-hop Architectural Model Simulation Environment.

By comparison, Figure 20 shows a three-channel, three-hop architectural model. In this case, six *nodes* are instantiated between the *dissembler* and *assembler*. For this example test, the characteristics are:

```
[Config three_chan_three_hop]
#network = one_chan_three_hop
network = QoSS_network
**.channels = 3
**.nodes = 2
**.decryption_r = ("0 0 0")
**.decoding_r = ("0 0 0")
**.intercept_r = ("25 15 25")
**.suppression_r = ("0 0 0")
**.corruption_r = ("0 0 0")
**.decryption_n = ("0 0 0 0 0 0")
**.decoding_n = ("0 0 0 0 0 0")
**.intercept_n = ("15 20 5 5 20 15")
**.suppression_n = ("0 0 0 0 0 0")
**.corruption_n = ("0 90 0 0 0 0")
description = "Three Channels with Two Intermediate
   Nodes / Three Hops"
seed-set = ${5}
```

[18]The complete description of the policies and their function are discussed at length in Chapter IV, Section 3.6.6.

where the probabilities of decryption, decoding, interception, suppression, and corruption again have test values applied to them. For this case, though, the characteristics are an array of values. The test characteristics for the *dissembler* is a three-item array, one characteristics for each of the three channels exiting the *dissembler*. Similarly, the test characteristics for the *nodes* are a six-item array, one characteristics for each of the six possible hops (three channels with two nodes each). This model is also set up to exercise Policy 5.


OMNeT++/Qtenv (release) - three_chan_three_hop #0 - omnetpp.ini - C:\omnetpp\omnetpp-5.6.2\projects\QoSS-V10

Next: sendMessageEvent (omnetpp::cMessage, id=34) | In: QoSS_network.Source (Source, id=10) | At: 0s (now+0s)

```
QoSS_network (QoSS_network) id=1
scheduled-events (cEventHeap)

QoSS_network (QoSS_network) id=1
    channels (cPar) 3
    nodes (cPar) 2
    suppression_r (cPar) "0 0 0"
    corruption_r (cPar) "0 0 0"
    decryption_r (cPar) "0 0 0"
    decoding_r (cPar) "0 0 0"
    intercept_r (cPar) "25 15 25"
    suppression_n (cPar) "0 0 0 0 0 0"
    corruption_n (cPar) "0 90 0 0 0 0"
    decryption_n (cPar) "0 0 0 0 0 0"
    decoding_n (cPar) "0 0 0 0 0 0"
    intercept_n (cPar) "15 20 5 5 20 15"
    sys_policy (cPar) 5
  Node[0] (Node) id=2
  Node[1] (Node) id=3
  Node[2] (Node) id=4
  Node[3] (Node) id=5
  Node[4] (Node) id=6
  Node[5] (Node) id=7
  Assembler (Assembler) id=8
  Dissembler (Dissembler) id=9
  Source (Source) id=10
  Sink (Sink) id=11
```

```
Initializing module QoSS_network.Node[4], stage 0
Initializing module QoSS_network.Node[5], stage 0
Initializing module QoSS_network.Assembler, stage 0
Initializing module QoSS_network.Dissembler, stage 0
Initializing module QoSS_network.Source, stage 0
Initializing module QoSS_network.Sink, stage 0
```

three_chan_three_hop #0: QoSS_network | Msg stats: 1 scheduled / 9 existing / 9 created

Figure 20. Three-channel, Three-hop Architectural Model Simulation Environment.

With that, the simulation environment is ready to be exercised and generate insights regarding the possible improvement to confidentiality and integrity of messages sent across the available channels.

3.6.4 Packet Structure

Before the architecture can be exercised and observed in the simulation environment, there is a set of design aspects that are not fully developed. Those aspects are the packet structure and the precise methods of scrambling or duplicating that are applied as detailed in the policies. Much of the packet development was iterative, using some of the initial results and tests cases as feedback. The chosen packet structure is a result of many iterative changes, based on the evolving needs within the simulation for improved feedback, more transmitted data, improved error detection, and error correction.

The policy number is assigned an eight-bit field. This allows for up to 256 possible policies, an adequately sized number for this initial research, but one that may need to change again based on possible custom policies. The concept of custom policies, and the ability to change policies on the fly, is included in the protocol requirements as the ability to scatter or shuffle data across available channels and the ability to increase or decrease duplication. Defining custom policies is a derived requirement that adds significant flexibility to the overall protocol.

The size of the transmitted data is set to eight bytes, for a total of eight ASCII-encoded characters. The size of the data field can increase further with future development, although the current size of 8-bytes is adequate for testing purposes. Increasing the size of the data field will provide numerous benefits, including greater throughput, optimization with lower-level protocols, and additional combinations of fragmentation and duplication.

A 10-bit packet number field allows up to 1024 unique packet numbers. This field may also need to increase as the size and complexity of the protocol increase, thus improving the fidelity of the feedback mechanism to identify bad packets. For initial testing purposes, the 10-bit packet number field is adequate.

A 22-bit CRC was selected to create a comparatively standard 128-bit data packet.

Finally, three fields of checksums were included in the packet to allow for cross-checking of the multi-channel packets, each being 8-bits in length. The final packet structure is shown in Figure 21. For the initial set of experiments using OMNeT++, the 128-bit packet provided all of the needed error detection and correction capabilities, packet numbering, and policies to function. It should be noted that there are 64-bits of overhead for each packet, thus having 100% overhead for the 64-bits of data transmitted per packet. The efficiency of this packet structure is admittedly low, but useful for this initial level of development. Extended development of a robust protocol would include much larger packet sizes, leading to a much lower overhead.

8 bits	64 bits	8 bits	8 bits	8 bits	10 bits	22 bits
policy	data	CS1	CS2	CS3	packet #	CRC

Figure 21. The Final Simulation Packet Structure.

3.6.5 Checksums and CRCs

Having a CRC in the data packet to perform error checking was always part of the design for the simulation environment. However, as the size of the packet changed and the amount of payload data increased, the requirements for error-checking fidelity also changed. Based on the 64-bits of message data, plus an additional 42-bits of overhead, the CRC would need to be able to perform error checking on 106 bits of payload. However, knowing that the message data field may increase in size, and using the table available at [77], it was determined that a CRC of 22-bits offered multiple options. Firstly, a CRC of 22-bits combined with the other 42-bits of overhead, yielded a total overhead of 64-bits, a relatively standard size of data. Second, a 22-bit CRC provides a wide range of Hamming distance (HD) options, which corresponds to the minimum number of bit errors that will be undetected. As there will also be checksums in the packet and the lower-layer error detection and correction algorithms will be used, this protocol will not require a $HD = 11$ or $HD = 12$. This system simply does not need that high level of error detection. A $HD = 4$ or $HD = 5$ is adequate to perform the gross error detection within the data packet. With that, the polynomial was selected to be $0x248794$ because it allows a maximum payload length of 2097129 bits, well over 262000 bytes per packet[19]. If too many bit errors were undetected, a $HD = 5$ is still an acceptable option as it allows over 253 bytes per packet, still within the range of valid Ethernet frame sizes.

Checksums in the data packets were introduced as a manner to perform cross-checking for errors across the available channels. Again using the table available at [77], it was determined that three checksums of 8-bits each offered an adequate amount of cross-checking ability, noting that the Hamming distance is not as critical in this application. The size of the message data field is 64-bits, potentially increasing in size as needed. Checksums 1 and 2, shown as CS1 and CS2 in Figure 21, provide a verification of the first and second half of the message data field, respectively. Therefore, the polynomial for checksums 1 and 2 was selected to be $0x83$ because it allows a maximum payload length of 119 bits at $HD = 4$. They were also assigned the same polynomial for ease of programming. Again, since this is for comparison between the various channels and not precisely as error detection, the payload size is not a critical requirement and may be exceeded. Data collisions may occur with larger sized message data fields, but since this is for comparison across channels, those collisions should prove to be exceptionally rare. Similarly, checksum 3, represented as CS3 in Figure 21, provides a gross packet comparison for the entire message data field. Therefore, as the payload of checksum 3 is twice as long as checksums 1 or 2, the polynomial was selected to be $0xE7$ because it allows a maximum payload length of 247 bits at $HD = 3$. Again, that maximum length may be exceeded as long as the possible, yet rare, data collisions are anticipated.

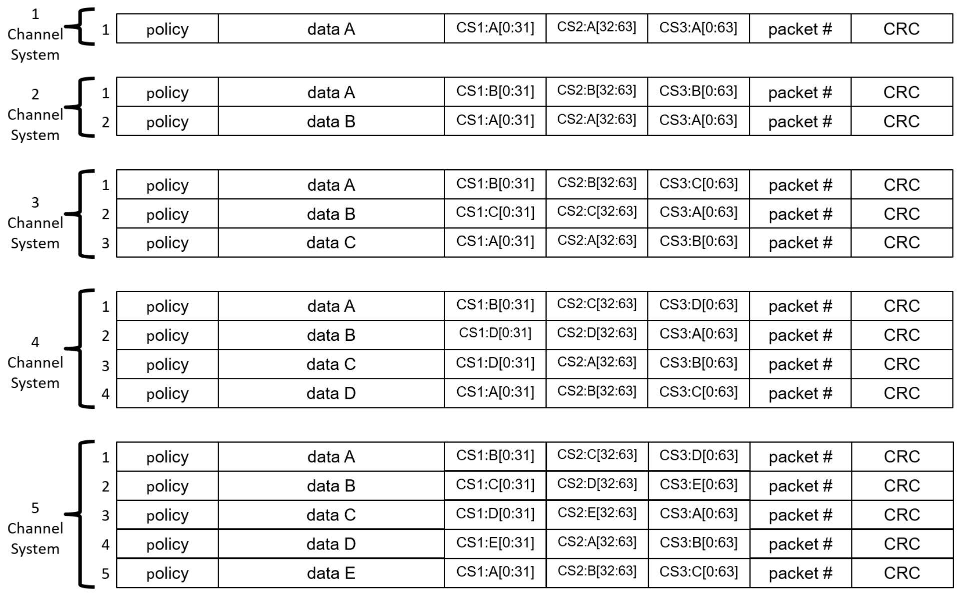
Where the use of the checksums becomes valuable is in the cross-checking of

[19]This number far exceeds the valid Ethernet frame size of between 64 and 1518 bytes, as specified by IEEE 802.3. The valid Ethernet frame size is a concern because this protocol, functioning at the session layer, must be compatible with the lower layer protocols and standard.

the data fields, as shown in Figure 22. In the single-channel system, the use of the checksums simply provides more granular error detection of the single packet; minimal value is added by the checksums. In the two-channel system, the checksums on channel 1 are calculated over the data field contained in channel 2, and the checksums on channel 2 are calculated over the data field in channel 1. It may be impossible to determine which is the bad channel in the presence of a sophisticated adversary who knows to change the data field and the checksum fields[20], but at least there is a clear indication when fields do not match. The cross-checking becomes more distributed as the number of channels increases which further aides in recognizing which channel, or channels, is being manipulated. For the 3-channel, 4-channel, and 5-channel systems, the checksums are distributed to provide information about two or three other channels, where possible. This directly aids in the ability to meet the design requirements to detect which is the bad channel (through comparing the checksum to the actual data across channels) and recover from maliciously injected bad data (once a bad channel is identified, if duplicated data is available, the data from the bad channel is ignored and filled in with verified good data). This will be explored in more depth in Section 4.4.

There is a risk to distributing too much information about the other channels. If an adversarial listener is able to intercept, decrypt, and decode a particular channel, they may be able to glean some information about the other channels or how many other channels are in use. The use of the different polynomials and the distribution of the resultant checksums is a method of obfuscating the possible number of channels, since the information does not provide any indication of the quantity. Assuming the adversarial listener has some sophistication, they would already know what the polynomials are and could calculate the checksums, revealing that the checksums do

[20]This also assumes that the sophisticated adversary who is attempting to inject malicious data also changes the CRC for the entire packet.



