



Calhoun: The NPS Institutional Archive
DSpace Repository

Theses and Dissertations

1. Thesis and Dissertation Collection, all items

2022-06

UXS AUTHENTICATION AND KEY EXCHANGE REQUIREMENTS FOR MULTIDOMAIN OPERATION AND JOINT INTEROPERABILITY

Leon, Andre; Britt, Christopher J.

Monterey, CA; Naval Postgraduate School

<http://hdl.handle.net/10945/70738>

This publication is a work of the U.S. Government as defined in Title 17, United States Code, Section 101. Copyright protection is not available for this work in the United States.

Downloaded from NPS Archive: Calhoun



Calhoun is the Naval Postgraduate School's public access digital repository for research materials and institutional publications created by the NPS community. Calhoun is named for Professor of Mathematics Guy K. Calhoun, NPS's first appointed -- and published -- scholarly author.

Dudley Knox Library / Naval Postgraduate School
411 Dyer Road / 1 University Circle
Monterey, California USA 93943

<http://www.nps.edu/library>



**NAVAL
POSTGRADUATE
SCHOOL**

MONTEREY, CALIFORNIA

THESIS

**UXS AUTHENTICATION AND KEY EXCHANGE
REQUIREMENTS FOR MULTIDOMAIN OPERATION
AND JOINT INTEROPERABILITY**

by

Andre Leon and Christopher J. Britt

June 2022

Thesis Advisor:
Co-Advisor:

Britta Hale
Aurelio Monarrez Jr.

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.			
1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE June 2022	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE UXS AUTHENTICATION AND KEY EXCHANGE REQUIREMENTS FOR MULTIDOMAIN OPERATION AND JOINT INTEROPERABILITY		5. FUNDING NUMBERS	
6. AUTHOR(S) Andre Leon and Christopher J. Britt			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A		10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.		12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Within the Joint All Domain Command and Control (C2) sensor network and the Navy's Project Overmatch, unmanned systems (UxS) are a shared capability that extends reach and capacity of the military force to enhance tactics in contested spaces. This has increased research into interoperable network frameworks to securely and efficiently C2 distributed UxS forces. To date, antiquated technologies, stove-piped and proprietary business practices limit or obscure the pursuit of emerging industry techniques that provide security features required for today's modernized force—leaving more questions than facts. Moreover, UxS power and processing limitations and constrained operating environments prohibit the use of existing modern communications protocols. However, developments in message layer security (MLS), a secure and efficient group communication protocol, could be the ideal choice for UxS teaming. This thesis documents results gathered from a qualitative study that finds MLS the best option for UxS group security and efficiency. It also documents the integration of MLS into the ScanEagle unmanned aerial vehicle (UAV) and Naval Information Warfare Pacific CASSMIR unmanned surface vehicle (USV). The implementation provides a concept of operation to demonstrate the use of MLS to provide secure and efficient C2 and exchange of data between the UAV and USV in a multi-domain ad-hoc network configuration. The experiments conducted are in a virtual environment and the physical UxS.			
14. SUBJECT TERMS authentication, key exchange, multidomain operation, Joint Interoperability, command and control, C2, unmanned systems, UxS, unmanned surface vehicle, USV, message layer security, MLS, unmanned aerial vehicle, UAV		15. NUMBER OF PAGES 107	
		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**UXS AUTHENTICATION AND KEY EXCHANGE REQUIREMENTS
FOR MULTIDOMAIN OPERATION AND JOINT INTEROPERABILITY**

Andre Leon
Lieutenant, United States Navy
BBA, Florida Atlantic University, 2012

Christopher J. Britt
Lieutenant, United States Navy
AA, University of Phoenix, 2006
BS, University of Phoenix, 2016

Submitted in partial fulfillment of the
requirements for the degrees of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

and

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
June 2022**

Approved by: Britta Hale
Advisor

Aurelio Monarrez Jr.
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

Alex Bordetsky
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Within the Joint All Domain Command and Control (C2) sensor network and the Navy's Project Overmatch, unmanned systems (UxS) are a shared capability that extends reach and capacity of the military force to enhance tactics in contested spaces. This has increased research into interoperable network frameworks to securely and efficiently C2 distributed UxS forces. To date, antiquated technologies, stove-piped and proprietary business practices limit or obscure the pursuit of emerging industry techniques that provide security features required for today's modernized force—leaving more questions than facts. Moreover, UxS power and processing limitations and constrained operating environments prohibit the use of existing modern communications protocols. However, developments in message layer security (MLS), a secure and efficient group communication protocol, could be the ideal choice for UxS teaming. This thesis documents results gathered from a qualitative study that finds MLS the best option for UxS group security and efficiency. It also documents the integration of MLS into the ScanEagle unmanned aerial vehicle (UAV) and Naval Information Warfare Pacific CASSMIR unmanned surface vehicle (USV). The implementation provides a concept of operation to demonstrate the use of MLS to provide secure and efficient C2 and exchange of data between the UAV and USV in a multi-domain ad-hoc network configuration. The experiments conducted are in a virtual environment and the physical UxS.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	Introduction	1
1.1	Problem Statement	2
1.2	Scope	3
1.3	Related Research	4
1.4	Thesis Organization	5
2	Background: Unmanned Systems in the Department of Defense	7
2.1	Unmanned Systems	7
2.2	Chapter Summary	15
3	UxS Security Study	17
3.1	Methodology	17
3.2	Study Results	20
3.3	Interpretation	27
3.4	Study Conclusion	30
4	Protocol Alignment and Selection	31
4.1	Standardized Protocols	32
4.2	Proprietary-Based Protocols	34
4.3	Protocol Selection	37
4.4	Chapter Summary	38
5	UxS MLS Implementation	41
5.1	MLS Design Concept	41
5.2	MLS Architecture	41
5.3	Robot Operating System (ROS).	46
5.4	Use Case Methodology	46

5.5	Chapter Summary	58
6	Simulation and Results	59
6.1	Phase One: MLS Chat	59
6.2	Phase Two: MLS ROS	60
6.3	Phase Three: MLS C2	64
6.4	Limitations.	68
6.5	Chapter Summary	69
7	Conclusion and Future Work	71
7.1	Conclusion.	71
7.2	Future Work	71
	Appendix	75
A.1	Recruitment Sample Email	75
A.2	Interview Questionnaire	76
A.3	Repository Links	80
	List of References	81
	Initial Distribution List	89

List of Figures

Figure 2.1	NSA approved ciphersuites for TLS	14
Figure 2.2	NSA approved ciphersuites for IPsec	15
Figure 4.1	Security protocol properties and features comparison	38
Figure 5.1	MLS Architecture	42
Figure 5.2	MLS Protocol Functionality Sequence	45
Figure 5.3	char* to bytes conversion function	50
Figure 5.4	bytes to char* conversion function	50
Figure 5.5	Functionality for chat client	52
Figure 5.6	Functionality for chat server	54
Figure 5.7	Depicts architecture used to test the MLS C2 Application	56
Figure 5.8	MLS C2 Application third thread design.	58
Figure 6.1	Overhead cost per ciphersuite comparison	60
Figure 6.2	Screenshot of MLS ROS application testing prompt	62
Figure 6.3	Ciphersuite setup time comparison	63
Figure 6.4	Message handling time comparison	64
Figure 6.5	ScanEagle UAV virtual environment	65
Figure 6.6	CASSMIR USV platform	66
Figure 6.7	ScanEagle and CASSMIR data exchange	67
Figure 6.8	ScanEagle receiving C2 commands from ground station	68

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

A2/AD	Anti-Access and Area Denial
AAD	Active Adversary Detection
AES	Advanced Encryption Standard
AKE	Authenticated Key Exchange
API	Application Programming Interface
AS	Authentication Service
ATO	Authority to Operate
C2	Command and Control
CASSMIR	Collaborative Autonomous Systems for Standoff Maritime Inspection and Response
CAVR	Center for Autonomous Vehicle Research
CSfC	Commercial Solutions for Classified
CP	Capability Package
COTS	Commercial Off-The-Shelf
DH	Diffie–Hellman
DHE	Diffie–Hellman Ephemeral
DOD	Department of Defense
DON	Department of the Navy
DS	Delivery Service
DSA	Digital Signature Algorithm

ECDSA	Elliptic Curve DSA
FS	Forward security
GCM	Galois Counter Mode
IAS	Intelligent Autonomous Systems
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPSec	Internet Protocol Security
JADC2	Joint All Domain Command and Control
JREAP	Joint Range Extension Application Protocol
KDF	Key Derivation Function
MDO	Multi-Domain Operations
MHz	Megahertz
MitM	Man-in-the-Middle
MA	Mobile Access
MLS	Message Layer Security
NIST	National Institute of Standard and Technology
NIWC-PAC	Naval Information Warfare Command-Pacific
NPS	Naval Postgraduate School
NSA	National Security Agency
NSS	National Security Systems

PCS	Post-Compromise Security
PGP	Pretty Good Privacy
ROS	Robotic Operating System
RMF	Risk Management Framework
RSA	Rivest–Shamir–Adleman
SHA	Secure Hash Algorithm
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UAV	Unmanned Aerial Vehicle
UDP	User Datagram Protocol
USN	U.S. Navy
USV	Unmanned Surface Vehicle
UxS	Unmanned Systems
VM	Virtual Machine

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

Thank you, Allu and Gizette; we could not have achieved this significant career milestone without your unconditional love and support. Thank you, Dr. Britta Hale; we appreciate all your knowledge, guidance, and patience through the long hours you devoted to helping to make this thesis and our degrees possible. We both are extremely grateful. Special thanks to Anthony and the CASSMIR team, the NPS CAVR Lab, Matt and Jose from Astroforge, and all those who participated in the study associated with our thesis research.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:

Introduction

Essential to the Joint All-Domain Command and Control (JADC2) architecture is the diverse array of unmanned systems (UxS) and sensors. These distinct devices will interconnect the future maritime force centered around human-machine teaming.

Consider, for example, a joint-all-domain use case of UxS delivering capabilities against near-peer adversaries. Command and control (C2) of the UxS is reliant on communication links—the security and design of which determines the capacity to scale at speed, interoperability, and harm in case of adversarial attack. In contrast, inadequacies or use of legacy architectures in the same C2 communication links translate to tactical and strategic disadvantages that could potentially place the traditional combatant force in harm’s way. Our research aims to identify and implement a viable C2 link security option that has the potential to provide a secure, scalable, and interoperable solution for UxS in a distributed multi-domain environment.

Currently, the Department of Defense (DOD) and Department of the Navy (DON) are making significant strides to take advantage of the unique missions and opportunities across the enterprise [1]. These new possibilities include the increased employment of unmanned systems and sensors beyond the current use-case platforms. When enabling data sharing across unmanned platforms and systems, cybersecurity must be considered a top priority among the many-core technologies. These efforts must consider securing key enablers such as networks, infrastructure, and C2 with robust security protocols and authentication methods. These considerations will become vital as the JADC2 enterprise attempts to shift from stove-piped to more unified data environments that will be accessible to all amidst adversaries that have developed highly sophisticated Anti-access and area denial (A2/AD) capabilities [2].

UxS as distributed force multipliers in today’s modern warfare will depend on secure and efficient C2. As UxS developments mature, the demand for interoperability will increase. This research analyzes current and emerging security protocols and matches them to JADC2 and Project Overmatch requirements to assess and identify optimal properties and protocols

that support them. This work then selects the Messaging Layer Security (MLS) protocol based on desired security alignments for implementation on a UxS platform for viability, specifically the program of record ScanEagle unmanned aerial vehicle (UAV).

Near peer adversaries continue to pursue A2/AD capabilities to defeat traditional U.S. military power. Suppose the cyber and physical security attributes of the UxS go unaddressed or are poorly designed. Then other core technologies such as positioning, navigation and timing, reliability, interoperability, communications, and the platform's ability to sense and decide can become degraded or compromised. Migrating from the current siloed and antiquated accreditation processes to an efficient integrated development, security, and operations environment is essential to successfully incorporate UxS platforms and sensors within the JADC2 environment. This migration is also challenged by the need to rapidly evolve from legacy-based technologies and development frameworks to rapidly-emerging technologies that are more capable of remaining relevant amidst advances of near-peer threats [3]. Solving this challenge will require fusing technological barriers with cultural, fiscal, procedural, and political siloes [4]. Once the DOD solves this challenge, it will then be empowered to achieve seamless integration, synchronization, and security necessary for UxS to become multi-domain operations force multipliers.

1.1 Problem Statement

In an era of rapid technological competition, the JADC2 infrastructure relies upon technologies conceptualized in the 1990s (e.g., IPsec [Internet Protocol Security] and TLS [Transport Layer Security]), while being restricted to the common protocols and standards set by the National Security Agency (NSA) to communicate securely [5]. These network security protocols are point-to-point, requiring separate channel establishment with every existing network device for each new one added to the command overview. Although cutting-edge at inception, it is notable that we still rely on such point-to-point security connections today, decades later, forcing a high-latency and outmoded security overlay on top of dynamic autonomous device mesh networks. Integrating improvements not only requires assessment of appropriate modern alternatives, but also an Authority to Operate (ATO) procedure capable of addressing UxS security challenges with emergent industry solutions in a timely and effective manner.

To address these problems we developed the following research questions:

- What are JADC2 and Project Overmatch C2 protocol security requirements?
- What are the features needed from a modern C2 security protocol to meet JADC2 environment needs, based on DOD subject matter experts working in JADC2 related domains?
- Which security protocol(s) can best address all of these needs, and what is the use viability for UxS C2 links?

1.2 Scope

This research supports discussion throughout the DOD and DON on current norms, and that inadequate cybersecurity practices and accreditation processes must not hinder the future state of communications security of UxS. These methodologies must evolve to adequately address the increasing need for both speed and security in our sensors and unmanned assets amidst highly technical peer rival threats.

This research assesses and guides security and network resiliency development options for autonomous devices and sensors. This assessment leverages use cases and lessons from industry and military applications of secure and unprotected UxS systems and deployment models. Additionally, this research considers currently used synchronous protocols and the development of ratcheting and asynchronous protocols. These assessments assists in selecting a protocol that can be best implemented in a use case specifically tailored for multi-domain agile group communication of UxS platforms.

Based on study results, a selected protocol is implemented, tested, and bench-marked virtually in a controlled lab environment. Upon successful completion of controlled virtual testing, the virtual implementation transitions to practical application on the NPS Center for Autonomous Vehicle Research (CAVR) ScanEagle unmanned aerial vehicle (UAV) and the Navy Information Warfare Center-Pacific (NIWC-PAC) Collaborative Autonomous Systems for Standoff Maritime Inspection and Response (CASSMIR) unmanned surface vehicle (USV).

For the purpose of this thesis, there is no distinction made between *unmanned systems* and *unmanned vehicles*, regardless of domain, i.e., air and surface; all are referred to as UxS.

However, during experimentation, testing will occur on a UAV and USV. This research aims to address the need for a C2 link security protocol solution that is platform agnostic.

Achieving the following primary objectives constitutes this thesis' contributions through a mixed-method (qualitative and quantitative) research undertaking:

- Conduct a qualitative study that identifies UxS security protocol needs for JADC2 and Project Overmatch.
- Align qualitative study results with an assessment of current military and industry security protocol options.
- Select a viable security protocol option for a multi-domain operation (MDO) UxS use case.
- Implement and evaluate the selected security protocol under optimal network conditions for a UxS simulation.
- Implement and evaluate the selected security protocol on ScanEagle and CASSMIR.

1.3 Related Research

The UxS research space is vast and evolving. As discussed in this section, the topic of UxS security has been researched and documented in various research fields. However, there is limited research comparing different protocols against military requirements to develop a C2 protocol standard for enhanced security, efficiency, and interoperability. Nonetheless, selected prior studies provide insight that relates to or supports our research.

Researchers from the University of Oregon, University of South Florida, Naval Post Graduate School, and Case Western Reserve University, focus on establishing the most efficient ciphertext algorithm or cryptologic framework based on a balance between performance and security [6]–[8]. These papers explain that our current most used ciphersuites are too computational and power intensive for small UxS such as the Craziefile 2.0 which uses an ARM Cortex M-4 architecture and operates at 168 MHz. Other research studies the security primitives of the software used to develop UxS such as of the Robot Operating System (ROS), and explains the security vulnerabilities and mitigation's for safe, reliable deployment of unmanned systems [9]. This last study sheds light on how vulnerable these underlying technologies are and the need to protect them.

From the related research, there is an emphasis on finding the best topology, routing protocol, or data messaging to support an ever-increasing amount of UxS and sensors to work and operate together [10], [11]. Most of these works aim to find the most efficient way to maintain C2 by reducing the overhead cost of transmission to a minimum [10], [11]. Other UxS research topics focus on cybersecurity best practices, emphasizing the ranges between vulnerabilities found in unmanned systems to possible new attack vectors and possible mitigation techniques [12].

There is an overabundance of guidance and research that outline requirements and solutions; however, none have genuinely quantified the importance of C2 link security for UxS platforms and sensors within the DOD and the Navy. Even less guidance and research match protocols and algorithms to such needs. The related research shows that these perspectives do not directly cover the selection and use of standardized protocols holistically to improve UxS C2 link security, efficiency, and interoperability. These approaches consider the internal performance of ciphersuites, security services and capabilities of the ROS software, UxS vulnerabilities, and overall network performance. This thesis aims to investigate the implementation of a standardized security protocol that can serve as an application-layer security software that is agnostic to device and Internet Protocol network.

There is an overabundance of guidance and research that outline security needs for UxS; however, none have genuinely quantified these security needs for military use. real-world experience from the importance of C2 link security for UxS platforms and sensors within the DOD and the DON.

1.4 Thesis Organization

The remainder of this thesis is organized as follows:

Chapter 2 provides an overview of JADC2 and Project Overmatch initiatives to understand the security protocol requirements to these initiatives. This chapter also discusses the roles National Institute of Standards and Technology (NIST) and the NSA play in the standardization and selection of cryptographic protocols. It reviews the industry and military security approaches to secure communication protocols, associated performance, and security concerns that are addressed through the use of proprietary and standards based

security protocols.

Chapter 3 provides a qualitative study consisting of cybersecurity-oriented interview questions. Subjects from the study are military, civilian, and contractor personnel with experience in security, autonomous device and sensor networking, acquisition, or overlaps. The data collected from the interviews provide a deeper understanding of the current state of communications security of UxS and associated cybersecurity and accreditation processes for the DOD and DON.

Chapter 4 provides a protocol comparison and selection based on cross-analysis of the results from Chapter 3 and Chapter 2. It discusses proprietary and standardized security protocols that are critical cybersecurity components to the networks and initiatives discussed in Chapter 2. It also matches DOD and DON UxS security requirements with the results from the qualitative study, and the current and emerging security protocols discussed in order to select the most capable security protocol desired for C2 of UxS platforms.

Chapter 5 outlines the methodology and implementation of MLS for an MDO UxS scenario. It describes the MLS and ROS architectures. It outlines protocol functionality overview, code development phases, and core functions created to support the implementation. It also covers an overview of the step-by-step methodologies used to create the MLS command and control (C2) application (MLS C2) to interface with ROS.

Chapter 6 discusses experimentation with the various MLS applications developed in 5 and analyzes its impact on the research use case. This chapter includes a description of the testing process and describes the results.

Chapter 7 provides a conclusion covering the implications of this thesis research, concludes the study, and recommends options for continued work and alternate approaches.

CHAPTER 2: Background: Unmanned Systems in the Department of Defense

This chapter provides background information on the overarching DOD and DON strategies and requirements that enable JADC2 and Project Overmatch to allow the warfighting advantages required to prevail in day-to-day competition, crisis, and conflict. It also highlights the lack of consideration for incorporating C2 link security protocols for UxS platforms and sensors into the architectures above. This chapter also describes the information found on National Institute of Standards and Technology (NIST) standards, the Risk Management Framework (RMF) process, and National Security Agency (NSA) security products.

2.1 Unmanned Systems

To fully achieve mission objectives and success in modern warfare, protocols and associated algorithms are at the heart of how two or more UxS will securely communicate with each other. The DOD's unique mission sets and associated warfighter requirements should drive protocol design and selection. UxS have historically focused on efficiency and reliability to meet operational necessities and economic constraints opposed to interoperability requirements. Today, these same systems undergo post-design modifications to account for interoperability [13]. What further complicates the use and selection of protocols for efficient interoperability is the requirement to interface with the myriad of existing legacy military systems still in use today [14].

From the newly transmitted 2022 National Defense Strategy (NDS) to service component mission statements, there is a clear correlation between national defense and service priorities to the relevance of C2 link security of UxS platforms and sensors. Per the Naval Intelligent Autonomous Systems (IAS) strategy, "Recent conflicts are proving that smaller asymmetric forces can decisively dominate traditionally larger superior forces" [4]; further emphasizing this correlation enables the DOD to select meaningful and relevant protocols as modern warfare tactics for UxS capabilities integrated within JADC2 and Project Overmatch.

2.1.1 JADC2 and Project Overmatch

Per the Congressional Research Service, “JADC2 is the DOD’s concept to connect sensors from all of the military services—Air Force, Army, Marine Corps, Navy, and Space Force—into a single network” [2]. For this research, UxS platforms function as nodes within the enterprise and are analogous to “sensors.”

Historically, tactical military networks have been created in silos, leading to interoperability gaps among the service branches [15]. These stove-piped architectures do not permit the timely issuance and execution of commands. With the volatility of today’s modernized operating environment and threats, these approaches render current networks incapable of addressing the priorities outlined in the NDS, and highlight a need for targeted planning for future UxS interoperability.

Project Overmatch is the Navy’s sub-component to JADC2. Its goal is to synergize Navy and Marine Corps global capabilities for maritime dominance. As stated by NAVWAR this is achieved through, “delivering synchronized lethal and non-lethal effects from near-and-far, on every axis and every domain” [16].

JADC2 and Project Overmatch goals share priorities and challenges that span multiple strategic visions. For example, common threads outlined in the NDS and Chief of Naval Operation (CNO) NAVPLAN 21, such as multi-domain threats, strong joint and allied partnerships, and building a resilient joint force, will position the DON for success [17]. These goals guide the subsequent UxS Strategies and Unmanned Campaign Plan. Therefore, the following UxS strategies should drive security protocol development standards for C2 link security that meets the efficiency and interoperability requirements desired for UxS. However, such C2 link security protocol standards are not yet described [3], [4], [18].

2.1.2 DOD UxS Strategies

As called for in the CNO NAVPLAN 21, to deliver a superior all-domain naval power, the Navy requires a host of integrated unmanned platforms—under, on, and above the sea [19]. Furthermore, there is also a need for a robust and secure network infrastructure capable of linking together distributed forces. The DON Unmanned Campaign Framework and IAS strategy re-emphasize the need to adopt creative solutions that leverage the latest innovations

to seamlessly integrate UxS platforms and sensors as part of a superior force.

Unmanned Campaign Plan

The Navy's Unmanned Campaign plan aims to integrate UxS into the Navy's maritime arsenal at both speed and scale. These systems must be trustworthy and sustainable across the fleet. The plan's goals are as follows:

- Advance manned-unmanned teaming within the full range of Naval and joint operations;
- Build a digital infrastructure that integrates and adopts unmanned capabilities at speed and scale;
- Incentivize rapid incremental development and testing cycles for unmanned systems;
- Disaggregate common problems, solve once, and scale solutions across platforms and domains;
- Create a capability centric and sustainable approach for unmanned contributions (platforms, systems, subsystems) to the force. [3]

The DOD must seize the advantages of an enterprise comprised of interconnected sensors from all services and partners. To hold this advantage requires levels of creativity and ingenuity that break the mold of traditional paradigms. By doing so, the DOD will enable the Unmanned Campaign Plan's vision to achieve technological and cultural change.

Intelligent Autonomous Systems

Similar to the vision and goals outlined in the Unmanned Campaign Plan, the IAS strategy seeks to synthesize the intersections of UxS, AI, and Autonomy into a future enabled by IAS [4]. The strategy aims to establish a logical and consistent strategic investment framework that will accelerate the development, operationalization, and adoption of practices that leverages today's cutting-edge technologies.

As the DOD and DON strategies seek to rise to the challenge of rapidly improving conceptual and proven capabilities at a pace and scale to outstrip adversarial advantage, the DOD and DON are providing actionable and measurable steps to initiatives such as JADC2 and Project Overmatch. Before rewriting the narrative of traditional warfare to where UxS platforms

and sensors provide the required force, agility, and competitive advantage, it is necessary to consider the full spectrum of existing requirements. These requirements, particularly the NIST RMF, and associated accreditation processes, often impede incorporating unmanned systems and security at the speed of technology, and places UxS visions and goals farther from reach [20].

2.1.3 National Institute of Standards and Technology

Since 2017, federal agencies must follow NIST compliance standards [21]. Understanding the federal compliance world is no easy task when navigating through a myriad of acronyms that exists in this space. As DOD UxS strategies attempt to impact change, the NIST RMF and Cybersecurity Framework provide a broad level of guidance that the DOD will strictly follow.

Risk Management Framework

DOD and DON organizations must abide by NIST RMF process for system development. This framework provides the required criteria that support the application of risk management plans necessary to fulfill Federal Information Security Management Act mandates (FISMA) [22]. The RMF framework aims to be a comprehensive multi-step process that can easily be measured and repeated, providing DOD organizations with a standardized mechanism to manage privacy and security risks before a military system reaches full operating capability [18]. Below is an excerpt of the multi-step RMF Process definitions by NIST.