1 Channel System	1	policy	data A	CS1:A[0:31]	CS2:A[32:63]	CS3:A[0:63]	packet #	CRC
2 Channel System	1	policy	data A	CS1:B[0:31]	CS2:B[32:63]	CS3:B[0:63]	packet #	CRC
	2	policy	data B	CS1:A[0:31]	CS2:A[32:63]	CS3:A[0:63]	packet #	CRC
3 Channel System	1	policy	data A	CS1:B[0:31]	CS2:B[32:63]	CS3:C[0:63]	packet #	CRC
	2	policy	data B	CS1:C[0:31]	CS2:C[32:63]	CS3:A[0:63]	packet #	CRC
	3	policy	data C	CS1:A[0:31]	CS2:A[32:63]	CS3:B[0:63]	packet #	CRC
4 Channel System	1	policy	data A	CS1:B[0:31]	CS2:C[32:63]	CS3:D[0:63]	packet #	CRC
	2	policy	data B	CS1:D[0:31]	CS2:D[32:63]	CS3:A[0:63]	packet #	CRC
	3	policy	data C	CS1:D[0:31]	CS2:A[32:63]	CS3:B[0:63]	packet #	CRC
	4	policy	data D	CS1:A[0:31]	CS2:B[32:63]	CS3:C[0:63]	packet #	CRC
5 Channel System	1	policy	data A	CS1:B[0:31]	CS2:C[32:63]	CS3:D[0:63]	packet #	CRC
	2	policy	data B	CS1:C[0:31]	CS2:D[32:63]	CS3:E[0:63]	packet #	CRC
	3	policy	data C	CS1:D[0:31]	CS2:E[32:63]	CS3:A[0:63]	packet #	CRC
	4	policy	data D	CS1:E[0:31]	CS2:A[32:63]	CS3:B[0:63]	packet #	CRC
	5	policy	data E	CS1:A[0:31]	CS2:B[32:63]	CS3:C[0:63]	packet #	CRC

Figure 22. Cross-checking of Packets With Checksums, For Data Validation and Data Correction.

not match the data file on channel they have intercepted. This is the only indication they have, that there is more than one channel in use.

3.6.6 Fragmentation and Duplication Within the Policies

The sole purpose of requiring checksums in the data packet is to perform cross-channel packet verification of the data. This has the two-fold benefit of detecting when a channel, or possibly multiple channels, have been tampered with, and providing a means toward error correction within the message data field. However, error correction is only feasible when fragment duplication is employed across multiple channels. After initial experiments of transmitting data with no duplication or all duplication showed numerous systemic inadequacies, a larger set of policies were developed that appears to cover the requirements of fragmentation and duplication across multiple channels, while also aiding in detecting exploited channels, and correcting those channels where possible.

A total of nine policies were programmed into the simulation environment, although the framework is in place to add many more. To add more policies, the C++ files for the *dissembler* and *assembler* need have the data-handling instructions coded for the five channel-count cases. These policies represent most of the basic performance aspects and several of the possible corner cases to be explored. Listed here are the twelve originally planned policies; policy numbers have no particular order or significance.

- **Policy 0**: No fragmentation, scrambling, or duplication ($DF = 1$). Eight 8-byte character fragments (a total of 64 bits of data, same as the message data field) are sent across all channels. This was one of the two original test policies.
- **Policy 1**: Data is split into 4-byte fragments. No duplication ($DF = 1$). One fragment (4 bytes) is sent across each channel in a round-robin manner until each channel's data field is filled with 8 bytes of data, i.e. two fragments.
- **Policy 2**: Data is split into 2-byte fragments. No duplication ($DF = 1$). One fragment (2 bytes) is sent across each channel in a round-robin manner until each channel's data field is filled with 8 bytes of data, i.e. four fragments. This policy was not implemented.
- **Policy 3**: Data is split into 1-byte fragments. No duplication ($DF = 1$). One fragment (1 byte) is sent across each channel in a round-robin manner until each channel's data field is filled with 8 bytes of data, i.e. eight fragments.
- **Policy 4**: Data is split into 4-bit fragments. No duplication ($DF = 1$). One fragment (4 bits) is sent across each channel in a round-robin manner until each channel's data field is filled with 8 bytes of data, i.e. 16 fragments.
- **Policy 5**: Data is split into 1-bit fragments. No duplication ($DF = 1$). One fragment (1 bit) is sent across each channel in a round-robin manner until each channel's data field is filled with 8 bytes of data, i.e. 64 fragments.

- **Policy 10**: No fragmentation. All data is duplicated on all channels ($DF = n$). The same 8-byte character fragment (a total of 64 bits of data, same as the message data field) is sent across all channels. This was one of the two original test policies.

- **Policy 11**: Data is split into 4-byte fragments. Each fragment is sent twice ($DF = 2$). One fragment (4 bytes) is sent across each channel in a round-robin manner, and then a duplicate of that fragment is sent on the next adjacent channel until each channel's data field is filled with 8 bytes of data, i.e. two fragments.

- **Policy 12**: Data is split into 2-byte fragments. Each fragment is sent twice ($DF = 2$). One fragment (2 bytes) is sent across each channel in a round-robin manner, and then a duplicate of that fragment is sent on the next adjacent channel until each channel's data field is filled with 8 bytes of data, i.e. four fragments.

- **Policy 13**: Data is split into 1-byte fragments. Each fragment is sent twice ($DF = 2$). One fragment (1 byte) is sent across each channel in a round-robin manner, and then a duplicate of that fragment is sent on the next adjacent channel until each channel's data field is filled with 8 bytes of data, i.e. eight fragments.

- **Policy 14**: Data is split into 4-bit fragments. Each fragment is sent twice ($DF = 2$). One fragment (4 bits) is sent across each channel in a round-robin manner, and then a duplicate of that fragment is sent on the next adjacent channel until each channel's data field is filled with 8 bytes of data, i.e. 16 fragments. This policy was not implemented.

- **Policy 15**: Data is split into 1-bit fragments. Each fragment is sent twice

($DF = 2$). One fragment (1 bit) is sent across each channel in a round-robin manner, and then a duplicate of that fragment is sent on the next adjacent channel until each channel's data packet is filled with 8 bytes of data, i.e. 64 fragments. This policy was not implemented.

Other policies are possible, although these implemented policies explore the most obvious or logical test cases. One set of variations can include the non-uniform fragmentation of data across the channels, e.g., a 3-byte fragment on channel 1, a 4-byte fragment on channel 2, and a 1-byte fragment on channel 3, although this policy would require careful reconstruction of the data at the assembler. Another set of variations can include duplicating different sized fragments on all of the channels, which could potentially improve the ability to reconstruct the original messages correctly while giving an eavesdropper more opportunities to capture data. Similarly, interleaving smaller fragments and duplicated fragments from other channels could be a more complex version of this policy. Another policy variation can change the order that the fragments are organized onto the available channels, e.g., changing endianness, order of fragments, or other methods of rudimentary scrambling. These other policies did not appear to add particular value or insight to the observations, but clearly added complexity to the reassembly process.

3.7 Summary

The framework for defining security-focused communication protocol is directly based on the model architecture, the set of QoSS metrics, and the simulation environment. These pieces form the foundational elements that, based on exercising the protocol with the simulation, may demonstrate one solution to quantifiable and provably secure communications. The next step in the development of the protocol is to thoroughly exercise the simulation, collect data and metrics, and perform analysis on the data to potentially observe unexpected emergent behavior.

IV. Observations and Analysis

4.1 Objective

The simulation environment created in Chapter III, Section 3.6.1 is based on the architectural model, the QoSS metrics, and the security-focused communication protocol proposed in the chapter. Variations of the data-handling policies and test scenarios provided compelling test cases to exercise the simulation, representing realistic combinations of the policies, characteristics, and number of channels and nodes. The simulation environment was exercised and various data were generated. An analysis of the various results is presented to show the possible improvements available to communications security provided by the security-focused protocol and architectural model.

4.2 Simulation Results

With the architecture defined, the QoSS metrics derived, and the simulation environment assembled, it is possible to set the test characteristics, inject the 8-bit ASCII-encoded messages, and analyze the results. The test cases exercise the architecture as a way to develop understanding about the security aspects of various communication architecture scenarios. The scenarios range from one through five channels and one through five possible network hops. One network hop represents a point-to-point network; five network hops is a network with four intermediary relay nodes. Additional nodes and channels beyond five did not appear to add insight. All emergent behaviors were expected to manifest with five or fewer channels or hops. Each communication link, for example **A1-D1**, **C-D2**, or **A3-X** in Figure 12, was programmed to have specific probabilities of interception, decryption, decoding, suppression, or injection, thus allowing the exploration of the experimental test characteristics and their effects on the transmitted message. The probability of noise was not included in the simulation environment for several reasons, including the fact that most communication systems employ error correction, and since noise is a natural phenomena, it would potentially distract from the analysis of the probability of corruption.

4.2.1 Test Case 1: 1 Channel, 3 Hops

The baseline case of one channel is somewhat trivial. The number of hops is effectively irrelevant, as is the amount of duplication or fragmentation built into the policy number, since there is only one path for the message to travel on. For this example, the test characteristics demonstrate a system that is fully monitored by an adversarial listener, and an adversarial disruptor also has access to the channel. For this test case, the probability of interception $P(int) = 0.90$, meaning 90% of all messages on that particular channel will likely be intercepted, and with the probabilities of decryption and decoding $P(dcr) = 1.0$, and $P(dco) = 1.0$, implying that anything the adversary intercepts is immediately decrypted and decoded into ASCII text. This is a key assumption about the adversary's capabilities, recognizing that data may or may not be ASCII text. Also, the probability of injection $P(inj) = 0.90$, meaning that the disruptor has the ability to maliciously change 90% of the data on that particular channel. Within the simulation environment, the test characteristics were initialized with the following .ini parameters:

```
[Config one_chan_three_hop]
#network = one_chan_three_hop
network = QoSS_network
**.channels = 1
**.nodes = 2
**.decryption_r = ("0")
**.decoding_r = ("0")
**.intercept_r = ("0")
**.suppression_r = ("0")
**.injection_r = ("0")
**.decryption_n = ("0 100")
**.decoding_n = ("0 100")
**.intercept_n = ("0 90")
**.suppression_n = ("0 0")
```

```
**.injection_n = ("0 90")
description = "One Channel with Two Intermediate Nodes /
    Three Hops"
seed-set = ${5}
```

Since there is only one channel to carry all the data, and assuming that there is $\sigma = 1$ listener and $\rho = 1$ disruptor, the adversarial listener should intercept on average 90% of the message and the adversarial disruptor should cause on average 90% of the final message to be corrupted. These two values represent the effective QoSS probability of confidentiality $P(C) = 0.1$, as calculated using Equation 9, and the effective QoSS probability of integrity $P(I) = 0.1$, as calculated using Equation 16. Extracting the observed confidentiality metric for this case is somewhat easier than with multiple channels because the amount of the intercepted message can be directly compared to the actual message, effectively counting the number of characters that are intercepted by the eavesdropper. Similarly, the observed integrity metric can be calculated by comparing the number of corrupted characters to the actual characters. The average results for this test case are shown in Table 6 for the probability of confidentiality, and in Table 7 for the probability of integrity.

Comparing to modern communications systems employing the typical single-channel architecture, remediation techniques for expected data integrity of 10% include multiple forms of error detection or correction based on bit errors, or even rerouting transmissions around poor connections. If a modern system is expected to have only 10% confidentiality, the primary method of defending against likely eavesdropping is encrypting the data. As will be demonstrated in the next test case, these do not need to be the only protection techniques.

4.2.2 Test Case 2: 3 Channels, 3 Hops, Policy 0

The test case of three channels and three hops is far more interesting because of the added complexities. To ease the comparison, the test characteristics remain the same and the set of adversarial listeners is only able to access one of the three channels such that the probability of interception for one channel is $P(int) = 0.90$, the probabilities of decryption and decoding are $P(dcr) = 1.0$ and $P(dco) = 1.0$, and the probability of injection are $P(inj) = 0.9$. The simulation environment is initialized with the following .ini parameters:

```
[Config three_chan_three_hop]
#network = one_chan_three_hop
network = QoSS_network
**.channels = 3
**.nodes = 2
**.decryption_r = ("0 0 0")
**.decoding_r = ("0 0 0")
**.intercept_r = ("0 0 0")
**.suppression_r = ("0 0 0")
**.injection_r = ("0 0 0")
**.decryption_n = ("0 100 0 0 0 0")
**.decoding_n = ("0 100 0 0 0 0")
**.intercept_n = ("0 90 0 0 0 0")
**.suppression_n = ("0 0 0 0 0 0")
**.injection_n = ("0 90 0 0 0 0")
description = "Three Channels with Two Intermediate Nodes /
    Three Hops"
seed-set = ${5}
```

This test configuration generates some rather compelling results. The primary differences highlighted by this specific test are sending data over three channels instead of the just one, as was presented in the previous test case. If Policy 0 is used to transmit the 8-byte fragments with no duplication, what is received at the assembler is:

> ********e in the ground ********ved a hobbit. No********y, dirty, wet ho********ed with the ends********s and an oozy sm******** yet a dry, bare********hole with nothin********to sit down on o********: it was a hobbi********and that means c********It had a perfect******** door like a por********ainted green, wi********ny yellow brass ********the exact middle. The door opened on to ********haped hall like ********: a very comfort********nel without smok********panelled walls, ********rs tiled and car********rovided with pol********airs, and lots a********of pegs for hats********ts - the hobbit ******** of visitors. Th******** wound on and on********fairly but not q********aight into the s********he hill - The Hi********ll the people fo********iles round calle********nd many little r********rs opened out of********st on one side a********on another.

which is the original message with about 8 of every 24 characters changed. The asterisks in the message represent bytes of data, i.e. characters, that were changed by the adversary. This is approximately a corruption factor of 33%[1]. Using Equation 16 and assuming that there are $\sigma = 3$ listeners and $\rho = 3$ disruptors, and with a duplication factor $DF = 1.0$ and average channel loading $AL = 0.3$, the effective QoSS probability of confidentiality is $P(C) = 0.7$. The difference between the calculated 0.7 and observed 0.66 is likely because of the variations caused by the random number generator used in triggering the probability code. If a larger population Monte Carlo simulation were performed, the observed and calculated would be expected to converge. These results are reminiscent of a RAID 0 configuration of three separate drives and one of the drives is removed from the array[2]. The key difference between this message and a RAID array is that the data here is ASCII-encoded English language text. The frequency that characters are used in the English language is well known, and based on the combinations of characters, the actual text could be determined with additional processing and analysis. The adversarial listener receives:

> In a ho there l t a nas le, fil of wor ell, no , sandy r to ea t-hole, omfort. ly roun thole, th a sh knob in . The d a tube- a tunne able tu e, with and flo peted, ished c nd lots was fon e tunne , going uite st ide of ll, as r many d it - ound do it, fi nd then

which is approximately 33% of the original message because the message was split into three equal channels[3]. If this intercepted message were superimposed upon the corrupted message, there would be near-perfect alignment, filling in the corrupted portion with what the eavesdropper intercepted. Using Equation 9 and assuming that there are $\sigma = 3$ listeners and $\rho = 3$ disruptors, and with a duplication factor

[1]Recall that probability of integrity, per Equation 5, is $P(I) = 1 - P(c)$.

[2]RAID 0 configurations feature striping, but no mirroring or parity. If any of the drives in an array fails, it causes the entire RAID 0 volume and all files contained therein to be lost because striping distributes the contents of each file among all drives in the array [78].

[3]Recall that probability of confidentiality, per Equation 3, is $P(C) = 1 - P(l)$.

$DF = 1.0$ and average channel loading $AL = 1.0$, the effective QoSS probability of integrity is calculated to be $P(I) = 0.7$. Again, the difference between the calculated 0.7 and observed 0.66 is in the random variation and the sample size of one. From the perspective of the RAID 0 configuration of three separate drives, these results are similar to what would be stored on one of the three drives [78]. In this case, there is no context about the size or location of the fragments that would be needed for re-assembly. Natural redundancy in the English language, as well as character frequency analysis may provide indication about what words are present if the adversary knows that this is a block of published text. However, that contextual information about the message cannot be assumed.

4.2.3 Test Case 3: 3 Channels, 3 Hops, Policy 3

If all other factors for this test case stay the same except the policy is changed to 3, then the size of the fragments changes from 8-bytes to 1 byte. Duplication is still not employed for this policy. Therefore, what is received at the assembler is:

```
*n * h*le*in*th* g*ou*d *he*e *iv*d * h*bb*t.*No* a*na*ty* d*rt*,
*et*ho*e,*fi*le* w*th*th* e*ds*of*wo*ms*an* a* o*zy*sm*ll* n*r *et*a
*ry* b*re* s*nd* h*le*wi*h *ot*ing in it to sit down on o* t* e*t:*it*wa*
a*ho*bi*-h*le* a*d *ha* m*an* c*mf*rt* I* h*d * p*rf*ct*y *ou*d *oo*
l*ke*a *or*ho*e,*pa*nt*d *re*n,*wi*h * s*in* y*ll*w *ra*s *no* i* t*e
*xa*t *id*le* T*e *oo* o*en*d *n *o * t*be*sh*pe* h*ll*li*e * t*nn*l:*a
*er* c*mf*rt*bl* t*nn*l *it*ou* s*ok*, *it* p*ne*le* w*ll*, *nd*fl*or*
t*le* a*d *ar*et*d,*pr*vi*ed*wi*h *ol*sh*d *ha*rs* a*d *ot* a*d *ot*
o* p*gs*fo* h*ts and coats - the hobbit *as*fo*d *f *is*to*s.*Th* t*nn*l
*ou*d *n *nd*on* g*in* f*ir*y *ut*no* q*it* s*ra*gh* i*to*th* s*de*of*th*
h*ll*- *he*Hi*l,*as*al* t*e *eo*le*fo* m*ny*mi*es*ro*nd*ca*le* i* -*an*
m*ny*li*tl* r*un* d*or* o*en*d *ut*of*it* f*rs* o* o*e *id* a*d *he* o*
a*ot*er*
```

which is the original message with about 1 of every 3 characters changed[4]. The effec-

[4]The actual number of corrupted bytes is 31%, where asterisks represent corrupted character bytes.

tive QoSS probability of confidentiality is still calculated to be $P(C) = 0.7$ because no other factors have changed. However, by observation, this message is far more difficult to read because the corruption is more distributed throughout the original data set. As for the adversarial listener, they intercept

> Iao ertrleaoit s,iywl ldi e r dnoe,oy d,,ayo tngntoidnto,ntteoo.taaelrndri tl iegetahyeobkbnhecm.hdrpeoau-ada aue vyoaeuewhtewhaldaa osidnpe od tieci,nlnlsfe r dos eow novireuewnoa,ogalb uetitn i ei Tl lhppra l u dt da todospeontnnnh.

which is again about a third of the original message bytes, but with far less context for the eavesdropper because of the increased rudimentary scrambling of the character bytes. The effective QoSS probability of confidentiality is still calculated to be $P(C) = 0.7$, but the rudimentary scrambling has made it more difficult to piece together meaning from this message even if the message is assumed to be ASCII-encoded English text. This test case is also similar to a RAID 0 configuration, such that data is striped without any duplication. It is also similar to a RAID 5 configuration due to the distribution of the parity block across all the drives. The key differences is that the block size in this test case is 1 byte. RAID 3 configurations use block sizes of 1 byte, although RAID 3 also requires an additional drive for the parity blocks and is rarely used in practice [78].

4.2.4 Test Case 4: 3 Channels, 3 Hops, Policy 13

One final specific test case using the same test characteristics will be presented, but using Policy 13, such that the message is divided into 1-byte fragments and each fragment is sent twice ($DF = 2.0$). This specific test highlights the difference between Policy 3, 1-byte fragments with no duplication, and Policy 13, 1-byte fragments with each fragment sent twice. What is received at the assembler is

> In a hole in the ground there lived a hobbit. Not a nasty, dirty, wet hole, filled with the ends of worms and an oozy smell, nor yet a dry, bare, sandy hole with nothing in it to sit down on or to eat: it was a hobbit-hole, and that means comfort. It had a perfectly round door like a porthole, painted green, with a shiny yellow brass knob in the exact middle. The door opened on to a tube-shaped hall like a tunnel: a very comfortable tunnel without smoke, with panelled walls, and floors tiled and carpeted, provided with polished chairs, and lots and lots of pegs for hats and coats - the hobbit was fond of visitors. The tunnel wound on and on, going fairly but not quite straight into the side of the hill - The Hill, as all the people for many miles round called it - and many little round doors opened out of it, first on one side and then on another.

which is the original message without any corrupted bytes. By comparison to the previous examples, the minimal amount of duplication provides significant improvement to the overall integrity of the message in conjunction with the rudimentary error correction offered by the checksums. The effective QoSS probability of integrity is calculated at $P(I) = 0.91$, based on $\sigma = 3$ listeners, $\rho = 3$ disruptors, and a duplication factor $DF = 2.0$. This approach is directly comparable to a hybrid RAID 5+0 architecture. It features byte-level striping and mirroring. The mirroring is a full duplication of every byte, and in practice, the block size in RAID 5 configurations are much larger, i.e. 64 kilobytes. However, the use of distributed checksums to perform error checking is similar to the parity block in RAID 5 configuration. RAID 5 features block-level striping and distributed parity. It requires that all drives but one be present to operate. Upon failure of a single drive, subsequent reads can be calculated from the distributed parity such that no data is lost [78]. When employing the duplication across multiple channels and using rudimentary error correction, in practice, the probability of integrity is much higher than 91%. As for the adversarial listener, they intercept

> I ael h euonteervieahbo.t tan yt,it,yte l,f eldiht e n fw sm da nyz el , ry rdb a,snah ogi nt oi odo nrt o:t a sohbthloa nta tnasofroI ta afrel or ddrlki

> aptoelapit aniyelwoarsko bt heatcdid.Tehoorpnde ntatbuhsadhlailkatnu:l al ennewtohs me iwp ale dllsie d dcptderpoddw hpihdeahi,adntosn olo fesf h a n dtas h ebbiwsf dovsti.s etnu lwn nodn ,giof al ubontut earititnhteieo hteil- eh l,a lalh epel rmnaiml onuaclayl lteon drospndetu t ,srtnoendien hto nnteh

which is again about a third of the original message bytes, and again with less context than the previous example due to the smaller fragment size. The effective QoSS probability of confidentiality is still $P(C) = 0.7$, and again, the rudimentary scrambling has made it increasingly difficult to piece together meaning from this data. The presence of duplicated character bytes also adds to the rudimentary scrambling, reducing the context of character frequency and placement.

4.2.5 Generalized Results

The three specific test cases presented anecdotal indications that security may be improved using the QoSS model architecture. Using the results from these three examples as a starting point, Table 6 provides a broader view of the calculated probability of confidentiality versus the first 16 characters observed to be intercepted by a capable eavesdropper, varying the number of channels and the policy number. One significant challenge in building this simulation environment is creating a meaningful and repeatable metric that actually describes the number of bytes intercepted by a sophisticated eavesdropper compared to the data transmitted. This is the key aspect that makes calculating the confidentiality, and by extension security, so difficult. Unfortunately, that metric still does not exist. The closest to providing meaningful feedback is showing the first 16 characters that the eavesdropper intercepted. For these test cases, the test characteristic of probability of interception is $P(int) = 0.90$. The other characteristics that directly affect confidentiality, the probability of decryption $P(dcr)$ and the probability of decoding $P(dco)$, are set to 1.0 to maintain focus on the possibility an adversarial eavesdropper can fully intercept and observe transmitted messages[5]. The architecture utilizes two intermediary nodes for a total of three hops. In these tests, the probability of confidentiality is calculated using Equation 9, and the first 16 characters intercepted by the eavesdropper are shown.

Table 6. Calculation versus simulation of probability of confidentiality P(C) with probability of interception $P(int) = 0.9$, and probabilities of decryption and decoding $P(dcr) = 1.0$ and $P(dco) = 1.0$.