- **Prepare:** Essential activities to prepare the organization to manage security and privacy risks.
- **Control:** Categorize the system and information processed, stored, and transmitted based on an impact analysis.
- **Select:** Select the set of NIST SP 800-53 controls to protect the system based on risk assessment(s).
- **Implement:** Implement the controls and document how controls are deployed.
- **Assess:** Assess to determine if the controls are in place, operating as intended, and producing the desired results.

- **Authorize:** Senior official makes a risk-based decision to authorize the system (to operate).
- **Monitor:** Continuously monitor control implementation and risks to the system. [18]

All steps from the RMF are essential when integrating security into the system development life cycle of UxS. However, this research only analyzes steps three and six of the RMF since they provide insight into the selection and authorization of security control requirements for UxS.

RMF: Select Controls

The NIST SP800-53 defines security controls as, “a safeguard or countermeasure prescribed for an information system or an organization designed to protect the confidentiality, integrity, and availability of its information and to meet a set of defined security requirements” [23]. During the development of UxS platforms and sensors, while executing the RMF, the appropriate security and privacy controls for the device should be selected before transitioning to the Authorize step. Organizations have to reduce risk that could surface from the use of UxS during operations [24]. A major challenge in this mitigation process is determining the appropriate set of security controls. These controls are based on the prescribed requirements and impact analysis conducted in the prior prepare and categorize steps of the RMF.

To effectively conduct security control selection, the organization must categorize the impact on the UxS security objectives of confidentiality, integrity, or availability per Federal Information Processing Standards (FIPS) 199 and 200 [25], [26]. These categories range across low-impact, moderate-impact, or high-impact for each type of information processed, stored, or transmitted via the UxS. Once the security categorization process determines the impact level of the system, the organization selects one of three sets of baseline security control categories from NIST SP 800-53B [23]. These categories are low, medium, and high and are associated with the specified access controls that closely align with the system impact ratings. Following selection of preliminary security controls, it is necessary to customize these controls. This process is essential to subsequent RMF steps, in this case, the Authorize step, to appropriately modify and align the security controls more closely with the conditions specific to the system or its operating environment.

RMF: Authorize

Before a UxS is promoted to production, becomes operational, or receives a significant change to platform baseline, it must first receive an Authority to Operate (ATO). Before ATO approval, the UxS owner gathers all relevant information and materials into what is known as the authorization package. The authorization package contains security and privacy plans, assessment reports, and individual action plans and milestones. The authorizing official (AO) then reviews and decides to approve or deny the systems authorization package. The AO is the designated senior official held accountable as the ATO approval authority, ensuring all risk determinations and responses are adequately met based on the selected security controls. As the development life-cycle of UxS enters the “Authorize” step of the RMF, there is no room for error when categorizing the appropriate security controls from the prior “Select Controls”. Inaccuracy in identifying the appropriate security controls can incur significant delay in UxS production or lead to the lessening of overall system protections.

Although NIST is the military go-to for advanced communications and cybersecurity frameworks, its documentary standards and reference data fall short of providing the required details to select appropriate UxS and C2 link security protocol options. NIST SP 800-37 Rev 2 includes updated alignment to the NIST cybersecurity framework profiles, which considers the need for modern cryptography, yet only provides a broad overview of FIPS impact levels [24]–[26]. Conversely, Tables 3-1 through 3-20 of NIST SP 800-53 B provide a detailed listing of security controls and control enhancements, yet also lack protocol specificity for National Security Systems (NSS) that the UxS platforms in this research fall under [23]. NIST defines NSS as follows:

Any information system (including any telecommunications system) used or operated by an agency or by a contractor of an agency, or other organization on behalf of an agency— (i) the function, operation, or use of which involves intelligence activities; involves cryptologic activities related to national security; involves command and control of military forces; involves equipment that is an integral part of a weapon or weapons system; or is critical to the direct fulfillment of military or intelligence missions (excluding a system that is to be used for routine administrative and business applications, for example, payroll, finance, logistics, and personnel management applications); or (ii) is protected

at all times by procedures established for information that have been specifically authorized under criteria established by an Executive Order or an Act of Congress to be kept classified in the interest of national defense or foreign policy. [18]

2.1.4 NSA

Military UxS developers and organizations must also consider NSA roles and responsibilities. To emphasize this point, the National Security Directive-42, designates the NSA as the manager for all NSS [27]. Additionally, as stated in the NIST Cryptographic Standards and Guidelines Development, the NIST works closely with federal agencies such as the NSA to develop its cybersecurity standards and guidelines, including ciphersuites [28]. NSA approved ciphersuites are defined as a set of cryptographic algorithms that enable the implementation of various encryption schemes for secure communication.

Per CNSSI No. 4009, NSA Type 1 encryption equipment is any NSA-certified product approved to handle classified information through Top Secret for NSS [29]. The term “Type 1” or “Type 2” both refer to the hardware equipment, also known as a HAIPE (High Assurance Internet Protocol Encryptor) device. Type 2 encryption, however, supports unclassified cryptographic equipment, assemblies, or components for sensitive but unclassified U.S. government information. The term ciphersuites or suites refer to the respective set of cryptographic algorithms supporting Type 1 and 2 devices.

Type 1 encrypting devices and Suite A encryption algorithms are classified and come with requirements for usage and protection. The NSA CSfC (Commercial Solutions for Classified) program provides Commercial Off-the-Shelf (COTS) options to overcome the challenges associated with the use of Type 1 and 2 products. The program leverages cybersecurity best practices and the layering of standardized security solutions across various Capability Packages (CPs) that provide the ability to use commercial standards to communicate securely [30].

NSA Mobile Access CP lists approved ciphersuites for the level of system protection needed for data in transit [31]. While not all inclusive, Figures 2.1 and 2.2 list NSA approved protocol ciphersuites for Transport Layer Security (TLS) and Internet Protocol Security

(IPsec) respectively.

A vital requirement of the CSfC program is double encryption for data in transit. Similarly, double encryption is common in military communications, which encrypts the data from several previously encrypted systems before being transmitted regardless of the security protocol used to protect the data. The mechanism to encrypt this data uses pre-shared keys that are updated at a specific interval. We will not discuss this additional layer of encryption based on pre-shared keys, but rather the security protocol used in the underlining C2 link for this thesis.

Security Service	Approved Algorithms	Specifications
TLS Cipher Suite	TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 or TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 or TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	FIPS PUB 180-4 FIPS PUB 186-3 FIPS PUB 197 FIPS 800-56A IETF RFC 5288 IETF RFC 5289 IETF RFC 8422 IETF RFC 8423 IETF RFC 8446 IETF RFC 8603
Authentication (Digital Signature)	RSA 3072 or ECDSA over the curve P-384 with SHA-384	FIPS PUB 186-4 IETF RFC 6239 IETF RFC 6380 IETF RFC 6460
Key Exchange	ECDH over the curve P-384 (DH Group 20) Or Diffie-Hellman 3072	NIST SP 800-56A IETF RFC 6239 IETF RFC 6379 IETF RFC 6380 IETF RFC 6460 IETF RFC 7296

Figure 2.1. This table shows the NSA approved ciphersuites for the TLS protocol and supporting documentation, adapted from [31].

Security Service	Approved Algorithms	Specifications
Confidentiality (Encryption)	AES-256	FIPS PUB 197 IETF RFC 6239 IETF RFC 6379 IETF RFC 6380 IETF RFC 6460
Authentication (Digital Signature)	RSA 3072 or ECDSA over the curve P-384 with SHA-384	FIPS PUB 186-4 IETF RFC 6239 IETF RFC 6380 IETF RFC 6460
Key Exchange/ Establishment	ECDH over the curve P-384 (DH Group 20) Or Diffie-Hellman 3072	NIST SP 800-56A IETF RFC 6239 IETF RFC 6379 IETF RFC 6380 IETF RFC 6460 IETF RFC 7296
Integrity (Hashing)	SHA-384	FIPS PUB 180-4 IETF RFC 6239 IETF RFC 6379 IETF RFC 6380 IETF RFC 6460

Figure 2.2. This table shows the NSA approved ciphersuites for the IPsec protocol and supporting documentation, adapted from [31].

It is essential to distinguish that the hardware and software components across these products are uniquely different, including the manufacturers of the technologies, which can mean that the use of disparate cryptographic technologies could negatively impact UxS interoperability. Whereas CSfC manages approved commercial vendors, NSA manages Type 1 and 2 devices and those found on the trusted integrator list. Due to operational requirements that the CSfC program cannot fulfill, NSA Type 1 and 2 products remain an integral part of military systems; despite classification restrictions and long lead times required for integration.

2.2 Chapter Summary

This chapter uncovered broad, fragmented, and lengthy approaches to UxS C2 link security that only have a general emphasis on security considerations, amidst essential strategic guidance and the underlying RMF security control and accreditation processes [23], [24]. This chapter explains that the RMF provides guidance for security; however, neither NIST nor the RMF cover the development or security considerations for the security protocols used, rather they define component algorithms and vulnerability mitigation. Chapter 3 will

describe the process of the qualitative study conducted in this thesis to define security and network resiliency development options lacking in the aforementioned procedures and guidance.

CHAPTER 3: UxS Security Study

This thesis conducts a qualitative study to gain a genuine and applicable understanding of the current state of UxS communications security, security needs, and associated cybersecurity and accreditation processes for the DOD and DON. This chapter discusses the approach to the study conducted for this thesis. The goal of this study is to obtain information from those working with the actual processes to help assess and guide viable network resiliency development options to be compared and implemented in Chapters 5 and 6.

3.1 Methodology

To assist in gaining this real-world perspective, we developed a qualitative study. This qualitative study consists of a series of questions specific to the community of interest. This section discusses the rationale behind the study format and the participant pool; and the methods used to recruit, screen, and interview the participants.

3.1.1 Study Format

The related research in Chapter 1 and the processes examined in Chapter 2 revealed a lack of specificity regarding C2 link security protocols. These chapters also showed a lack of standard requirements to support the integration of UxS within JADC2 and Project Overmatch initiatives. This absence of specificity led to the following questions: What are UxS C2 security requirements? What is the needed security feature for UxS? How can the current accreditation process be improved? How are security features integrated and selected? How are security features prioritized? Are features common to standardized security protocols important? Lastly, how are security development options interpreted and prioritized? From these questions, we developed six primary question areas for our study.

- Demographics
- UxS C2 Link Security: Questions 1–5
- Authority To Operate: Questions 6–11
- Security Criteria and Integration: Questions 12–20

- Trade-Offs: Questions 21–23
- Desired Security Features: Questions 24–27
- Security Protocol Guidance and Integration: Questions 28–32

3.1.2 Qualitative Data

We generated a series of 33 factual and opinion-oriented questions from these six primary question areas for the qualitative study, see Appendix A.2. The qualitative data from the survey is measured using a conventional content analysis where the input collected from the study participants is used to identify trends. For example, in the experience of some participants standardized security protocols are incorporated in UxS cybersecurity technology, while in others' proprietary security protocols present for the same technology. Several questions are based on a 10point Likert scale. The scale ranges from “Not Important” and “Very Inefficient and Irrelevant” at 1, to “Very Important” and “Most Relevant and Efficient” at 10. We take “majority” baselines in the following discussion to be when 80% of the total number of participant responses concur inclusive of skipped questions. If less than 80% of the participants answered a particular question, it was excluded from the study.

3.1.3 Participant Pool

The NPS Institutional Review Board (IRB) approved the protocol utilized for this study with the following restrictions. The scope of participants is limited to 30 to 75 participants. These participants can only be Department of the Navy active-duty military and civilian personnel (to exclude members of the Marine Corps). Of the participant pool, no more than nine members of the general public or government contractors are allowed. For our study, all military participants must be of equal rank or higher than the interviewers (Lieutenant/O3) to minimize coercion and undue influence of the voluntary interview. An additional IRB requirement mandated that all tracked participant Personal Identifiable Information (PII) and interview data be stored according to the protocol mandate and deleted upon completion of this research.

Recruitment

To best answer the questions of this study, we identified the community of interest participants to be subject matter experts (SMEs) in the areas of C2 links and UxS. We recruited

these participants via two methods, solicitations and referrals. These solicitations and referrals were conducted in person, virtually (Microsoft Teams and Zoom), by email or telephone. See Appendix A.1 for example recruitment email. To best reach the C2 link and UxS SMEs, we solicited participation through the following organizations and institutions: The Armed Forces Communications and Electronics Association (AFCEA), the NavalX Agility Summit supported by the Office of Naval Research, the Office of the Chief of Naval Operations N9I Unmanned Task Force, the Unmanned Vehicles and Autonomous Systems (UVAS) Working Group, and NPS faculty. This collective includes the following expertise areas related to air, surface and sub-surface platforms for UxS and C2: program managers, acquisition, and cybersecurity experts. UxS developers from the DON, robotics academia, and UxS operators. We chose this mixture of SMEs to best capture critical data points from those contributing to the many facets of the UxS development life-cycle, from concept to operation. Because these organizations and institutions comprise a wide array of UxS knowledge and skillsets, these organizations and institutions were also primary sources of referrals to participating in our study.