Calculated P(C)	Number of Channels, With 3 Hops, One $P(int) = 0.9$				
1st 16 Characters	1	2	3	4	5
policy = 0[a]	0.1	0.55	0.70	0.78	0.82
	In a hole in the	In a hol ground	In a holthere li	In a holved a ho	In a holed with
policy = 1[a]	0.1	0.55	0.70	0.78	0.82
	le in the ground	grothert a y, w	In a thethera ho	In a groved t a	In aund ed wworm
policy = 3[a]	0.1	0.55	0.70	0.78	0.82
	In a hole in the	I oei h rudteel	trleaoi l ldi en	I e utevab.tnyi	Ihieuhl b ati h
policy = 4[a]	0.1	0.55	0.70	0.78	0.82
	In a hole in the	F&&fbf'f&vvbvgbf	*invalid chars*	*invalid chars*	*invalid chars*
policy = 5[a]	0.1	0.55	0.70	78	0.82
	In a hole in the	'DFvDgDdEWGDFEDf	*invalid chars*	*invalid chars*	*invalid chars*
policy = 10[b]	0.1[c]	0.55[d]	0.70[e]	0.78[f]	0.82[g]
	In a hole in the	In a hole in the	In a hole in the	In a hole in the	In a hole in the
policy = 11[b]	0.1[c]	0.55[d]	0.70[e]	0.78[f]	0.82[g]
	In aIn a hol hol	In a hole in the	In ae in theund	In a theved . No	In a ground a ho
policy = 12[b]	0.1[c]	0.55[d]	0.70[e]	0.78[f]	0.82[g]
	InIn a a h holol	In a hole in the	In he ol t gunro	Inol tinveho. it	Ine ghebbt y,st
policy = 13[b]	0.1[c]	0.55[d]	0.70[e]	0.78[f]	0.82[g]
	IInn aa hhooll	In a hole in the	I ael nteervie	Iaohenht odntrl	I elnioruteriloh

[a] Duplication Factor $DF = 1.0$ and Average Channel Loading $AL = 1.0$.
[b] Duplication Factor $DF = 2.0$.
[c] Average Channel Loading $AL = 2.0$.
[d] Average Channel Loading $AL = 1.0$.
[e] Average Channel Loading $AL = 0.67$.
[f] Average Channel Loading $AL = 0.5$.
[g] Average Channel Loading $AL = 0.4$.

Looking at the results in Table 6, it is obvious that the intercepted portion of text is more readable in the single-channel cases and when Policy 0 and Policy 10 are utilized. In those case, there is no scrambling of the data, and the fragments are the largest possible size. It is also clear that the most obfuscated messages occur with Policy 4 and Policy 5 which employ bit-wise fragmentation, as was demonstrated in Figure 10 and Figure 11. The bit-wise fragmentation thus causes the intercepted message to capture only a portion of a character byte which causes the intercepted

[5]Recall that as $P(dcr)$ and $P(dco)$ are reduced, that reduces the amount of usable information the adversary is able to gather. Keeping those two values set to 1.0 focuses attention on the advantages and disadvantages of using multiple channels and message fragmentation across those channels.

message to be invalid characters. This again assumes the transmission of, and that the adversarial listener is expecting, ASCII-encoded text. Other types of data or other methods of encoding data are expected to have different results. The test case employing five channels and Policy 13 also demonstrates significant levels of obfuscation and a general lack of context on how to reassemble or rearrange the data into meaningful text.

Once again, using the results from the three example test cases as a starting point, Table 7 provides a broader view of the calculated probability of integrity versus the observed integrity, varying the number of channels and the policy number. For these test cases, the test characteristic of probability of injection is $P(inj) = 0.90$. The other characteristics that directly affect integrity, the probability of suppression $P(s)$ and the probability of noise $P(n)$, are set to 0 to maintain focus on the possibility a malicious attacker can spoof data in these systems. The architecture utilizes two intermediary nodes for a total of three hops. In these tests, the probability of integrity is calculated using Equation 16, and the observed integrity is an average of five experiment results as calculated by performing a bit-wise comparison between the transmitted and received message. As the resultant observed integrity value is an average of five samples, it remains a small sample size.

The results shown in Table 7 demonstrate that the security of the transmitted message is improved by using either more channels or a policy with more fragmentation and duplication. The observed integrity metric is calculated as a bit-wise comparison of the entire received message to the original message. For example, if the original message were exactly 100-bits long and the adversarial disruptor was able to inject three bad bits, then the message would be 3.0% corrupted, or conversely have 97% data integrity. This calculation is similar to the bit-error metric, bit is strictly for the transmitted message. The observed integrity values in this table are also an average

Table 7. Calculation versus simulation of probability of integrity P(I) with one compromised channel; probability of injection $P(inj) = 0.9$ for one test characteristic.

Calculated P(I)	Number of Channels, With 3 Hops, One Bad Channel, $P(inj) = 0.9$				
Observed I	1	2	3	4	5
policy = 0[a]	0.1	0.55	0.70	0.78	0.82
	0.12	0.58	0.71	0.79	0.79
policy = 1[a]	0.1	0.55	0.70	0.78	0.82
	0.11	0.54	0.70	0.76	0.78
policy = 3[a]	0.1	0.55	0.70	0.78	0.82
	0.14	0.58	0.71	0.78	0.80
policy = 4[a]	0.1	0.55	0.70	0.78	0.82
	0.08	0.20	0.44	0.41	0.67
policy = 5[a]	0.1	0.55	0.70	0.78	0.82
	0.12	0.08	0.29	0.26	0.47
policy = 10[b]	0.1[c]	0.80[d]	0.80[e]	0.89[f]	0.93[g]
	0.13	0.10	1.00	1.00	1.00
policy = 11[b]	0.1[c]	0.80[d]	0.80[e]	0.89[f]	0.93[g]
	0.1	0.55	1.00	1.00	1.00
policy = 12[b]	0.1[c]	0.80[d]	0.80[e]	0.89[f]	0.93[g]
	0.11	0.33	1.00	1.00	1.00
policy = 13[b]	0.1[c]	0.80[d]	0.80[e]	0.89[f]	0.93[g]
	0.09	0.10	1.00	1.00	1.00

[a] Duplication Factor $DF = 1.0$ and Average Channel Loading $AL = 1.0$.
[b] Duplication Factor $DF = 2.0$.
[c] Average Channel Loading $AL = 2.0$.
[d] Average Channel Loading $AL = 1.0$.
[e] Average Channel Loading $AL = 0.67$.
[f] Average Channel Loading $AL = 0.5$.
[g] Average Channel Loading $AL = 0.4$.

of five separate simulations. In some cases, the variation is caused by the random number generation withing the dissembler code. However, the observed integrity of Policy 4 and Policy 5 are far lower than expected. Conversely, the observed integrity of policies 10, 11, 12, and 13 are all 1. This is because these policies send duplicated fragments and the rudimentary error correction algorithm is able to filter out the injected bad data and replace it with correct data.

As a way to dig deeper into the performance of the architecture and simulation, two channels are now be set to be compromised. This test case does not provide any additional information when only one channel is utilized since we know that the one available channel has been compromised. For those cases, we assume that two nodes have been compromised, which simply suggests a very bad connection or poor routing. Even the two channel system yields little new information. For the two-channel system, it means that both channels have been compromised. In the remaining situations, this set of experiments can provide insight into the survivability of the data when under attack. The simulation data shown in Table 8 is an average of five runs using the same input, although in this set of experiments, two non-adjacent probabilities of injection are $P(ing) = 0.90$. All other test characteristics and configurations remain the same from the previous data set.

Table 8. Calculation versus simulation of probability of integrity P(I) with two compromised channels; probability of injection $P(inj) = 0.9$ for two test characteristic.

$\underline{\text{Calculated P(I)}}$	Number of Channels, With 3 Hops, Two Bad Channels, $P(inj) = 0.9$				
Observed I	1	2	3	4	5
policy = 0[a]	$\underline{0.01}$	$\underline{0.10}$	$\underline{0.40}$	$\underline{0.55}$	$\underline{0.64}$
	0.01	0.07	0.39	0.55	0.63
policy = 1[a]	$\underline{0.01}$	$\underline{0.10}$	$\underline{0.40}$	$\underline{0.55}$	$\underline{0.64}$
	0.00	0.13	0.43	0.42	0.66
policy = 3[a]	$\underline{0.01}$	$\underline{0.10}$	$\underline{0.40}$	$\underline{0.55}$	$\underline{0.64}$
	0.01	0.13	0.45	0.53	0.63
policy = 4[a]	$\underline{0.01}$	$\underline{0.10}$	$\underline{0.40}$	$\underline{0.55}$	$\underline{0.64}$
	0.01	0.01	0.14	0.19	0.47
policy = 5[a]	$\underline{0.01}$	$\underline{0.10}$	$\underline{0.40}$	$\underline{0.55}$	$\underline{0.64}$
	0.01	0.01	0.01	0.16	0.28
policy = 10[b]	$\underline{0.01}$[c]	$\underline{0.10}$[d]	$\underline{0.60}$[e]	$\underline{0.77}$[f]	$\underline{0.85}$[g]
	0.00	0.14	0.17	1.00	1.00
policy = 11[b]	$\underline{0.01}$[c]	$\underline{0.10}$[d]	$\underline{0.60}$[e]	$\underline{0.77}$[f]	$\underline{0.85}$[g]
	0.01	0.09	0.20	0.57	0.65
policy = 12[b]	$\underline{0.01}$[c]	$\underline{0.10}$[d]	$\underline{0.60}$[e]	$\underline{0.77}$[f]	$\underline{0.85}$[g]
	0.02	0.09	0.12	0.60	0.75
policy = 13[b]	$\underline{0.01}$[c]	$\underline{0.10}$[d]	$\underline{0.60}$[e]	$\underline{0.77}$[f]	$\underline{0.85}$[g]
	0.01	0.09	0.24	0.62	0.74

[a] Duplication Factor $DF = 1.0$ and Average Channel Loading $AL = 1.0$.
[b] Duplication Factor $DF = 2.0$.
[c] Average Channel Loading $AL = 2.0$.
[d] Average Channel Loading $AL = 1.0$.
[e] Average Channel Loading $AL = 0.67$.
[f] Average Channel Loading $AL = 0.5$.
[g] Average Channel Loading $AL = 0.4$.

The results shown in Table 8 again demonstrate that the security of the transmitted message is improved by using either more channels or a policy with more fragmentation and duplication. As was the case previously, the observed integrity values in this table are also an average of five separate simulations. Just by observation, the calculated probability of integrity is very poor for all systems with one, two, or even three channels. What is notable is that there are clearly a few situations where the metrics do not accurately account for having two compromised channels. Those cases are Policy 4 and Policy 5 with the systems employing 4 or 5 channels. The discrepancy for those cases needs to be analyzed and the calculations need to be refined. It should also be pointed out that Policy 10 performed better than expected with 4 or 5 channels, and that Policy 12 and Policy 13 actually demonstrated poorer than expected results. The discrepancy is too big for it to simply be a matter of random variation in the testing, thus there must be lingering effects of multiple exploited channels that have not been anticipated.

One final set of generalized test cases was explored, and Table 9 provides a different perspective on the calculated probability of integrity versus the observed integrity. For this set of experiments, the test characteristic of probability of suppression is $P(s) = 0.90$. The other characteristics that directly affect integrity, the probability of injection $P(inj)$ and the probability of noise $P(n)$, are set to 0 to maintain focus on the possibility a malicious attacker can DoS on these systems, and if the error correction algorithm can correct for the missing data. The architecture again utilizes two intermediary nodes for a total of three hops, and the probability of integrity is calculated using the Equation 16. The observed integrity is an average of five sets of experimental results as calculated by counting the missing bits between the transmitted and received message. As the resultant observed integrity value is an average of five samples, it remains a small sample size.

The results shown in Table 9 are dramatically different than the calculated metrics. Some variation was expected, and a few significant discrepancies would be tolerable. However, the entire data set, excluding the single-channel system or Policy 0, is unexpectedly wrong. In troubleshooting the simulation, it became clear that the

Table 9. Calculation versus simulation of probability of integrity P(I) with one suppressed channel; probability of suppression $P(s) = 0.9$ for one test characteristic.