Screening

We opened the study to any voluntary participant who specialized within C2 links and UxS. Our recruitment efforts across the various organizations and institutions generated 75 possible interviewee candidates, full-filling the maximum number of participants per IRB protocol mandate. We grouped volunteers together from the pool of participants based on their role in C2 links and UxS for analysis. These established groups are as follows: program managers, developers, security managers, and others. We will henceforth refer to the answers provided by these groups as *data sets*. Participants who fall in the “others” data set are participants with experience as UxS operators or acquisitions. Overlaps among these areas of expertise will exist. Following the opening demographic questions, the remainder of the study consists of a series of qualitative questions that help answer the six primary question areas. To ensure the quality of captured data aligns with the overall goal of our study, we pre-screened participants based on rank, area of expertise, and branch of service affiliation based on the IRB requirements discussed in Section 3.1.3. From our initial 75 possible interviewee candidates, four candidates were filtered out based on expertise. Meaning that these candidates did not indicate background constituting resident knowledge in UxS or C2 link security. Eleven candidates were filtered out based on rank or affiliation. A further

30 participants either self-selected out based on expertise or others became unavailable to conduct an interview based on our research timelines, leaving us the minimum required participants per IRB protocol mandate.

Interview Format

Of the 30 validated interviewee participants, we individually asked each participant to provide their preferred modality to conduct the interview. We performed the interviews via Microsoft Teams, telephone, or in-person from their feedback. The interview questions were provided to the participant to review no later than 24 hours before the start of the interview. Dependent upon the participants' responses to the interview line of questioning, the interview lasted approximately 60 to 90 minutes.

At the beginning of the interview, we requested consent to record the conversation for post-facto analysis. We also restated to the interviewees that all data provided is voluntary, labeled as anonymous, stored, and deleted upon completion of our research in accordance with NPS IRB protocol mandates. Before commencing the 33 questions, we reviewed UxS and security protocol terminologies with the interviewee. We also confirmed the interviewee's role or rank within their affiliated organization and how they professionally align with the study. Additionally, during the interview, questions one, three, four, six and eight assess the participants' overall understanding of the application of security to C2 links and UxS. Lastly, we informed the interviewee that if a specific response to any of the questions added significant value to the research as a direct statement, we would contact that individual to request permission to include a quote from them in the results summary. Once the participant grants permission, per NPS IRB mandates, we must obtain written consent from the interviewee before directly quoting the participant within the research.

3.2 Study Results

This section discusses the data collected from the study and the findings from our analysis of the interviews conducted. For the complete study questionnaire containing the interview line of questions refer to Appendix A.2.

3.2.1 UxS C2 Link Security: Questions 1–5

This first set of questions aims to address UxS security requirements and the needed and desired security features for UxS. Participants were asked how advances in security protocol technologies, features and capabilities (current or future), could modernize UxS in denied and contested environments. The majority of the responses across all data sets indicated no knowledge of emerging security protocol techniques. However, the majority did reply that if a protocol could be developed, they would desire the following characteristics. 1) Low network overhead due to bandwidth limitations of Navy networks and the increasing number of UxS being deployed throughout the fleet. 2) A protocol that has low resource consumption requirements due to UxS resource limitations allows for universal implementation on most UxS regardless of size. Additionally, due to UxS size limitations and use for more sensitive operations, a UxS capable of 3) software defined protocols, which eliminates the need for encryption hardware and the manual loading of keys, ultimately increasing UxS operational use cases. As a study participant described, there would be no need to retrieve a UxS equipped with a software defined solution if it went down in enemy territory. Moreover, this increased operational use of UxS extends the devices functions to volatile and denied environments, which implies 4) the need for asynchronicity. Interviewee responses also suggested the need for a protocol that supports 5) platform-agnostic implementation (e.g., “plug-and-play”) as a desired characteristic. This trait, as described by the participants, would be device agnostic and allow for ease of implementation and increase overall interoperability.

3.2.2 ATO: Questions 6–11

When exploring ways to improve current DON accreditation processes, participants evaluated the ATO process efficiency and relevancy on the Likert scale. The majority responded with 3 to 4 out of 10. However, when asked to elaborate, most participants made the distinction that if efficiency and relevancy were split, efficiency would be 1 to 2, and relevance would be 6 to 7. The majority of the participants consider the ATO process to be inefficient and slow because it could take several years to receive approval. As an Assistant Program Manager expressed, the ATO process is unforgiving: if ATO requirements are not appropriately understood at all stages, the ATO documentation can easily get kicked back. It was also expressed that when designing similar systems, the accreditation process must be

repeated for all, leading to duplication of effort, which adds significantly more time prior to operational testing and fielding. However, the majority of the participants considered the ATO to have some relevancy as it provides a framework to address cybersecurity but lacks specificity.

Participants were also asked how standardized C2 link security protocols affect UxS ATO efficiencies. The majority of participants' responses agreed that the use of a standardized protocol would highly benefit the ATO process's speed, including baselines and re-accreditation. This is because having a standardized protocol will reduce assumptions on what to use and how to implement it. The recurring stipulation in the interviewees' responses was that a chosen standard must be vetted and approved by a recognized body. Some participants provided the IETF as an example. As an Information Systems Security Engineer indicated, if a standardized protocol is used, the Authorizing Official (AO) does not have to belabor the process by taking additional time to decipher security requirements, ultimately speeding up the process. However, a standard security protocol, in the opinion of many of the interviewees, would not solve the remaining inefficiencies of inherent in the ATO processes and possible delays from an ATO package being kicked back along the chain of approval.

Following the responses on how standardized protocols can affect ATO efficiencies, we asked participants from an accreditation perspective to compare the benefits of standardized protocols versus proprietary protocols and which should be used, in their opinion. The majority agreed that standardized protocols bring better interoperability and ease of maintenance and are vetted and approved by an international standards body. Some interviewees gave the example of TLS 1.3 as a global standard [32]. Using a standardized protocol, a developer anywhere in the world would be able to securely provide or access services online as long as they adhere to the connection standard in their software.

Across the majority of the participants, there was a consensus that proprietary protocols have a time and place, which is whenever there is no standard protocol that is capable of supporting a particular UxS design or use-case need. However, they noted that proprietary protocols could lead to vendor lock. For example, in the case of JREAP-C, a proprietary communication protocol is used to allow multiple disparate systems to inter-operate for a common operating picture. Since the introduction of JREAP-C, the DOD has been depen-

dent on its developers to maintain and supply the necessary software upgrades leading to vendor lock.

To further expound upon ATO efficiencies, participants were asked at what point ATO cybersecurity requirements should be considered. The majority indicated that they should be considered at the very beginning of the ATO process. A cybersecurity engineer emphasized that cybersecurity should be considered at the design phase, transforming the process from a compliance drill to engineering with security in mind. This response was further reinforced in that the majority of the participants also agreed that cybersecurity requirements should be considered during the design phase.

3.2.3 Security Criteria and Integration: Questions 12–20

In the next set of questions we examined how security features are integrated and evaluated. To this end, we asked interviewees how C2 link security fits within the overall operational requirements. The majority stated that C2 security is a fundamental and integral component. An Irregular Warfare expert expressed that it is paramount that operational requirements for C2 security fit in, and that they must be seriously considered when facing near-peer adversaries in modern warfare.

Regarding integration, we asked at what stage of development C2 link security options should be integrated into UxS platforms. The consensus from the group is that options should be incorporated during initial development; however, there were a few participants who were not familiar with the matter to make a comment. Per a U.S. Navy Captain, this should be done at the beginning of development and as part of the system requirements. It is not intelligent to redo system architectures to accommodate cybersecurity after the system is developed because this could lead to interoperability issues. We will extrapolate on this in the interpretation Section.

In order to integrate appropriate security, we inquired if protocol security requirements should fall under one executive agent and, if so, whom. The majority agreed that there should be a single overarching executive agent. That agent should oversee that the proper protocols and standards are being implemented as a cybersecurity best practice and enabler of interoperability. However, there was no consensus on whom this entity should be. The National Security Agency (NSA), U.S. Cyber Command, or an overarching Systems

Command were frequently named. The majority also expressed that a single oversight body would create bottlenecks in the ATO approval process, which means too many ATO requests for a single entity to manage and approve. The bottleneck issue is presented across all interviewees, from program managers to developers. Participants suggested that the bottleneck issue can be mitigated by delegating ATO approval to the lowest level possible. However, delegating down increases the risk to the Authorizing Official (AO) since they are ultimately responsible; therefore, the AO is disinclined to delegate ATO approval down to the necessary level to improve ATO turnaround significantly. Additionally, the interviewees established no common ground, i.e., system owner or architecture owner, on whom the lower-level AO ought to be.

Since the industrial-military complex plays a significant role in the development of UxS, we asked participants what the commercial sector is doing to enhance UxS C2 link security for DOD systems. The majority consensus was that the commercial sector is not doing anything for the government and are focused on their own non-DOD needs, which sometimes have the potential for DOD application. This means that the security enhancements industry is doing to solve their customers' needs are not created with DOD needs in mind; however, from time-to-time these enhancements apply to DOD use cases. For example, as mentioned by one of the participants, the IETF, in an attempt to address evolving internet security issues, matured the TLS protocol over time. The TLS protocol is also incorporated into the design of JREAP-C for security. Additionally, most participants believed that industry should be in charge of developing advanced security protocols for the DOD since they have the technical and developmental capacity that the DOD lacks and that the DOD merely needs to provide the operational requirements and funding.

Irrespective of who manages requirements or what organizations are doing to improve security, we need to explore how to best integrate security protocol requirements into UxS. Therefore we asked participants how security protocols can be integrated into UxS design, development, and acquisition. The majority of the participants expressed that to better incorporate security protocols, regardless of the UxS development stage, there is a need for a well-defined and agreed-upon standardized security protocol. As a Systems Command (SysCom) program manager explained, this standard should be platform-agnostic and consider C4I requirements instead of platform or mission-specific needs. These responses correlate with the majority of the participants' assessment that standardized protocols would

improve the efficiency of the ATO process as seen in the summary of Question 9.

Questions 12 and 14 were excluded from data collection for this section. The answers to these questions were too inconsistent to discern a trend and it was judged by the interviewers that the questions were insufficiently clear to be interpreted uniformly.

3.2.4 Trade-Offs: Questions 21–23

Due to the broad guidance found within the National Institute of Standards and Technology (NIST) guidelines and Risk Management Framework, it was necessary to understand how security is prioritized based on platform acquisition type. We first asked participants how they would prioritize between COTS and contracted platforms for UxS. The majority indicated that the priority between the two is mission dependent and that that ultimately determines the security controls used. A requirements and capabilities officer reported that if the mission is low-risk, it is sufficient to use COTS devices because such devices are cheaper to acquire. If the mission requirement is highly sensitive, then it is necessary to build a contracted platform that can be tailored to provide higher levels of security. As mission risk increases, a deeper understanding of the platform's security capabilities matched to mission requirements is required, clearly demarcated through the acquisition process.

We then asked participants when the benefits of C2 link security outweigh the cost of development and implementation. The trade-offs offered by the participants are mission-dependent because security solutions can significantly impact UxS form, fit, and function. Participants associated security with Type 1 and 2 hardware encryption devices, which is natural following the current operating norm based on the participants' experience. For example, a UxS conducting imagery intelligence will not require an NSA Type 1 encryptor, compared to the same platform conducting signals intelligence that would. As a lead SysCom UxS engineer stated, it is a trade-off between more security or less payload, especially as the UxS becomes smaller.

Finally, we inquired about the importance of C2 link security requirement standardization across all services, which means that all branches of military services should use the same common security protocol when communicating over a common link. Most participants agreed that a standard for C2 link cybersecurity that defines link interoperability is critical. A Project Overmatch expert emphasized that a common protocol across all services must

be the foundation for JADC2 since interoperability is at the core of the JADC2 initiative. One should note that a common protocol in the sense referred to by the interviewee implies interoperability across branches, and may be standardized or proprietary.