$\frac{\text{Calculated P(I)}}{\text{Observed I}}$	Number of Channels, With 3 Hops, One Bad Channel, $P(s) = 0.9$				
	1	2	3	4	5
policy = 0[a]	$\underline{0.1}$	$\underline{0.55}$	$\underline{0.70}$	$\underline{0.78}$	$\underline{0.82}$
	0.11	0.57	0.70	0.76	0.79
policy = 1[a]	$\underline{0.1}$	$\underline{0.55}$	$\underline{0.70}$	$\underline{0.78}$	$\underline{0.82}$
	0.13	0.34	0.69	0.35	0.36
policy = 3[a]	$\underline{0.1}$	$\underline{0.55}$	$\underline{0.70}$	$\underline{0.78}$	$\underline{0.82}$
	0.11	0.22	0.20	0.24	0.13
policy = 4[a]	$\underline{0.1}$	$\underline{0.55}$	$\underline{0.70}$	$\underline{0.78}$	$\underline{0.82}$
	0.07	0.18	0.23	0.15	0.09
policy = 5[a]	$\underline{0.1}$	$\underline{0.55}$	$\underline{0.70}$	$\underline{0.78}$	$\underline{0.82}$
	0.12	0.05	0.04	0.14	0.05
policy = 10[b]	$\underline{0.55}$[c]	$\underline{0.78}$[d]	$\underline{0.85}$[e]	$\underline{0.89}$[f]	$\underline{0.91}$[g]
	0.08	0.45	0.30	0.45	43
policy = 11[b]	$\underline{0.55}$[c]	$\underline{0.78}$[d]	$\underline{0.85}$[e]	$\underline{0.89}$[f]	$\underline{0.91}$[g]
	0.02	0.32	0.35	0.42	0.45
policy = 12[b]	$\underline{0.55}$[c]	$\underline{0.78}$[d]	$\underline{0.85}$[e]	$\underline{0.89}$[f]	$\underline{0.91}$[g]
	0.01	0.31	0.29	0.34	0.36
policy = 13[b]	$\underline{0.55}$[c]	$\underline{0.78}$[d]	$\underline{0.85}$[e]	$\underline{0.89}$[f]	$\underline{0.91}$[g]
	0.01	0.36	0.21	0.19	0.20

[a] Duplication Factor $DF = 1.0$ and Average Channel Loading $AL = 1.0$.
[b] Duplication Factor $DF = 2.0$.
[c] Average Channel Loading $AL = 2.0$.
[d] Average Channel Loading $AL = 1.0$.
[e] Average Channel Loading $AL = 0.67$.
[f] Average Channel Loading $AL = 0.5$.
[g] Average Channel Loading $AL = 0.4$.

rudimentary error-correcting algorithm in conjunction with the received-versus-sent comparison code failed to correctly handle the missing bytes of data. They also failed to correctly handle the missing checksums. Output messages were jumbled, and in some cases, text was duplicated. What is even more disappointing is that the values in the table are an average of five test runs; individual test results varied widely. Despite these failures, the simulation environment did correctly identify the suppressed channel in all but the two-channel test cases, and recorded feedback stating that, for example, "`Channel A is corrupted; do not trust data from Channel A!`". In the two-channel exercises, because the checksums were not verified correctly, the rate of incorrect versus correct failed channel was approximately 50%. This is an unexpected setback to the simulation environment, although it is an opportunity to evaluate how to improve the overall error detection and error correction algorithms.

4.3 Systemic Effects on Confidentiality

The effects of data fragmentation on confidentiality may be viewed from two perspectives, that of the authorized user and that of the eavesdropper. The authorized user, in most cases, wants as little information to be intercepted by the eavesdropper as possible. The ideal case is exactly zero bits being intercepted, although a limited amount of intercepted data may be acceptable as long as it is not useful. Conversely, the eavesdropper wants to collect as much useful or actionable information as possible. The ideal case is to collect, decrypt, and decode all information. However, this suggests a specific minimum amount of data that must be intercepted to be useful[6]. Because of these opposing viewpoints, the implications must be assessed from both the authorized user's perspective and from the adversarial perspective.

4.3.1 From the User's Perspective

From the authorized user's perspective, to achieve secure communications and reduce possible data leakage, the simulation environment points to two possible solution sets. One solution is to use strong encryption for all transmissions and regularly change the password. It is irrelevant if an eavesdropper is able to intercept messages if they are not able to decrypt them. However, many systems are unable to perform necessary encryption processes. In these cases, the second solution set may be useful.

The second solution utilizes multiple parallel channels and distributes message fragments across those channels with minimal duplication. As demonstrated in Table 6, there is significant obfuscation of the data when multiple channels and fragmentation are used. That obfuscation, even though the adversarial listener was able

[6]The minimum amount of data that must be intercepted to be useful is expected to be different based on the type of date, e.g., text, audio, video, or binary sensor data.

intercept, decrypt, and decode those messages, the reassembly still provided an improvement to the system's confidentiality. The technical hurdles contained in this solution include: the transmitter and disassembler must set up and maintain numerous separate channels; the receiver must accurately reconstruct the data; the system must allow for, or survive a threshold amount of latency; and the system must appear as a single point-to-point connection. All of these were enumerated in Section 3.3 in some manner. Modern communication networks have mastered the ability to minimize and tolerate latency, whereas this architecture would harness that requirement across multiple connections, while also managing possible timing disparities, jitter, or potentially lost packets or channels. In [51], the author further addresses some of these challenges.

The probability of leakage presents a trade space between security and network complexity. As demonstrated by the simulation environment and reflected in Table 6, with data split evenly ($DF = 1.0$ and $AL = 0.5$ across two channels), an adversary with access to only one channel has access to 50% of the data. Across three channels with the same amount of fragmentation, the adversary's challenges become harder. Furthermore, due to the size of the fragments and the manner those fragments are assembled into the transmission packets, the probability of decoding the data also decreases. Since the original message is split across multiple channels, the overall time the connection must be maintained is correspondingly reduced. The effective throughput of the system is increase with each channel. Therefore the temporal opportunity for exploitation by an adversary is also reduced, especially if they are only aware of one channel. These benefits continue to increase as the number of channels increases.

Clearly, the more channels that are instantiated, the more complex the system becomes. If each channel is routed differently, there is no guarantee that data packets from each channel will be received at the same time. This obviates the need for packet enumeration so the protocol recombines the fragments correctly. A buffering process that dynamically compensates for delayed packets would certainly be useful. These are technical trade-offs that exist within most technologies. However, in situations where using encryption is not an option, at least there is some opportunity to thwart eavesdroppers through the use of rudimentary scrambling and multiple channels.

A third solution set does exist and may be applied to situations where driving data leakage to zero is imperative. In those cases, to ensure near-absolute security, the original message may be encrypted and then fragments of the encrypted message may be distributed across multiple channels, similar to the techniques that are proposed in other research [47]. Each fragment may also be encrypted before transmission for added confidentiality. This would have the ultimate effect of layering protection mechanisms, possibly to a degree beyond the capabilities of the eavesdropper and driving the probability of leakage to near zero. This solution adds technical challenges to achieve exceptionally high levels of security. This should be considered a hybrid solution that incorporates all technologies and theoretical solutions.

4.3.2 From the Adversary's Perspective

Estimating capabilities and intentions of an adversarial listener is always challenging. In [79], the authors develop a model to analyze the possibility, goals, and capabilities of an eavesdropper. Without any additional information about the adversary, a logical assumption is that an adversary has equal or greater knowledge or capabilities than an authorized user. In this research, a foundational assumption is that the adversary is aware of other channels, and of the potential to fragment data across multiple channels. A second assumption is that they are aware of the protocols used in the transmitter and receiver. However, the number of utilized channels, the channel characteristics, and the size and distribution of the data fragments remain unknown to the adversary. That level of obfuscation works to the adversary's disadvantage.

The challenge for the eavesdropper is to align the probabilities of interception, decryption, and decoding. If strong encryption is used, then the data the eavesdropper intercepts, regardless of number of channels or fragments, will appear to be randomized data. However, in the cases where encryption is not used, the adversary has a reasonable opportunity to intercept data, and systems that do not utilize encryption will be targeted.

Ignoring the challenges of receiving specific coherent transmissions on different physical media, exploiting a communication system begins with intercepting a message transmission. If the architecture utilizes a single wired or wireless channel between transmitter and receiver, the adversary must tap into the physical media without revealing themselves. If, for example, data is transmitted across a single channel that has multiple intermediate relay nodes, any of those connections will provide the same data, thus the whole channel is compromised. If the architecture utilizes two channels and splits the data equally between the two channels, the eavesdropper with access to one of the two channels has access to at most 50% of the data. If the two channels have multiple intermediary nodes, those provide additional points for infiltration, but no additional information. If the eavesdropper gains access to both channels simultaneously, then the eavesdropper has access to 100% of the data. As more channels are introduced, it becomes more challenging to intercept all those channels simultaneously, especially if they feature path diversity. The eavesdropper may be able to discern the existence of additional channels based on the message and packet formation, but that does not guarantee successfully finding and intercepting them.

The probability of decoding is the other key factor in the adversary's calculations, specifically reassembling and decoding the intercepted message. Much like an eavesdropper listening into a conversation, they need to figure out the content and context of the communications. They cannot assume that data is ASCII-based text, or even English language based. The data may appear scrambled, but additional information may be available about the messages elsewhere. Based on partial reassembly of the message or based on educated guess, the adversary may determine the existence of other channels, or they may determine what portions of the message are missing. Depending on the value of the target, the eavesdropper may take additional measures to discover techniques used in transmitting the messages. It must be assumed that the eavesdropper is relentless and, with a sufficient portion of data intercepted, will eventually discover any secret.

4.4 Systemic Effects on Integrity

The effects of multiple channels and data fragmentation on integrity revolves around the ability to compensate for any form of adversarial influence on the data. As previously stated, the probability of integrity is the compliment of the probability of corruption. There are three broad vectors for adversarial data corruption: data injection, suppression, and noise. The integrity of the data is directly affected by malicious injection of data. The same is true for suppression and noise. It should be noted that the calculated metrics for the probability of integrity based on suppression shown in Table 9 are very similar to the calculated probability of integrity metrics shown in Table 7 using probability of injection. This reinforces the concept that active, malicious suppression should be considered part of integrity, not availability. The probability of noise was not covered at length within the simulation environment, but noise also causes data corruption. Not including noise in the simulation was an analytical choice to simply reduce the number of variables and focus on the larger challenges. Furthermore, combining all of these variables together would make for an extremely difficult analysis. Separating them and observing how they behave individually helps clarify their effects.

Most modern networks have some methods to compensate for noise and to route messages around DoS attacks. However, most systems are unable to adequately address adversarial injection of bad data. As demonstrated in Table 7 and Table 8, employing multiple channels, data fragmentation, and duplication even in a rudimentary manner demonstrates significant benefits to the overall integrity of the data. Having multiple channels and a buffering mechanism to collect the data packets at the assembler add significant complexity. The metrics and the simulation have provided evidence that a properly designed system can survive numerous attack vectors. Having multiple channels, duplicated data, checksums, error detection, feedback mechanisms to request packet retransmission all together provides network operators various methods to alleviate the possible affects of noise, malicious data injection, or jamming attacks.

4.5 Combined Effects on Confidentiality and Integrity

Generally speaking, improving the security of a communication system focuses on either improving confidentiality or integrity. Attempts to improve one would either be detrimental to the other or would have no impact at all. However, they need not be mutually exclusive or orthogonal to each other. The effect of improving one to the detriment of the other may be observed in Table 6 and Table 7. Looking at the observed results, the confidentiality improvements of sending bit-wise fragmented data using Policy 4 or 5 were great. However, the integrity suffered because the cross-channel fragmentation of the bytes appears to amplify the effect of injection or suppression. However, maintaining a byte-wise fragmentation and including duplication showed significant improvement to both confidentiality and integrity. Furthermore, as the number of channels increased in conjunction with duplicating fragments, the observed confidentiality and integrity metrics both increased. Having a more intelligently designed error correction algorithm would provide even more dramatic improvements to the observed integrity metrics.

4.6 Dynamic Performance

One function listed in the protocol requirements that was not implemented in the simulation environment is the use of feedback. Dynamic performance is the active switching of policies and number of channels based on feedback from the assembler. The assembler would need to incorporate additional behaviors and complexity, and be thoroughly tested to anticipate and handle all possible performance behaviors. Many of these behaviors would require a form of threshold such that if the number of bad or delayed packets exceeded the threshold, then it would switch to another policy. The testing process would need to thoroughly exercise a wide array of attack vectors, and would need to be relentless in the attempted exploitation of the architecture. Additionally, the steps from one policy to another would need to be clearly defined, as would be the formatting and contents of the feedback messages. The current state of the feedback mechanism is after the checksums in the packets with matching packet number are compared, either a simple acknowledgement is sent, or a message stating that a certain channel appears bad and the respective packets appear bad, too. These feedback messages are modeled after TCP/IP message acknowledgements. However, to switch policies or number of channels, the feedback mechanism would need to be much more robust and detailed, beyond that of a simple acknowledgement. It would also require the dissembler to be fully synchronized with the assembler, recognizing the difference between changing policies and changing channel count. In this architecture, the assembler makes the request to change, and the dissembler confirms the transition by sending the data in that format. Much in the same manner that software defined networking allows routers and switches to reroute traffic around congested nodes or other inefficiencies, so too the dissembler and assembler would provide much more robust data and traffic security. Therefore, the level of sophistication needed to simulate dynamic performance was not available in the current design state of the protocol and simulation environment, but the possible continual improvement of communication security would be noticeable. If the adversary were always one step behind the dissembler-assembler pair because they were constantly changing settings in a synchronized manner, then the adversary would not be able to gather or tamper with any data.

4.7 Real-World Use-Cases

If this architecture were fully designed and developed, complete with a functional dissembler-assembler pair, it would be most useful in existing systems where security is critical but encryption algorithms add too much latency or require too much processing power. These systems would need to tolerate a protocol that can be inserted between the application layer and the TCP/IP stack. It would also be beneficial for the system to transmit small data sizes, for example binary strings, ASCII text, or similar types of data. Specific real-world use-cases could include Programmable Logic Controller (PLC) networks or Baseband Management Controller (BMC) interfaces for server-class computers. PLC networks would be a good use-case because they utilize and communicate via binary data strings. A standard PLC also operates at near real-time, and the processors tend to not be powerful enough to employ encryption algorithms. As the typical PLC is designed to run for many years without fail, they tend to be comparatively expensive, and not upgraded regularly. Thus, older hardware is prevalent. BMC interfaces allow for remote diagnostics and control of server-class systems, and therefore require as much security as possible when interacting with them. Furthermore, the BMC itself is a stripped-down processor that is designed to do only a few specific tasks in the booting and monitoring of the computers. There are certainly additional real-world use-cases, including military communication systems, vehicle-to-vehicle communication systems, satellite command and control, other critical infrastructure and Supervisory Control and Data Acquisition (SCADA) systems, or in-home monitors. In all these situations, confidentiality and integrity of data is critical, the possibility of malicious interaction is possible, and there are known limits to the existing hardware or software.

4.8 The Big Picture

Despite the protocol requirements listed in Chapter III, Section 3.3, there is one aspect that needs to be defined. A generalized requirement for data fragmentation across multiple channels must be to determine the minimum data size. The minimum data size is defined to be the smallest sized data block that may not be divided without causing further detriment to data integrity. The minimum data size may be 1 bit, 1 byte, 1 word, or even larger, depending on the requirements and type of data. Once that is determined, the minimum fragment size must be no smaller than the minimum data size. In the simulation communication architecture, the minimum data size is 1-byte, which represents one ASCII character. For example, the test cases where the fragments are 4-bits or 1-bit demonstrate significant applicable confidentiality because the rudimentary scrambling eliminates a great deal of the data context. However, because the fragment sizes for those case are smaller than the minimum data size, it causes the byte to be split across channels. Therefore, if one of the channels is tampered with, then the entire byte is corrupted, directly affecting two channels worth of data. Fragmenting data to be smaller than the minimum data size and distributing those fragments across multiple channels actually amplifies the effect of malicious injection, suppression, or noise. Therefore, in an effort to maximize both confidentiality and integrity, the minimum fragment size must be no smaller than the minimum data size. Further research is needed to define the minimum data size for other types of data, e.g., images, audio, or sensor data, which is beyond the scope of this development.

The most difficult aspect of developing the QoSS metrics is making assumptions about the network characteristics. For these examples, we began with an assumption that $P(dcr) = 1.0$ and $P(dco) = 1.0$ as a baseline value that an adversary would be able to access all critical data. What does it really mean to have $P(dcr) = 1.0$ and $P(dco) = 1.0$? Perhaps the assumption implies that no encryption is used, despite the fact that the use of encryption is strongly encouraged for all communications systems. It also implies that the adversary knows how the data is encoded, packaged, and arranged. These are very broad assumptions. Similarly, is it possible for $P(dcr) = 0$? That assumption implies that the encryption is unbreakable at this time and under these communication and environmental conditions. The fact that we do not know the adversary's fullest capabilities, nor do we know the adversary's intentions, are considerations that must be included, within a range, in the estimate for the probabilistic aspect of our metrics. More accurately, we estimate what is possible within the current state-of-the-art and under a set of operational characteristics.

Adversarial intention is much more difficult to estimate because intentions may change rapidly and on a case-by-case basis. In light of that, we have attempted to reflect all the adversarial intentions, whether it is jamming, spoofing, or eavesdropping, within the generalized probability of the QoSS metrics and associated architecture.

As these metrics are all reflected in probabilistic terms, estimations are unlikely to be perfectly correct. The QoSS metrics are also intended to be a single snap-shot in time, in which case, the model estimations may be updated based on new research, information, or changing environmental and systemic conditions, which directly calls into practice of using feedback mechanisms within the architecture to constantly update awareness. An inaccurate estimate that does not readily map to desired security requirements should be viewed as an opportunity to improve the data or the model.

4.9 Summary

The design of the architectural model, QoSS metrics, and security-focused data-handling protocol culminated with the simulated performance of transmitting ASCII-encoded text across the system. The simulation provided some insights about what is good in the architecture, what is bad in the protocol, and what aspects need more work. In most cases, the observed simulation data matched the calculated behavior. However, there are clearly issues with error correction and other aspects of data science. These issues appear to be directly related to how the error correction algorithms are affected by multiple compromised channels. The simulation provided clear indications that security, specifically confidentiality and integrity, may be improved. The simulation also shows some compelling insights that the metrics closely follow most performance expectations. The technology exists to achieve those improvements.

V. Conclusion

5.1 Overview

Engineering is a field full of trade-offs, typically between opposing technical goals. Security engineering manifests many such trade-offs, including cost, flexibility, reliability, complexity, and performance. The more secure systems become, the less flexible they are. In the research presented herein, we have explored the possibility that security need not be a trade-off between confidentiality and integrity, that both can be enhanced through careful development. This chapter provides the summary of the trade-offs and potential benefits to a security-focused communication protocol and associated QoSS metrics. This chapter will provide conclusions for the main research elements, results, as well as topic areas of focus for future research.

5.2 Research Summary

Improving security is the primary goal of this research. Unfortunately, security is a vague, qualitative term, and the goal of improving security is exceptionally broad. Therefore, to scope the effort more narrowly, security is framed as preventing the malicious infiltration of communication systems by adversarial listeners or disruptors. This is still quite broad, however it implies a series of questions about how to prevent infiltration. The research questions developed in Chapter I and explored in Chapter III and Chapter IV are:

- To what degree can we quantitatively determine the available or existing security of a communication network in an reliable and repeatable manner?
- What are the shortfalls of metrics?

- What are the essential elements of a session layer protocol that enforces or ensures a desired or required level of security?
- What are the benefits of dynamically adjusting network configurations based on feedback mechanisms?

These questions served as a guide through this research, development, and analysis of a security-focused session layer communication protocol. However, a follow-up question is whether the research, development, and analysis have adequately answered or addressed these concerns.

As presented in Chapter II, technical inspiration was derived from observations and analysis of various architectures, protocols, and approaches. Because IT security is an element of the discussion regarding communication security, a review of CIA triad was presented. Additionally, the OSI model, specifically the session layer, and several existing security-focused protocols and technologies were also evaluated. Individually, many of the presented technologies and protocols created isolated pockets of secure communications, but viewed from a broader lens, assembling them in a different manner may be the key to the development of a security-focused architecture, data handling protocol, and related metrics.

A broad methodology was presented in Chapter III. Within this chapter, the concept of a communication architectural model featuring multiple heterogeneous channels was introduced. The use of intermediate relay nodes was also discussed. This combination of multiple channels and intermediate nodes into the architectural model provided a foundation for the development of the communication protocol. Transmitting data across multiple channels next implied a method of exploiting those multiple channels. Various methods of fragmenting and duplicating data were reviewed as viable methods to utilize the multiple available channels. This in turn was demonstrated to obfuscate the messages in a non-cryptographic manner as they were sent across the available channels. Utilizing the multiple channels was also demonstrated to improve the possibility of reconstruction if one of the channels were compromised.

The need for additional design requirements for the data handling and communication protocol became obvious with the introduction of the architectural model and initial concepts of fragmentation and duplication. Those requirements were listed in Section 3.3. They serve as a foundation for the development and performance of the security-focused session layer communication protocol and help guide the design process. Defining this initial set of protocol requirements speaks directly to the third research question of defining the essential elements of a session layer protocol. This initial set details the major functions and features needed by the protocol to ensure desired or required levels of security, although additional elements and requirements will certainly become apparent with continued development.

A set of metrics were then derived to describe existing and available security. These metrics are based on the surrogate measurements of probability of confidentiality and the probability of integrity, as direct measurement of confidentiality and integrity are difficult if not impossible. The compliment of the probability of confidentiality is the probability of leakage which is decomposed into the probability of interception, probability of decryption, and probability of decoding. Similarly, the compliment of the probability of integrity is the probability of corruption which is decomposed into the probability of injection, probability of suppression, and the probability of noise. These various probabilities were all defined and a direct relationship between them and the number of adversarial listeners, malicious disruptors, number of channels, and the fragmentation and duplication of the messages was detailed. With the initial development of the metrics, we begin to demonstrate the degree that we can quantitatively determine the available or existing security. The available or existing security is only a snap-shot in time and is presented as a probability, although that is as accurate as possible without intimate knowledge of the adversary's abilities and intent.

With the derivation of the QoSS metrics and the development of the architectural model, the next step in the development and analysis is the development of the simulation environment. The simulation environment was designed to allow scrutiny of the architecture while injecting metrics and observing the outcomes. The results generated by the simulation environment point to several limitations of the metrics, specifically when individual channels are suppressed. Other limitations demonstrated a lack of clear quantification of the amount of data and information captured by an eavesdropper. However, having the ability to observe the simulation environment in action throughout the development and testing processes is directly in line with another of the research questions. Additionally, having simulated data travel across the network architecture achieves real data throughput, which highlights that at least some of the essential elements of the protocol and larger system were properly recognized and implemented. The element of feedback is one of several of the elements that were specified but not fully realized or implemented.

The development of the simulation environment allowed the collection and analysis of data. As was demonstrated in Chapter IV, there is still some significant limitations to the metrics, particularly with respect to accuracy in all situations of multiple channels, fragmentation, and duplication. There are some calculated measurements that simply did not match what was observed in the simulation environment. However, some metrics lined up very well. Furthermore, the ability of the metrics to accurately record the performance of the architecture pointed to the possibility of using the feedback mechanism to dynamically adjust system performance. Despite the fact that fully dynamic operation was not explored, the analysis of the performance data showed that switching system parameters would allow for improved operations in the presence of adversarial interference. Although the simulation environment provided some unique insights into the system performance across various parameters, the degree of accuracy of the metrics remains questionable without a real proof-of-concept system. However, it is encouraging because the metrics appear to be close.