3.2.5 Desired Security Features: Questions 24–27

To assess the importance of common security protocol features in a more tangible manner, we presented an operational use case scenario A.2. For the scenario, most of the participants agreed that data protection before and after a compromise is vital, with a Likert scale ranking of 9 out of 10. They also agreed that the ability to detect a compromise occurrence in real-time is of the utmost importance ranking it 10 out of 10. The importance of asynchronicity was ranked 7 out of 10; this variance of importance was based on the participants' interpretation of UxS tactics, techniques, and procedures (TTP) that cover the loss of connectivity. For example, some stated that if a UxS loses connectivity, it should return to base (RTB) due to the safety of flight requirements making the need for asynchronicity in C2 links less important. In contrast, others indicated that asynchronicity would allow passive operation in contested environments, reducing the probability of interception or detection deeming this feature as important.

3.2.6 Security Protocol Guidance and Integration: Questions 28–32

This last section of the interview aimed to explore how security development options are interpreted and selected. Most participants reported that C2 link security integration is considered at initial design; however due to a lack of specificity it is often implemented at a later stage in development.

When asked if they use proprietary or standardized protocols in their platforms, the majority of participants answered that it depended on factors ranging from the overall cost of developing and implementing a protocol that satisfies the security requirements, to the ability to reuse their existing C2 security capabilities already developed for other platforms. Due to operational timelines and budgetary considerations, developers try to reuse what they already have in-house, saving time and money rather than creating a solution from the ground up.

To further understand the participant method of protocol selection, we asked their opinion

about the advantages and disadvantages of using standardized and proprietary protocols from a security development standpoint. From their responses, the common trend was that standardized protocols are considered highly interoperable, easy to maintain, and widely accepted. In contrast, proprietary protocols bring specific solutions for unique use cases at a higher cost.

To gain a better understanding of how much latitude participants have in the selection of a security protocol we asked participants if security protocol selection was left to the discretion of the developer. This question differs from Question 10 in Section 3.2.2 in that Question 10 focused on a management perspective and general preference for proprietary vs. standardized options, while this question focused on security of such options. Participants answered that the DOD does not provide discrete requirements for security protocols, it is most often left up to the developer's discretion. Similar to the previous response, this selection is driven by cost, convenience, and their ability to reuse existing solutions, leading to the use of a mixture of standardized and proprietary protocols.

To conclude Section 3.2.6, regardless of how the participants and affiliated organizations interpreted and selected security development options, we needed to comprehend what they preferred if given the option of discrete requirements or general guidance from the DOD on what security protocols to use. From the responses, the preference depended on the use case. On the one hand, they preferred guidance if the use case was for research and development because it allows for design flexibility. On the other hand, if the use case was for a specific operational requirement, discrete requirements were desired, such as a particular security protocol option and algorithm, i.e., Transport Layer Security with Advanced Encryption Standard-256. This is because it saves money and development time since developers do not need to guess what to implement.

3.3 Interpretation

From our interpretation of this study, we can infer that the current DON accreditation process is considered highly inefficient by the participants due to cumbersome and lengthy administrative procedures. Participants also indicated that the ATO is somewhat relevant as it provides a structure for addressing cybersecurity requirements but does not offer the desired level of detail. The desired level of detail would include specific security protocols

to use for a use case. For example, if remote connection to a system is required, then there should be a protocol specified for that use case. This lack of protocol specificity is insufficient for the program offices and system developers attempting to meet RMF compliance mandates to save time, reduce costs, and overcome resource restraints. However, making the ATO process more relevant and efficient is a twofold challenge from the view of the study participants. First, the ATO process must detail what security protocol standards to use. Secondly, the authorizing official approving the ATO needs to have an in-depth understanding of cybersecurity and be willing to delegate their authority to the lowest level possible. These two changes, in turn, allow the development of UxS to have a quicker accreditation turnaround.

From the participants' responses, no common ground on whom and at what level ATO approval authority should be delegated to; however, from the participants' responses, there seems to be two distinct levels where approval authority should exist to improve ATO efficiency. The first is with the Program Management Office of the respective system. Although risk is delegated down, the program managers and respective development teams are the system subject matter experts and in the best position to fully understand cybersecurity requirements for the protection of that system. The second is with the Systems Command, who are responsible for all respective systems, and they are in the ideal position to streamline and standardized cybersecurity processes across a wider range of systems.

Furthermore, regardless of the use case, the majority favored standardized protocols for reasons of efficiency, interoperability, and ease of maintenance, as long as these standards are vetted by legitimate entities such as the IETF. Since a standardized protocol is platform-agnostic, it shifts the focus from platform/mission-specific needs to C4I requirements improving interoperability. At the same time, participants would consider a proprietary solution at the risk of potential vendor lock for those occasions where standardized protocols could not fit the operational need.

According to the participants, the decision on which security protocols to implement needs to be addressed at the beginning of the development cycle. If it is not done in this way, the selection of the security protocol may not be based on the developed platform and appropriate security needs, but rather only the security for the use case. This leads to the possible implementation of a security protocol that is not compatible with optimal system

operation, which in turn can further affect interoperability.

Participants mentioned that industry should be in charge of developing security protocols for the DOD. This is because industry has the know-how, talent, and capacity to develop such security protocols. However, for the DOD to assure reliable access and customization to DOD use-case needs, it must invest in the development of security protocols by providing both clear use-case requirements and funding.

From our interpretation of this study, we can infer that protocol security is considered from the initial design. The protocol choice is often left up to the developer's preference and, as a consequence, interoperability is subsequently limited. However, the participants preferred discrete requirements for platforms that satisfy the specific operational need. Meaning that if the project is going to be a program of record such as the ScanEagle, specific requirements for security protocols are preferred since it will save time and money.

Lastly, this study's results yielded that the following security protocol properties and features better support the identified security needs from the participant responses for UxS C2 link security: the security of data before and after a compromise, the ability to detect a compromise occurrence in real-time, asynchronicity, scalability overhead, software-definition, and interoperability.

The security of data before and after a compromise, along with the ability to detect a compromised in real time were assessed to be of high importance by the participants. Asynchronicity was chosen because it gives the platform the ability to operate in denied environments.

Participants expressed the need for scalability overhead, meaning the need for a protocol that does not exponentially add load to the network as the number of devices increases, because there is a need to interoperate with more and more devices. The need for interoperability not only comes from the JADC2 and Project Overmatch requirements but also the participants need for a protocol that can be easily installed on any device.

The need for a software-defined security protocol comes from the issue that although NSA Type 1 devices can secure C2 links, their use on UxS platforms and sensors is frequently infeasible due to the risk associated with losing such a device. i.e., if a UxS is lost in a

foreign country with a Type 1 device, physical recovery would be required. In the case of Type 2 hardware and Suite B algorithms, due to their lower data classification protection, the usability for DOD UxS assets is also significantly limited. Regardless, if the device is Type 1 or 2, a physical encryption device affects the UxS form, fit, and function considerably to the point of affecting UxS payload. The majority of participants' explained, the use of software-defined security protocols should remove the need for a physical Type 1 or 2 encryptor device with associated retrieval requirements, while keeping the same level of system protections.

3.4 Study Conclusion

The chapter has identified the need for rapid adaptation and application of modern cryptographic protocols to current DOD and DON UxS and C2 applications and processes, particularly those desired under the JADC2 concept and Project Overmatch. The results of this study and the DOD and DON requirements outlined in Chapter 2 are compared to current and emerging security techniques in Chapter 4 to select a security protocol that best addresses the security needs of all relevant stakeholders.

CHAPTER 4: Protocol Alignment and Selection

This chapter discusses standardized and proprietary security protocols. It also investigates secure and efficient protocol options for UxS C2 link security based on the desired communications security properties and protocol capabilities discovered in Chapter 3. These security properties and capabilities in security protocol terminology translate to Forward Secrecy (FS), Post-Compromise Security (PCS), active adversary detection (AAD), asynchronicity, software-defined, and scalability with low network overhead. Forward Secrecy (FS) is the protection of data before a compromise [33]. PCS is the protection of data after a compromise, also known as “self-healing” or “future secrecy” [34]. AAD is the capability of the protocol to detect an active attack such in the case of a Man-in-the-Middle (MITM) attack. The NIST defines an active attack as “an attack on the authentication protocol where the attacker transmits data to the claimant, Credential Service Provider, verifier, or Relying Party” [35]. Software-defined signifies the virtualization of key storage, generation, and management in the form of cryptographic applications capable of running on any physical architecture. Scalability overhead refers to a protocol’s ability to support many devices (thousands) within the same group and associated network overhead. For this thesis, the growth in network overhead is annotated as e.g., $O(N^2)$ for quadratic increase and $O(\log(N))$ for logarithmic increase. This annotation is based on the required number of keys needed to be shared to maintain group communication.

These properties are compared against current and emerging security communication protocols to select the most viable for this thesis use case. We chose the protocols reviewed in this chapter because they have been widely adopted for commercial and military use or offer security properties and features not found in currently adopted protocols.

We consider both standardized protocols and emerging ratcheting protocols which offer key management for improved end-to-end encryption [36]. This is achieved by leveraging Diffie-Hellman (DH) key exchange and key derivation function (KDF). After an initial key exchange, it manages short-lived session keys’ for ongoing renewal and maintenance. These two aspects combined provide meaning to the term double ratchet. Unlike the Pretty Good

Privacy (PGP) protocol that repeatedly encrypts messages with the same public key during a session, advanced cryptographic ratcheting leverages ephemeral key exchanges for each session. This short-lived exchange thwarts an attacker's ability to decrypt new messages based on known secrets. With the creation of new session keys at a predefined interval, the attacker loses access, minimizing the threat window and achieving PCS. Derivation of keys for each new epoch also ensures past data cannot be decrypted with a compromised key for a given epoch, achieving FS. Protocols that use this method are considered self-healing. However, PCS only considers *passive* attackers. For a protocol to detect an *active* attacker, a protocol must support ratcheted key exchange with PCS and a form of per-epoch entity authentication must be introduced that binds the transcript creation over every epoch and supports communication out-of-band [34] and [37].

4.1 Standardized Protocols

Standardized protocols are designed and developed by recognized organizations such as the Institute of Electrical and Electronics Engineers (IEEE) and the Internet Engineering Task Force (IETF) or as a joint effort to meet a mutual goal. They provide a common framework for formal descriptions of software and interfaces to allow for interoperability. Standardized protocols are openly accessible and can be utilized by anyone for adoption and implementation. The IETF is the Internet standards body composed of a large, open international community of network designers, industry experts, academia, operators, and others. For example, the IETF worked closely with major players in the private sector, such as Google and Apple, during the TLS upgrade development process. The mission of IETF is to make the Internet work better by setting standards that improve the way society interacts and connect online in a manner that is efficient, safe, and private [38]. Standard-based protocols are those that are agreed upon and accepted by the community. The following standards considered for this research were introduced and vetted via the IETF.

4.1.1 OpenPGP

We selected to include OpenPGP for comparison because it is currently the most widely used standard protocol for end-to-end email encryption and is a fairly simple standard that can improve security. Meaning this protocol has high interoperability and adoption by industry. First standardized by the IETF in 1997, OpenPGP has continued to evolve [39]. The protocol

supports a diverse range of client email applications and supports NIST cybersecurity requirements [40]. OpenPGP provides data protection by using digital signatures, encryption, compression, and Radix-64 encoding conversion [39]. However, the PGP protocol repeatedly encrypts messages with the same public key during a session. Of the desired security properties and features, per [39], OpenPGP documentation supports AES256, SHA384, Diffie-Hellman, and RSA; therefore it could be combined to have a similar ciphersuite as the approved TLS ciphersuites in the NSA Mobile Access (MA) Capability Package (CP) [31]. Therefore, we categorize OpenPGP as offering the following: support for the NSA ciphersuites and has $O(N^2)$ scalability.

4.1.2 IPsec and IKEv2

Internet Protocol Security (IPsec) [41] relies on the Internet Key Exchange (IKEv2) [42] protocol for security establishment. From a security and privacy standpoint, IPsec offers data confidentiality, integrity, availability, and access control. As part of the protocol suite, IKE provides mutual authentication, a critical security property of IPsec [42]–[44]. The RFCs for IKEv1 and IKEv2 do not explicitly state the protocol security goals. However, previous analysis has assigned the goals of aliveness, (weak) agreement, and secrecy of terms. Formal analysis also considered forward secrecy, resilience to key compromise impersonation, and resilience to known session key attacks [45]. Of the desired security properties and features, per [41] and [31], IPsec offers: support for the NSA ciphersuites and has $O(N^2)$ scalability.