The risks that plague this research are similar to the risks to other methods of improving security. The primary risk is that the series of metrics meant to quantify and improve security are either misleading or inaccurate. As a result, the deployed architecture and protocol might instead decrease security, leaking information to adversaries and allowing systems to be easily disrupted. It also must be noted that the available technology may not be mature enough to support the architecture and protocol, thus not meeting the promised improvements. The development of the requisite software and protocol shims may not account for all use-cases, and may not ever achieve acceptance. However, as this research is intended to improve security, the technology can be deployed in small test networks to verify and validate performance without impacting a wider audience. Additional risks include that the feedback mechanism may open up new, unexpected attack vectors, or the test network may not accurately reflect real-world scenarios. Therefore, care must be taken at all steps of the research, analysis, testing, and deployment to rigorously monitor performance and question all assumptions.

Taken as a whole, the architectural model, security-focused protocol, QoSS metrics, and simulation environment provide clear answers to the research questions. The degree to which we can quantitatively determine the available or existing security of a communication network is not with perfect fidelity or granularity. However, the reliability and repeatability are based on the existence of the QoSS metrics which codify how the characteristics fit together. The metrics, both calculated and observed in the simulation environment, capture many of the behaviors of the architecture. While not perfect, it indicates that the equations are mostly accurate, although they require some refinement to fully cover all the corner cases. The essential elements of the session layer protocol demonstrated many abilities to manage transmitting data across the architecture while enforcing a particular security level. It remains to be seen if, in the hands of a cyber security expert well versed in hacking techniques, the security mechanisms survive. This points directly to the potential benefits of dynamically adjusting network configurations based on feedback mechanisms as a way to obfuscate data and evade further adversarial and malicious interference.

5.3 Future Research and Development

Based on the research presented herein, there are numerous avenues for future lines of research and development. Many of them focus on further refinement of the architecture, the metrics, or the real-world functionality of a security-focused communications system.

One area of future research may be in the organization of the architectural model and its relationship to the metrics and simulation environment. For example, the number and behavior of nodes instantiated throughout the simulation test architectures was not fully explored. The nodes were developed to merely act as zero-delay relay stations. However, in real networks, there is some appreciable delay and transmission latency. As the simulation environment is tuned to be even more realistic, a variable amount of non-zero delay can be added to the performance of the nodes. In that manner, having more nodes in a channel can demonstrate more delay, and multiple channels can demonstrate different amounts of latency. The framework to make these changes exist within the code for the nodes, but it will likely require the addition of some buffering functions in the assembler, to collect in-coming packets and assemble matching packet numbers prior to verifying cross-channel checksums.

If the buffering timer exceeds a threshold value, then that channel can be considered faulty, and either feedback is sent requesting the lost packet, a new channel, or some other means of working around the failure. Nonetheless, there is ample topics to explore with respect to the behavior of the nodes and their downstream effects on the system metrics.

Another future consideration of data fragmentation and duplication is using RAID as a direct template, or bringing the policies into closer alignment with RAID configurations. There are several similarities between the security-focused communication protocol, and aligning the configurations would make further analysis and description easier. Those similarities include the fragmentation, or striping, of data, duplication, or mirroring, data, and the use of checksums, or parity, to validate data. In future evaluations, the security-focused architecture and protocol should be able to demonstrate similar performance enhancements that are available in RAID drives but with the added ability to change the fragment size[1]. A simple series of tests could apply similar policies and configurations and then analyze the data stored on one disk of an array for readability or information leakage. This method of testing the protocol could borrow from and build on the lessons learned in RAID development. These tests would also serve to highlight the difference in design goals between RAID and the security-focused architecture and protocol: where RAID was designed to enhance the reliability of disk drives and maintain large-scale data integrity, the design goal for the security-focused architecture and protocol is to ensure and maintain the integrity and confidentiality of data in transit.

On an similar subject, a generalized requirement for data fragmentation across multiple channels must be determined for the minimum data size. The minimum data size, as defined in Section 4.8, is suggested to be the smallest sized data block

[1]The standard block size in RAID configurations is 64 kilobytes (kB), however 16, 32, and 128 kB options are available, with some support for smaller block sizes or sizes as large as 256 kB.

that may not be divided without causing further detriment to the combined data integrity and confidentiality. As was noted previously, for RAID systems, the standard block size is 64 kB, but can be as small as 1 bit (rarely used) and as large as 256 kB (uncommon). However, RAID was designed to enhance data integrity; confidentiality was not a design constraint for RAID systems. Further research is needed to define the minimum data size be for data types other than ASCII-encoded English text. This line of research would require the knowledge and understanding of how various types of files and data are organized at the most granular level.

Generally, the fact remains that we do not know an adversary's fullest capabilities, nor do we know an adversary's intentions. Until a fully realized system is developed, tested, and compared to the theoretical and simulation results, adversarial capabilities and intentions are merely estimates represented in the probabilistic aspect of the QoSS metrics. Without actual hardware and software, the estimates are guesses of what is possible within the current state-of-the-art and under a set of operational characteristics. Furthermore, adversarial intention is much more difficult to estimate; intentions may change rapidly or may vary on a case-by-case basis. In light of that, the QoSS metrics attempt to reflect all the adversarial intentions, whether it is jamming, spoofing, or eavesdropping, within the generalized probability of confidentiality and probability of integrity. With these estimations, both adversarial capability and intention are difficult to concretely quantify in the initial pass, and they are, thus, cast in probabilistic terms. Therefore the most logical next step for this research is to develop and test a fully realized protocol and deploy it on a set of hardware. To do this properly would require end-point computers connected to a set of Ethernet switches, thus forming a rudimentary architecture. Even with a simple, stand-alone network of this type, all aspects of the session-layer protocol may be evaluated, from the initialization process through the ability to handle and deflect the various attack vectors. The end-point computers would have some application running that feeds data to the session-layer protocol which then sends packets through the rest of the TCP stack and on to the network switches via Ethernet cable. Ideally, this rudimentary network would allow for performance monitoring as well as packet investigation. Using other infiltration tools, tests may be performed to explore capturing data and injecting fake data packets, and evaluating the performance and behavior of the session-layer protocol, especially in response to adversarial interaction.

The effort to develop such a network would require a team of developers knowledgeable about software, protocols, and communication system development, as well as the assistance of cyber security experts capable of performing various forms of data exfiltration and exploitation. This team would be very helpful in determining the accuracy and systemic benefits of this architecture and security-focused session-layer protocol.

5.4 Conclusion

Can perfectly secure communications exist? It is probably unlikely, because, despite the theoretical constructs, there will always be flaws in systems, whether it is in design or implementation. Can the security of communications be improved to a point such that adversarial interference can be minimized, and without strictly relying on encryption? Existing methods of quantifying security in communication networks are mostly subjective. Without a metric for confidentiality and integrity, it is nearly impossible to state how secure one network is compared to another. The research presented herein demonstrates an approach to measure security, and that it can even be quantified. Using a probabilistic model that considers data leakage and data corruption as surrogates for confidentiality and integrity, a set of QoSS metrics allows the direct and repeatable quantification of the security available in a single-

or multi-channel network under static configurations. The quantification of security is based directly upon the probabilities that adversarial listeners and malicious disrupters are able to gain access to or change the original message. Fragmenting data across multiple channels demonstrates potential improvements to confidentiality through a method of scrambling across a distributed attack surface, while duplicating fragments provides improvements to the integrity of the data despite possible disruptions. The challenge of reconstructing data from multiple channels may push an adversary to reach a point of diminished returns. Simulation results will guide future protocol development to further improve communication security with various feedback mechanisms. The use of encryption can then be layered on top for a multifaceted approach toward security. In the end, no one security mechanism is completely secure, but better mechanisms, or combinations of mechanisms, require more time and effort to defeat. Understanding what security is available in a communication network will allow designers and managers to focus on developing techniques and technologies to keep adversaries one step behind or with but a small fragment of their ultimate goal: useful information.

Bibliography

1. Tech-FAQ, "The osi model – what it is; why it matters; why it doesn't matter," http://www.tech-faq.com/osi-model.html, accessed: 2022-04-15.

2. "Cyber vulnerability enumeration (cve) database," https://cve.mitre.org/, accessed: 2022-05-02.

3. J. Hughes and G. Cybenko, "Quantitative metrics and risk assessment: The three tenets model of cybersecurity," *Technology Innovation Management Review*, vol. 3, no. 8, p. 15, 2013.

4. J. Breier and L. Hudec, "New approach in information system security evaluation," in *2012 IEEE First AESS European Conference on Satellite Telecommunications (ESTEL)*. IEEE, 2012, pp. 1–6.

5. K. Jabbour and J. Poisson, *Cyber risk assessment in distributed information systems*. Army Cyber Institute, West Point, 2016.

6. B. Dunn, "An introduction to secrecy capacity," *IPronounced CHEE-sar*, 2006.

7. A. Gabillon and L. Gallon, "Availability constraints for avionic data buses." in *ARES*. Citeseer, 2006, pp. 124–131.

8. I. Almerhag, A. Almarimi, A. Goweder, and A. Elbekai, "Network security for qos routing metrics," in *Computer and Communication Engineering (ICCCE), 2010 International Conference on*. IEEE, 2010.

9. J. Wang, M. Xia, and F. Zhang, "Metrics for information security vulnerabilities," *Journal of Applied Global Research*, vol. 1, no. 1, pp. 48–58, 2008.

10. J. Barros and M. R. Rodrigues, "Secrecy capacity of wireless channels," in *2006 IEEE international symposium on information theory*. IEEE, 2006, pp. 356–360.

11. M. Bloch, J. Barros, M. R. Rodrigues, and S. W. McLaughlin, "Wireless information-theoretic security-part i: Theoretical aspects," *arXiv preprint cs/0611120*, 2006.

12. ——, "Wireless information-theoretic security-part ii: Practical implementation," *arXiv preprint cs/0611121*, 2006.

13. X. He and A. Yener, "A new outer bound for the secrecy capacity region of the gaussian two-way wiretap channel," in *2010 IEEE International Conference on Communications*. IEEE, 2010, pp. 1–5.

14. G. P. Gallert, “Mapping network protocols to layers of the osi model,” *International Magazine on Advances in Computer Science and Telecommunications*, vol. 1, no. 1, p. 31, 2010.

15. R. M. Gerdes, M. Mina, S. F. Russell, and T. E. Daniels, “Physical-Layer Identification of Wired Ethernet Devices,” *IEEE Trans on Information Forensics and Security*, vol. 7, no. 4, pp. 1339–1353, 2012.

16. G. Surman, “Understanding security using the osi model,” *SANS Institute InfoSec Reading Room*, 2002.

17. A. R. Modarressi and R. A. Skoog, “Signaling system no. 7: A tutorial,” *IEEE Communications Magazine*, vol. 28, no. 7, pp. 19–20, 1990.

18. T. Russell, S. Chapman, and B. Onken, *Signaling system 7*. McGraw-Hill, Inc., 1998.

19. D. J. Pohly and P. McDaniel, “Modeling privacy and tradeoffs in multichannel secret sharing protocols,” in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 371–382.

20. F. Castro-Medina, L. Rodríguez-Mazahua, M. A. Abud-Figueroa, C. Romero-Torres, L. Á. Reyes-Hernández, and G. Alor-Hernández, “Application of data fragmentation and replication methods in the cloud: a review,” in *2019 international conference on electronics, communications and computers (CONIELECOMP)*. IEEE, 2019, pp. 47–54.

21. K. Kapusta and G. Memmi, “Data protection by means of fragmentation in distributed storage systems,” in *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*. IEEE, 2015, pp. 1–8.

22. L. Feng, Y. Zhang, and H. Li, “Large file transmission using self-adaptive data fragmentation in opportunistic networks,” in *2015 Fifth International Conference on Communication Systems and Network Technologies*. IEEE, 2015, pp. 1051–1055.

23. C. D. Cannière and B. Preneel, “Trivium,” in *New stream cipher designs*. Springer, 2008, pp. 244–266.

24. M. Hayden, S. Graham, A. Betances, and R. Mills, “Multi-channel security through data fragmentation,” in *International Conference on Critical Infrastructure Protection*. Springer, 2020, pp. 137–155.

25. K. Shankar, "Special feature the total computer security problem: an oveview," *Computer*, vol. 10, no. 6, pp. 50–73, 1977.