4.1.3 Transport Layer Security

Since its inception, the most widely used internet security standard has gone through several upgrades. TLS 1.2 is still in use today and it is the minimum standard in some use cases per the NSA [46]. However, its successor TLS 1.3, is growing in use and replacing TLS 1.2 due to its more advanced security features. The TLS 1.3 handshake protocol negotiates cryptographic keys via an authenticated key exchange (AKE) mechanism. The transport layer uses these keys for data integrity and confidentiality, among other security guarantees [32]; however, TLS still a one-to-one protocol. According to the IETF RFC for TLS 1.3, the stated security goals are the establishment of the same session keys, the secrecy of the session keys, peer authentication, the uniqueness of the session keys, downgrade protection, the FS of the long-term keys, key compromise impersonation resistance, and

the protection of the endpoint identities [32]. It is important to note that FS in TLS is per session only [47], i.e., typically subject to a longer window of compromise under FS than a ratcheting protocol, which may achieve FS per message [48]. Of the desired security properties and features, per [32] and [31], TLS offers: FS, support for the NSA ciphersuites and has $O(N^2)$ scalability.

4.1.4 Messaging Layer Security

MLS provides PCS and FS through the use of a key schedule that relies on key derivation functions (KDFs) to ratchet forward [48]. A KDF is used to generate one or more cryptographic keys from a private (secret) input value [49]. In MLS, KDFs are used to chain keys across epochs, combining fresh key material with old key material. MLS also aims to address inefficiencies in messaging groups greater than two members. MLS implements per-message encryption with $O(\log(N))$ overhead to address the exponential growth associated with messaging sessions, providing logarithmic and more efficient growth [50]. This protocol is an emerging standard built from the ground up to support communications for groups up to thousands of devices, and also supports the same ciphersuite the NSA has approved for TLS [31]. Of the desired security properties and features, per [51], MLS offers FS, PCS, asynchronicity, support for NSA ciphersuites, $O(\log(N))$ scalability, and per our categorization could offer AAD.

4.2 Proprietary-Based Protocols

Protocols within the proprietary category are designed and made for a particular use case or systems developed by a single organization. Even though these systems may use an underlying standard, these systems cannot interoperate with external systems, limiting shared usage among a community of interest. Thus, proprietary protocols limit changing the protocol design and implementation to the owner, which can enforce restrictions on the usage and lead to vendor lock. Owners usually enforce restrictions through patent rights and trade secrets, and do not disclose the technical information behind the protocol, significantly limiting DOD usage. Proprietary protocols tend to be either standard-based with proprietary changes or entirely developed in-house to address a particular niche market for commercial gain. Meaning that applications such as Signal, Wickr, and Cisco Webex leverage ratcheting protocols; however, they are proprietary to capture a particular market and use case.

Therefore, even though they may share certain functionality, they do not communicate with each other.

4.2.1 Signal

The Signal application uses ratcheting protocol techniques to provide robust end-to-end encryption for text, voice, and video. This robust encryption is achieved by leveraging Diffie-Hellman (DH) key exchange and key derivation function (KDF) [?]. After an initial key exchange, it manages short-lived session keys for ongoing renewal and maintenance. Signal has made its code open source removing security through obscurity, which has opened it up to vetting by the international body of researchers. Signal uses the double ratchet algorithm to derive new keys for every message so that an attacker cannot calculate earlier keys from compromised ones. This short-lived exchange thwarts an attacker's ability to decrypt old messages based on known secrets. Also, with the creation of new session keys at a predefined interval, the attacker loses access, minimizing the threat window. Protocols that use this method are considered self-healing. The maximum Signal group size is 1000 members per group [52]. Signal's use of Diffie-Hellman, and the protocol's adaptability to new ciphersuite combinations, means that it could be adapted to match the ciphersuites approved for TLS by the NSA [31]. Of the desired security properties and features, per [36], Signal offers FS, PCS, asynchronicity, support for NSA ciphersuites, $O(N^2)$ scalability, and per our categorization could offer AAD.

4.2.2 Wickr

Wickr, an Amazon Web Services company, offers similar security guarantees as Signal and also provide their source code for review. The main difference is that the Wickr messaging protocol uses the concept of node-to-node communication instead of user-to-user, meaning that a user can be using the Wickr app on multiple associated devices, which together create the user node [53]. One of the significant differences between Signal and Wickr protocols is their security focus and approach, and use of the double ratcheting protocol. Signal aims to provide cryptographic deniability, which creates zero evidence that participants took part in a given transaction. In contrast, Wickr looks for a more practical approach that disassociates your physical persona from your digital one [54]. Wickr's maximum group size is 300 members per group [55]. The protocol ciphersuite includes AES256, RSA4096, ECDH521,

and SHA-256 which could be combined to match the ciphersuites approved for TLS by the NSA [31]. Of the desired security properties and features, per [53], Wickr offers FS, PCS, asynchronicity, support for NSA ciphersuites, $O(N^2)$ scalability, and by our categorization could offer AAD.

4.2.3 Cisco Webex

The Webex approach aims to maintain the security and privacy of online meetings. Like Signal and Wickr, Webex provides end-to-end encryption, but it also offers end-to-end verified identity. That is, besides having a secure connection, there are guarantees that you are talking to the person or persons you believe you are talking [56]. These distinctions between the different applications of ratcheting protocol variations might seem small but are very important when looking for a suitable protocol. Regardless, the theme among these ratcheting protocols is as follows: end-to-end encryption at the application layer, so not even the provider of the services has access to the information, the use of standardized and formally-verified protocols that are device-agnostic, that aims to maximize the need for interoperability. Most importantly, Webex's current variation of ratcheting protocols utilizes MLS for its key exchange and management; therefore, Webex offers similar security properties and features as MLS.

4.2.4 Joint Range Extension Application Protocol (JREAP)

JREAP is an application layer protocol developed to reduce reliance on military service unique communication protocols during tactical data exchange. JREAP-A supports satellite communications, JREAP-B supports point-to-point connections, and JREAP-C is designed for use over IP-based networks [14]. We look at JREAP-C for this research. JREAP-C transmits data over military networks fast and reliably using both unicast and multicast modes of operation. It uses Northrop's proprietary management system with TLS as the baseline security protocol within the system to communicate over IP-based networks. Therefore, JREAP-C offers the same security properties as TLS. Northrop Grumman developed this protocol, one of the main communication methods used for tactical message delivery in modern military systems. For the past 60 years, Northrop Grumman has provided tactical data link solutions to the DOD [57]. This simple fact highlights the issue that this research attempts to address; stove-piped communication and the dependence on proprietary

protocols, which lead to vendor lock and less robust interoperability.

4.3 Protocol Selection

Table 4.1 compares the security properties and features of the standardized and proprietary protocols examined in this chapter. From our evaluation of these protocols, all of them could support or already support key NSA ciphersuites because either they contain ciphersuites matching those used in other NSA-approved protocols or the NSA has already created RFC guidance for secure deployment of the protocol. However, only Signal, Wickr, MLS, and Cisco Webex offer forward secrecy, post-compromise security, adversary detection, and asynchronicity. Only Cisco Webex and MLS offer scalability with a low overhead out of these four protocols. MLS is the base protocol of Cisco Webex and will be a standardized protocol, removing the possible issue of vendor lock. Based on our analysis, MLS offers the security properties and features needed for our use case.

Protocol	S = Standardized P = Proprietary	Forward Secrecy	Post Compromise Security	Per Epoch Authentication / Active Adversary Detection	Asynchronous	NSA Ciphersuite Support Possible	Independent of manual pre- shared keys	Scalability Overhead
OpenPGP	S	No	No	No	No	Yes	Yes	$O(N^2)$
IPsec	S	Yes*	No	No	No	Yes	Yes	$O(N^2)$
TLS	S	Yes*	No	No	No	Yes	Yes	$O(N^2)$
MLS	S	Yes	Yes	Yes**	Yes	Yes	Yes	$O(\log N)$
Signal	P	Yes	Yes	Yes**	Yes	Yes	Yes	$O(N^2)$
Wickr	P	Yes	Yes	Yes**	Yes	Yes	Yes	$O(N^2)$
Cisco Webex***	P	Yes	Yes	Yes**	Yes	Yes	Yes	$O(\log N)$
JREAP-C	P	Yes*	No	No	No	Yes	Yes	$O(N^2)$

* FS per session only

** Active adversary detection possible as an add-on feature

*** Uses MLS as a base

Figure 4.1. This comparison table depicts the different security properties and features per each reviewed protocol

4.4 Chapter Summary

This chapter investigated and compared security protocols that are a critical cybersecurity component to modern networks and the DOD and DON initiatives discussed in Chapter 2. Furthermore, this chapter analyzed each protocol's ability to provide the desired security properties and features identified in the qualitative study in 3. Our comparison finds the MLS protocol as most suited out of the selection for implementation and integration into the research use case of providing multi-domain secure data exchange and command and

control of distributed UxS over and ad-hoc wireless network. Chapter 5 will describe the process of integrating MLS into our thesis use case.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5: UxS MLS Implementation

This chapter outlines the phased development and implementation approach used to integrate the Messaging Layer Security (MLS) protocol selected from Chapter 5 within UxS platforms. It provides an overview of the use case design concept and the MLS and Robot Operating System (ROS) architectures. This chapter also covers an overview of the step-by-step methodologies used to create the MLS command and control (C2) application (MLS C2) to interface with ROS. It documents the setup, integration, and testing of the MLS application within the research use case.

5.1 MLS Design Concept

The goal of our design concept is to develop an integration application for MLS for secure and efficient C2 of distributed UxS platforms. The application is capable of providing secure exchange of data between the ScanEagle unmanned aerial vehicle and Naval Information Warfare Pacific CASSMIR unmanned surface vehicle (USV) in a multi-domain ad-hoc network configuration. This application uses the MLS library developed and maintained by Cisco at their public GitHub repository [58]. Additionally, our design approach is to create an application that is device agnostic and can be installed on any Unix-based system.

5.2 MLS Architecture

As described in Chapter 4 Section 4.1.4, MLS provides asynchronous communication and authentication for groups. Our MLS integration application leverages vital components of the MLS architecture. Figure 5.1 depicts the protocol's architecture and overall functionality, where the Authentication Service (AS) and Delivery Service (DS) are service functionalities, versus necessarily centralized servers.

their own public and private keys. User credentials created and stored by an AS offer future capabilities similar to single sign-on for user control within an organization. **Create user** relies on two key variables, the first is the selected ciphersuite for the group, and the second is the username identifier. As the name implies, the **create group** creates a group or series of groups for a user. Lastly, **group maintenance** includes adding a user to the group, updating the keys for the group, and removing a user from the group. Each of the supporting group maintenance functions is followed by the **commit** sub-function. This **commit** sub-function provides an additional layer of security for agreement on adding, updating, or removing members. This group maintenance cycle is completed by sending a **welcome** message to the newly added group member and distributing a key **update** to existing members of the group. Furthermore, MLS allows for multiple simultaneous user additions followed by a single **commit**, a single **welcome** message, and a single **update** message to the group's current members. This functionality assists with the group key exchange performance capabilities of MLS.

MLS version 12 supports the following ciphersuites:

- X25519_AES128GCM_SHA256_Ed25519 = 0x0001
- P256_AES128GCM_SHA256_P256 = 0x0002
- X25519_CHACHA20POLY1305_SHA256_Ed25519 = 0x0003
- X448_AES256GCM_SHA512_Ed448 = 0x0004
- P521_AES256GCM_SHA512_P521 = 0x0005
- X448_CHACHA20POLY1305_SHA512_Ed448 = 0x0006

Figure 5.2 is a step-by-step representation of the sequence in which each function executes for successful encrypted message exchange. As the current MLS protocol stands, these steps must be taken in sequence since the protocol does not have contingencies to deal with out-of-sequence maintenance cycle processes. For example, messages being sent or received before **add**, **remove**, or **update** operations are concluded, and a **commit** by all parties is executed will break the users' connection to the group. It is the responsibility of the DS to ensure in-sequence messages. Future MLS applications will require DS design(s) that support concurrency mechanisms to ensure proper sequencing to overcome the lack of protocol contingencies.

MLS Sequence:

1. Input user name and select ciphersuite
2. Create user
3. Create group
4. Wait to receive an ADD message
 - ADD user to group
 - COMMIT the changes
5. Send WELCOME message with the new generated group key to newly added user and send UPDATE message to N-1 parties in the group

User requesting to be added to the group makes the same steps 1 through 3 from above, then:

1. Create key package
2. Send key package information
3. Wait for WELCOME message
 - Commit WELCOME message
4. Current group members ADD and COMMIT the UPDATE message

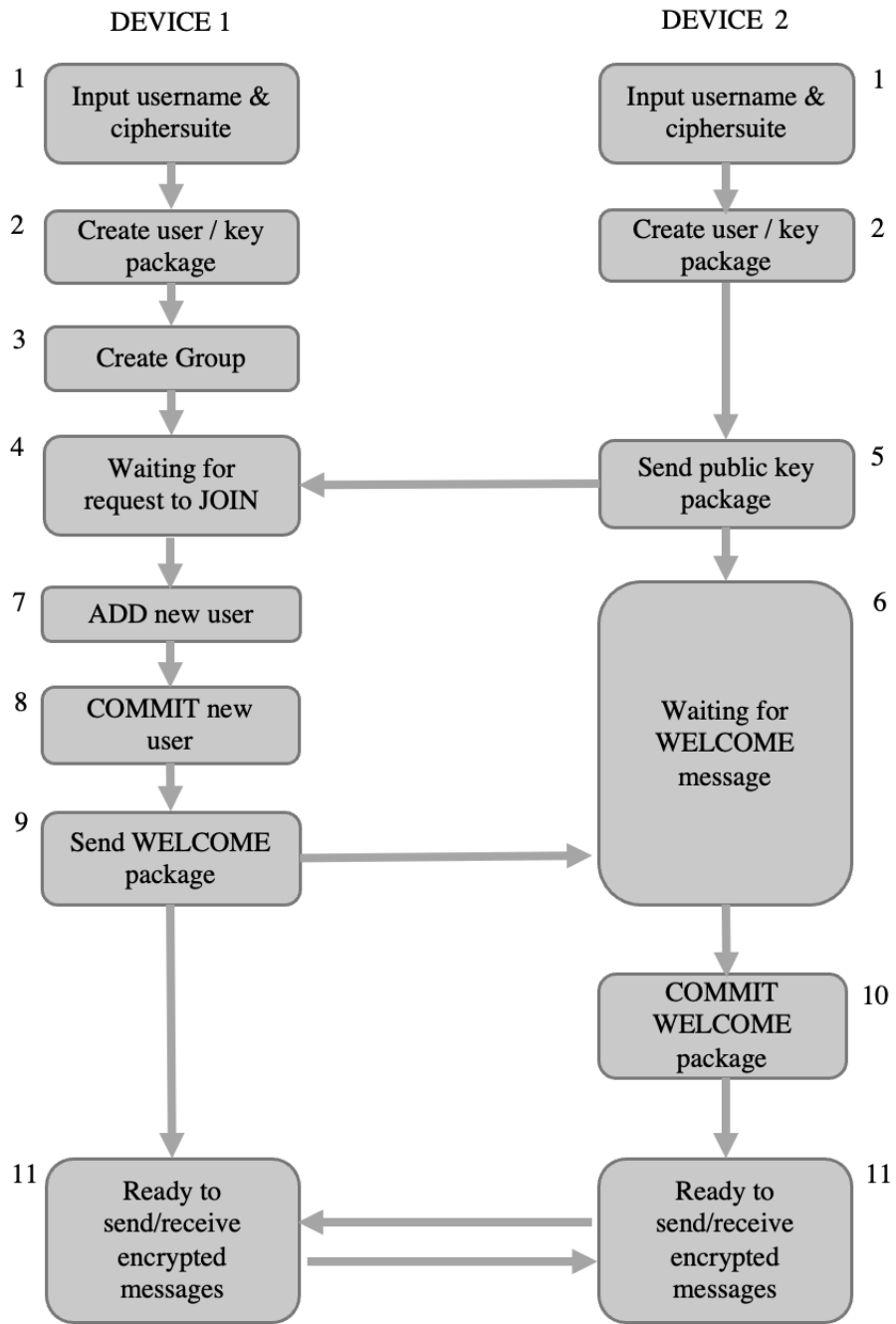


Figure 5.2. MLS Protocol Functionality Sequence

5.3 Robot Operating System (ROS)

ROS is open-source software that provides a publisher and subscriber service and the necessary libraries to develop robotic applications. ROS applications are written in C++, Python, or Lisp. ROS runs on Unix-based systems with stable testing on Mac OS and Ubuntu [59].

5.3.1 ROS Communications

ROS communication works on a publish and subscriber scheme, which has two main components, a ROS Master and ROS nodes [60]. The ROS Master is the service that helps the nodes communicate with each other. A node is a process that performs a function. All nodes must be connected to the same ROS Master to communicate. For a node to communicate, it publishes a message to a specified topic. Any node that is subscribed to that configured topic receives the message and executes a function [61]. For example, in a UAV, a “control” node publishes a new speed message to the “speed” topic, and the “motor” node, which is also subscribed to the same topic, receives the new speed message and changes the motor speed. Although the UxS platforms chosen for this use case utilize ROS, this implementation of MLS is agnostic to ROS internal functionality, leaving ROS functionality as an abstraction. The MLS application created for this use case interfaces with the existing subscribe and publish aspect of the ROS API as a ROS node for selected topic messages that will be encrypted and sent over a wireless ad-hoc network.

5.4 Use Case Methodology

We develop an application that uses the MLS protocol as a viable means to provide secure C2 for UxS platforms. We first test our application in a simulated environment and then integrate the application on two unique UxS platforms from NIWC-PAC unmanned systems division CASSMIR and the NPS CAVR ScanEagle developed by Insitu. The development of our application occurs in three sequential phases that build upon each other. These phases include Phase One, a basic chat application called MLS Chat. Phase Two is the addition of local ROS functionality to share static messages called MLS ROS. Finally Phase 3, ROS C2 applications update the MLS ROS application to directly interface with the ROS Masters of the respective UxS platforms and adds the ability to send C2 control messages to the UxS.

Chapter 7 outlines the testing and results of all MLS applications developed throughout the three phases.

Development Environment

The development environment where MLS is installed uses the Ubuntu 18.04 operating system or a later version, CMake 3.18.xx, ROS Noetic, C++ 17, and a compiled MLS library. The MLS library used for this implementation is draft C++ version 12 from the Internet Engineering Task Force (IETF) MLS Working Group GitHub repository [58]. VSCode is the integrated development environment used due to the simplistic synchronization with our GitHub repository [62]. Leading up to MLS implementation on physical UxS platforms, we employed virtual instances for application development. These virtual instances are configured to the following parameters: one core CPU and 8 GB of RAM. This configuration sets a minimum baseline for UxS processing requirements for our test, which is below the performance baseline of our use case UxS. We made no modifications to the original MLS source code, functionality, or dependencies required to facilitate our MLS applications. The following steps outline how to compile the MLS library:

1. Unzip the files
2. Access the file via terminal and run the following commands
3. `$ mkdir build`
4. `$ cd build`
5. `$ cmake ../`
6. `$ make`
7. The MLS Library is compiled and ready for use

After compiling the MLS library, all library files and include directories are manually added to our MLS application folder hierarchy. Our application's CMakeList.txt file is updated to link the necessary MLS libraries to our program. Refer to our GitHub repository [62] for all three phases and a pre-compiled MLS library.

5.4.1 Phase One: MLS chat

The development of MLS Chat in phase one allows us to understand the baseline requirements to implement the MLS functionality. The MLS Chat application was developed and run in Ubuntu virtual environments to simulate the UxS environment. We test the three pri-

mary components of the MLS protocol: create user, create group, and group maintenance. During this testing, we monitored the communication via WireShark to assess the packet contents to ensure the encryption of plaintext.

The following components are necessary for the initial application to create a secure chat session between two clients: MLS Functionality, Networking Functionality, and Message Exchange. The source code for MLS chat is available in our GitHub repository named `mls_chat` [63]. To enable MLS functionality, we added the compiled MLS library to the development root folder containing the `main.cpp` file.

MLS Functionality

To access the MLS API library needed for our implementation, we added the following MLS header files throughout our code:

- `#include <mls/credential.h>`
- `#include <mls/crypto.h>`
- `#include <mls/session.h>`
- `#include <mls/messages.h>`

These header files provide access to required MLS classes and data types: **`mls::Client`**, **`mls::Session`**, **`mls::PendingJoin`**, and **`bytes_ns::bytes`**. All members of an MLS group are initialized by the client class that contains the required user credentials for that specific user. These stored credentials are then used to create or join a group for that user through the Session class. The Session class maintains a list of all groups initialized by a user and any groups joined by the user, managed by the MLS maintenance cycle functions (ADD, REMOVE, UPDATE, and COMMIT). For members to be added to the group, a user key package is created by the PendingJoin class. Once a user is added and committed to a group, that user receives the group's WELCOME message containing all the required artifacts to communicate with the group. This information is then saved to that user's session information. Once the groups are established, members can securely communicate over the chosen network.

To effectively encrypt and decrypt data across all our MLS applications, the functions **`protect`** and **`unprotect`** from the MLS library are used. These functions required data to be in type **`bytes`**, therefore, we need to convert other data types to and from **`bytes`**. To this end, we

created a **convert** class that contains the necessary conversion functions for our application. These functions are **bytes_to_char**, **char_to_bytes**, **str_to_bytes**, and **bytes_to_str**.

Network Functionality

The MLS Chat application uses the Transmission Control Protocol (TCP) to establish server and client connections over an ad-hoc wireless network. The user creating the MLS group acts as the server for the TCP connection, in our case, the ScanEagle. Any other users trying to JOIN the group are denoted in our application as clients. To connect, every client needs to know the server's IP address. This IP address is entered at runtime. TCP was selected as the transport layer protocol because of the packet delivery guarantees inherent to the protocol. These guarantees allow us to implement MLS Chat connection in a more stable environment which mitigates packet loss that can adversely affect the sequencing of the MLS maintenance cycle [64]. For future work, a User Datagram Protocol (UDP) network can be implemented with the packet delivery guarantees at the application layer to improve the group key management performance since UDP allows for the broadcast of messages [65].

A program written in C++ uses the data of type **char*** to send data over an internet protocol (IP) network. To address these conversion needs, we used our **bytes_to_char** function to convert **bytes** to **char*** and the **char_to_bytes** function to convert **char*** to **bytes** as shown in Figure 5.3 and 5.4, respectively.

```

bytes
convert::char_to_bytes(char* array,int length)
{
    bytes data(length);
    for(int i = 0; i < length; i++)
    {
        try
        {
            data[i] = array[i];
        }
        catch (...) {
            std::cout << "Exception" << std::endl;
        }
    }
    data.erase(data.begin());
    return data;
}

```

Figure 5.3. char* to bytes conversion function

```

char *
convert::bytes_to_char(bytes data, unsigned char packetType)
{
    data.insert(data.begin(),packetType);
    char* array = (char*)malloc(data.size());
    for(int i = 0; i < data.size(); i++) {
        array[i] = data[i];
    }
    return array;
}

```

Figure 5.4. bytes to char* conversion function

Message Exchange

Following the initial group setup and the established TCP connection between server and client, the client then requests to join the group. If the client is added to the group, it receives the **welcome** message and joins the MLS group. The client is now part of the group, sends the first secure message to the server, and waits for a response. The message exchange is as follows.

The client generates a plaintext messages of type **string** which is then converted to type **bytes** via our **convert** class. As indicated above, type **bytes** is then encrypted by the MLS library **protect** function. The ciphertext output of the **protect** function is in type **bytes** and then it is converted to type **char*** to be transmitted over the wireless ad-hoc network. The receiver converts the encrypted message and converts it from type **char*** to type **bytes**, then calls the MLS library **unprotect** function to decrypt the message. Once the message is decrypted, it is converted from type **bytes** to type **string** in order for the message to be displayed. This bidirectional exchange of secure messages repeats until the server or client sends a message with the pound sign “#” to close the TCP connection. Figure 5.5 and 5.6 depicts the message exchange functionality created for both the server and client.

```

do {
    // Once it reaches here, the client can send a message first.
    //SEND 1
    std::cout << "ME: ";
    std::cin >> plaintext;

    //TURN STRING TO BYTES
    bytes plain_bytes = convert_routine.str_to_bytes(plaintext);

    //ENCRYPT MSG
    bytes enc_msg = client_session[0].protect(plain_bytes);

    //TURN BYTES TO CHAR
    char * buffer_ready = convert_routine.bytes_to_char(enc_msg, 3);

    send(new_socket, buffer_ready, enc_msg.size()+1, 0);

    if (plaintext.compare(exit_loop) != 0){
        //RECEIVE 2

        std::cout << "SERVER said: ";
        error_check = recv(new_socket, buffer, bufsize, 0);

        //CONVERT FROM CHAR to BYTES
        bytes encbyte = convert_routine.char_to_bytes(buffer, error_check);

        //DECRYPT MSG
        bytes dec_bytes = client_session[0].unprotect(encbyte);

        //TURN MSG FROM BYTES INTO STRING
        std::string dec_string = convert_routine.bytes_to_str(dec_bytes);
        plaintext = dec_string;

        std::cout << dec_string << std::endl;
    }
} while (plaintext.compare(exit_loop) != 0);

```

Figure 5.5. The client chat functionality.