26. B. Archana, A. Chandrashekar, A. G. Bangi, B. Sanjana, and S. Akram, "Survey on usable and secure two-factor authentication," in *Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2017 2nd IEEE International Conference on.* IEEE, 2017, pp. 842–846.

27. J. Marcel, "How bluetooth technology uses adaptive frequency hopping to overcome packet interference," https://www.bluetooth.com/blog/how-bluetooth-technology-uses-adaptive-frequency-hopping-to-overcome-packet-interference/, accessed: 2021-04-25.

28. D. Spill and A. Bittau, "Bluesniff: Eve meets alice and bluetooth." *WooT*, vol. 7, pp. 1–10, 2007.

29. H. Bolcskei, "Mimo-ofdm wireless systems: basics, perspectives, and challenges," *IEEE wireless communications*, vol. 13, no. 4, pp. 31–37, 2006.

30. A. Ford, C. Raiciu, M. Handley, O. Bonaventure, and C. Paasch, "Tcp extensions for multipath operation with multiple addresses," Internet Requests for Comments, RFC Editor, RFC 8684, March 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8684 [Accessed: 2022-08-25]

31. C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath {TCP}," in *9th USENIX symposium on networked systems design and implementation (NSDI 12)*, 2012, pp. 399–412.

32. Y. Liu, Y. Ma, Q. D. Coninck, O. Bonaventure, C. Huitema, and M. Kühlewind, "Multipath Extension for QUIC," Internet Engineering Task Force, Internet-Draft draft-ietf-quic-multipath-02, July 2022, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-quic-multipath/02/ [Accessed: 2022-08-25]

33. S. H. R. Bukhari, M. H. Rehmani, and S. Siraj, "A survey of channel bonding for wireless networks and guidelines of channel bonding for futuristic cognitive radio sensor networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 924–948, 2015.

34. M. Han, S. Khairy, L. X. Cai, and Y. Cheng, "Performance analysis of opportunistic channel bonding in multi-channel wlans," in *2016 IEEE Global Communications Conference (GLOBECOM).* IEEE, 2016, pp. 1–6.

35. S. Lee, T. Kim, S. Lee, K. Kim, Y. H. Kim, and N. Golmie, “Dynamic channel bonding algorithm for densely deployed 802.11 ac networks,” *IEEE Transactions on Communications*, vol. 67, no. 12, pp. 8517–8531, 2019.

36. A. Faridi, B. Bellalta, and A. Checco, “Analysis of dynamic channel bonding in dense networks of wlans,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 8, pp. 2118–2131, 2016.

37. C. Cook and H. Marsh, “An introduction to spread spectrum,” *IEEE communications magazine*, vol. 21, no. 2, pp. 8–16, 1983.

38. V. Moulika, L. Bhagyalakshmi *et al.*, “Performance investigation of cooperative diversity techniques for 5g wireless networks,” in *2019 IEEE 1st International Conference on Energy, Systems and Information Processing (ICESIP).* IEEE, 2019, pp. 1–5.

39. J. Gao, Y. Zhang, and Y. Liu, “A novel diversity receiver design for cooperative transmission system,” *IEEE Access*, vol. 6, pp. 27 176–27 182, 2018.

40. C. Wolfe, S. Graham, and P. Simon, “Securing data in transit using tunable two channel communication,” in *International Conference on Cyber Warfare and Security.* Academic Conferences International Limited, 2018, pp. 627–XVI.

41. J. A. Wampler, C. Hsieh, and A. Toth, “Efficient distribution of fragmented sensor data for obfuscation,” in *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM).* IEEE, 2017, pp. 695–700.

42. D. Dolev, C. Dwork, O. Waarts, and M. Yung, “Perfectly secure message transmission,” *Journal of the ACM (JACM)*, vol. 40, no. 1, pp. 17–47, 1993.

43. R. Lundin, S. Lindskog, and A. Brunstrom, “A model-based analysis of tunability in privacy services,” in *The Future of Identity in the Information Society.* Springer, 2008, pp. 343–356.

44. M. R. Clarkson and F. B. Schneider, “Quantification of integrity,” *Mathematical Structures in Computer Science*, vol. 25, no. 2, pp. 207–258, 2015.

45. I. Sweet, J. M. C. Trilla, C. Scherrer, M. Hicks, and S. Magill, “What’s the over/under? probabilistic bounds on information leakage,” in *International Conference on Principles of Security and Trust.* Springer, Cham, 2018, pp. 3–27.

46. A. Garnaev and W. Trappe, “The eavesdropping and jamming dilemma in multichannel communications,” in *2013 IEEE International Conference on Communications (ICC).* IEEE, 2013, pp. 2160–2164.

47. V. Ciriani, S. D. C. D. Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati, “Combining fragmentation and encryption to protect privacy in data storage,” *ACM Transactions on Information and System Security (TISSEC)*, vol. 13, no. 3, pp. 1–33, 2010.

48. A. Khisti and T. Liu, “Private broadcasting over independent parallel channels,” *IEEE transactions on information theory*, vol. 60, no. 9, pp. 5173–5187, 2014.

49. Q. Duan, “Modeling and analysis of end-to-end quality of service provisioning in virtualization-based future internet,” in *2010 Proceedings of 19th International Conference on Computer Communications and Networks.* IEEE, 2010, pp. 1–6.

50. V. Firoiu, J.-Y. Le Boudec, D. Towsley, and Z.-L. Zhang, “Theories and models for internet quality of service,” *Proceedings of the IEEE*, vol. 90, no. 9, pp. 1565–1591, 2002.

51. M. Pitkanen, A. Keranen, and J. Ott, “Message fragmentation in opportunistic dtns,” in *2008 International Symposium on a World of Wireless, Mobile and Multimedia Networks.* IEEE, 2008, pp. 1–7.

52. J. L. Hennessy and D. A. Patterson, *Computer architecture: a quantitative approach.* Elsevier, 2011.

53. E. Creedon and M. Manzke, “Impact of fragmentation strategy on ethernet performance,” in *2009 Sixth IFIP International Conference on Network and Parallel Computing.* IEEE, 2009, pp. 30–37.

54. P. G. Leon and A. Saxena, “An approach to quantitatively measure information security,” in *Proceedings of the 3rd India Software Engineering Conference, ISEC*, 2010.

55. P. M. Simon and S. R. Graham, “Probability of data leakage and its impacts on confidentiality,” in *ECCWS 2022 21st European Conference on Cyber Warfare and Security.* Academic Conferences and publishing limited, 2022.

56. K. McKay and D. Cooper, “Guidelines for the selection, configuration, and use of transport layer security (tls) implementations,” National Institute of Standards and Technology, Tech. Rep., 2017.

57. J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, “Crystals-kyber: a cca-secure module-lattice-based kem,” in *2018 IEEE European Symposium on Security and Privacy (EuroS&P).* IEEE, 2018, pp. 353–367.

58. S. B. Wicker and V. K. Bhargava, "An introduction to reed-solomon codes," *Reed-Solomon codes and their applications*, pp. 1–16, 1994.

59. G. D. Forney, "The viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

60. M. R. Clarkson, "Quantification and formalization of security," Ph.D. dissertation, Cornell University, 2010.

61. N. Parveen, M. R. Beg, and M. Khan, "Model to quantify confidentiality at requirement phase," in *Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering & Technology (ICARCSET 2015)*, 2015, pp. 1–4.

62. P. M. Simon, S. Graham, C. Talbot, and M. Hayden, "Model for quantifying the quality of secure service," *Journal of Cybersecurity and Privacy*, vol. 1, no. 2, pp. 289–301, 2021.

63. M. J. Abdel-Rahman, H. K. Shankar, and M. Krunz, "Qos-aware parallel sensing/probing architecture and adaptive cross-layer protocol design for opportunistic networks," *IEEE transactions on vehicular technology*, vol. 65, no. 4, pp. 2231–2242, 2015.

64. K. Srinathan, A. Narayanan, and C. P. Rangan, "Optimal perfectly secure message transmission," in *Annual International Cryptology Conference.* Springer, 2004, pp. 545–561.

65. Y. Wang and Y. Desmedt, "Perfectly secure message transmission revisited," *IEEE Transactions on Information Theory*, vol. 54, no. 6, pp. 2582–2595, 2008.

66. A. Hudic, S. Islam, P. Kieseberg, and E. R. Weippl, "Data confidentiality using fragmentation in cloud computing," *Int. J. Communication Networks and Distributed Systems*, vol. 1, no. 3/4, 2012.

67. M. Rausand and A. Høyland, *System Reliability Theory: Models, Statistical Methods, and Applications.* John Wiley & Sons, 2003, vol. 396.

68. A. Patra, A. Choudhury, C. Pandu Rangan, and K. Srinathan, "Unconditionally reliable and secure message transmission in undirected synchronous networks: Possibility, feasibility and optimality," *International Journal of Applied Cryptography*, vol. 2, no. 2, pp. 159–197, 2010.

69. J. Menčík, *Concise reliability for engineers.* BoD–Books on Demand, 2016.

70. P. M. Simon and S. Graham, "Extending the quality of secure service model to multi-hop networks," *Journal of Cybersecurity and Privacy*, vol. 1, no. 4, pp. 793–803, 2021.

71. G. Spini and G. Zémor, "Efficient protocols for perfectly secure message transmission with applications to secure network coding," *IEEE Transactions on Information Theory*, vol. 66, no. 10, pp. 6340–6353, 2020.

72. E. Winjum and T. J. Berg, "Multilevel security for ip routing," in *MILCOM 2008-2008 IEEE Military Communications Conference*. IEEE, 2008, pp. 1–8.

73. J. Jin and M. Shen, "Analysis of security models based on multilevel security policy," in *2012 International Conference on Management of e-Commerce and e-Government*. IEEE, 2012, pp. 95–97.

74. A. Varga and R. Hornig, "Omnet++ discrete event simulator," https://www.omnetpp.org/, accessed: 2022-04-15.

75. ——, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, 2008, pp. 1–10.

76. J. Tolkien, *The Hobbit*. HarperCollins, 2012.

77. P. Koopman, "Best crc polynomials," https://users.ece.cmu.edu/~koopman/crc/, accessed: 2022-04-15.

78. P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson, "Raid: High-performance, reliable secondary storage," *ACM Computing Surveys (CSUR)*, vol. 26, no. 2, pp. 145–185, 1994.

79. Z. Hu, Y. Vasiliu, O. Smirnov, V. Sydorenko, and Y. Polishchuk, "Abstract model of eavesdropper and overview on attacks in quantum cryptography systems," in *2019 10th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1. IEEE, 2019, pp. 399–405.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704–0188*

1. REPORT DATE *(DD–MM–YYYY)* 15–09–2022	**2. REPORT TYPE** Doctoral dissertation	**3. DATES COVERED** *(From — To)* October 2013 — September 2022

4. TITLE AND SUBTITLE Development of a Security-Focused Multi-Channel Communication Protocol and Associated Quality of Secure Service (QoSS) Metrics	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Simon, Paul M., Ctr.	**5d. PROJECT NUMBER**
	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-DS-22-S-038
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**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

The threat of eavesdropping, and the challenge of recognizing and correcting for corrupted or suppressed information in communication systems is a consistent challenge. Effectively managing protection mechanisms requires an ability to accurately gauge the likelihood or severity of a threat, and adapt the security features available in a system to mitigate the threat. This research focuses on the design and development of a security-focused communication protocol at the session-layer based on a re-prioritized communication architecture model and associated metrics. From a probabilistic model that considers data leakage and data corruption as surrogates for breaches of confidentiality and integrity, a set of metrics allows the direct and repeatable quantification of the security available in single- or multi-channel networks. The quantification of security is based directly upon the probabilities that adversarial listeners and malicious disruptors are able to gain access to or change the original message. Fragmenting data across multiple channels demonstrates potential improvements to confidentiality, while duplication improves the integrity of the data against disruptions. Finally, the model and metrics are exercised in simulation. The ultimate goal is to minimize the information available to adversaries.

15. SUBJECT TERMS

None

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Scott Graham, AFIT/ENG
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			**19b. TELEPHONE NUMBER** *(include area code)*
U	U	U	UU	144	(937) 255-6565 x4581; scott.graham@afit.edu