The client chat functionality has two sections: sending and receiving. The sending section has the following sequence: input a message (data type string), convert the message (data

type string to bytes), encrypt the message with MLS, convert encrypted message (data type bytes to char*), send encrypted message over TCP (data type char*). The receive section commences after a message is sent. This section has the following sequence: receive encrypted message, convert the message (data type char* to bytes), decrypt the MLS message, convert the message (data type bytes to string), and finally display string. Between these two sections, the while loop checks if the message contains “#” sign. If the “#” sign is present, then the while loop stops and the program exits, else the while loop continues and prompts the user to input another message.

The server chat functionality has two sections: receiving and sending. The receive section has the following sequence: receive encrypted message, convert the message (data type char* to bytes), decrypt the MLS message, convert the message (data type bytes to string), and finally display string. The receive section commences after a message is received. The sending section has the following sequence: input a message (data type string), convert the message (data type string to bytes), encrypt the message with MLS, convert encrypted message (data type bytes to char*), send encrypted message over TCP (data type char*). Between these two sections, the while loop checks if the message contains “#” sign. If the “#” sign is present, then the while loop stops and the program exits, else the while loop continues and waits for another message.


```

do {
    std::cout << "CLIENT said: ";
    error_check = recv(new_socket, buffer, bufsize, 0);

    //TURN CHAR TO BYTES
    bytes encbyte = convert_routine.char_to_bytes(buffer, error_check);

    //TURN CHAR INTO BYTES
    bytes dec_bytes = client_session[0].unprotect(encbyte);

    //TURN BYTES INTO STRING
    std::string dec_string = convert_routine.bytes_to_str(dec_bytes);
    plaintext = dec_string;
    if (plaintext.compare(exit_loop) != 0)
    {
        std::cout << dec_string << std::endl;

        //SEND 2
        std::cout << "\nME: ";
        std::cin >> plaintext;
        //std::cin >> plaintext;

        bytes plain_bytes = convert_routine.str_to_bytes(plaintext);
        bytes enc_msg = client_session[0].protect(plain_bytes);

        //SEE THE ENCRYPTED
        //std::string enc_string2(enc_msg.begin(), enc_msg.end());
        //std::cout << "Encrypted: " << enc_string2 << std::endl;

        //TURN BYTES TO CHAR
        char * buffer_ready = convert_routine.bytes_to_char(enc_msg, 3);
        send(new_socket, buffer_ready, enc_msg.size()+1, 0);
    }
} while (plaintext.compare(exit_loop) != 0);

```

Figure 5.6. The server chat functionality.

5.4.2 Phase Two: MLS ROS

The second MLS application, named MLS ROS, is built upon all components of the MLS Chat application from phase one. MLS ROS replaces the manual generation of plaintext messages found in MLS Chat with ROS plaintext topic messages that our application subscribes to. The plaintext ROS topic messages are published by the default publisher node named **talker** found in the ROS.org tutorial [66]. By using this publisher setup, it allowed us to control the speed at which message are published, therefore, creating a controlled message delivery environment. During this phase, we install ROS Noetic [67] on two virtual environments. As in the MLS chat server and client setup, MLS ROS is installed on both virtual environments. One virtual environment simulates the user creating the MLS group, and the other, the user joining the MLS group. The default ROS installation creates a development folder named **catkin_ws/src**. We moved the MLS chat files to this new folder, renamed it MLS ROS, and modified the CMakeList.txt file to link the necessary ROS libraries.

The two MLS ROS applications send the data to the other node via TCP using MLS as the security protocol. Unlike MLS Chat, MLS ROS allows for simultaneous bidirectional message exchange via threading implementation. The application has two threads. The first thread subscribes to a predefined ROS topic, encrypts the received topic message, and transmits it over the TCP connection. The second thread remains in a constant loop listening for incoming TCP messages over the established connection. When a message is received, it decrypts the received message. The received messages are ROS topic messages. Thus all ROS data is sent via MLS in our test environment. This data exchange simulates two different UxS from different countries or services sharing information over an ad-hoc wireless network.

5.4.3 Phase Three: MLS C2

The third MLS application, named MLS C2, is built upon all components of the MLS ROS application from phase two. Phase three has two primary design objectives. The first is to exchange telemetry data between ScanEagle and CASSMIR. The second is to send control messages from a third party acting as ground control for ScanEagle. We installed MLS C2 within the ScanEagle, CASSMIR, and ground control virtual environments during this phase. At runtime, MLS C2 prompts the user to select which profile to run. The pro-

files ScanEagle, CASSMIR, or ground control. Each profile comes with specific functions enabled or disabled for overall use case functionality. The ScanEagle virtual environment creates the MLS group, and the CASSMIR and ground control virtual environments represent the second and third members of the MLS group. Figure 5.7, shows the architecture setup for the MLS C2 application.

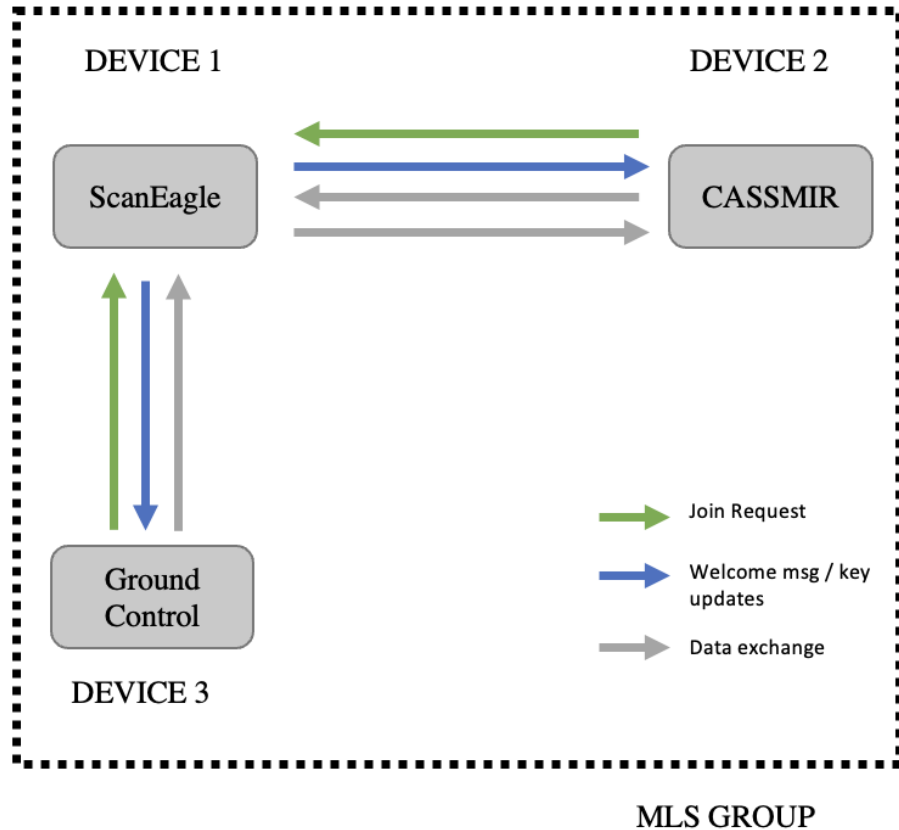


Figure 5.7. Depicts architecture used to test the MLS C2 Application

For the first objective, MLS C2 replaces the generic plaintext topic messages with actual ScanEagle and CASSMIR ROS topic data. The UxS unique topic data types of interest are the odometer data from ScanEagle and the GPS data from CASSMIR. Unlike MLS ROS, where the topic messages are of data type **string**, the odometer and GPS data are more complex data types containing multiple parameters. To handle these complex data types, we created the **odom_to_string** and **gps_to_str** functions that convert each parameter within

the unique data type to a string, and these newly concatenated strings are converted into a single string. This conversion allows MLS C2 to utilize all prior functions to encrypt, decrypt, and send and receive the ScanEagle and CASSMIR data over the IP network. Since the odometer and GPS data have been concatenated, the received message must be parsed to display the data in its original form. To address this, we created the **parse_string** function which was also added to the **convert** class.

For the second objective, MLS C2 adds the ability for the ScanEagle to receive commands from a third member acting as ground control. This functionality is accomplished by creating a third thread that allows for another TCP connection between ScanEagle and the ground control client as shown in figure 5.8. The ground station prompts a user to pass a control command to adjust the ScanEagle turn rate. The turn rate must be a number between -30 and 30. The negative numbers represent a left turn, and positive numbers represent a right turn. These control commands are sent utilizing all prior functions over the IP network. When the ScanEagle receives the control command, it also utilizes all prior functions. Once the message is converted into data type **string** it is then converted to a **float** data type and published to the **turn_rate** topic to control the ScanEagle within the simulated environment.

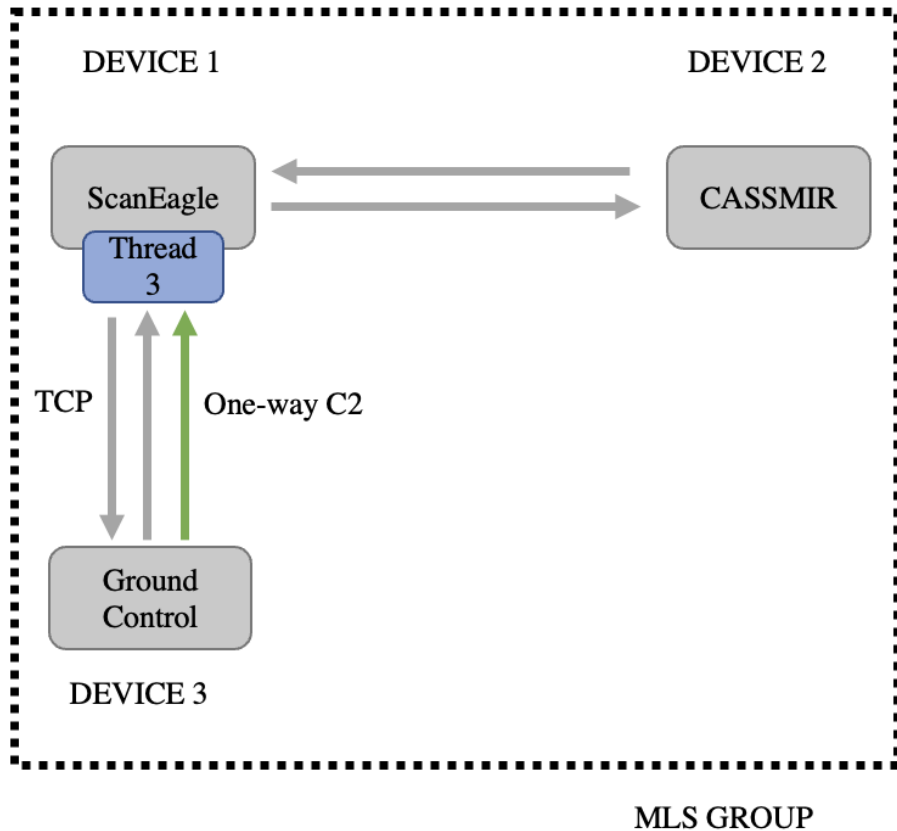


Figure 5.8. MLS C2 Application third thread design.

5.5 Chapter Summary

This chapter explained the details of MLS architecture and an overview of the ROS functionality. It also outlines the three-phase process utilized to create the MLS C2 application that integrates the MLS and ROS libraries for our secure C2 of distributed UxS platforms use case. Chapter 6 provides the testing results of the MLS applications from the three MLS C2 application development phases.

CHAPTER 6: Simulation and Results

This chapter discusses the simulation results and analysis of the MLS applications created for this research use case. As outlined in Chapter 4, these applications were developed in three distinct phases of implementation. In the first phase, simulation was conducted in a virtual environment between two host computers to provide benchmark statistics and the assurances of an errorfree MLS implementation over an 802.11 WiFi network. The second phase builds upon the first phase by replacing a single host computer with the ScanEagle platform virtual environment. The 802.11 WiFi network was replaced by using military-grade mesh network radios. This testing phase was used to assess performance in a semi-realistic communications environment and provide assurances of secure and efficient data exchange between the two devices. The final phase builds upon the prior two phases to provide command and control functionality between the ScanEagle and CASSMIR platforms. Live platform testing was also used in this phase to meet the overall research objective of providing C2 between two disparate UxS operating in the sea and air domain.

6.1 Phase One: MLS Chat

There were a total of two areas of interest identified to test and measure in phase one:

1. Encryption and Decryption of messages
2. MLS ciphersuite protocol overhead

6.1.1 Simulation Environment

The test environment for phase one consists of two logically separated Ubuntu virtual machines (VMs) connected to the same 802.11n wireless network. We install our MLS Chat application 5 in each Ubuntu instance. Once the applications were installed and compiled per 5.4, we ran our MLS Chat application on both VMs, one VM created the MLS group and the other joined the group. The message exchange configuration of MLS Chat allowed us to capture the variables of interest for this initial phase of testing.

6.1.2 Testing and Results

During this phase of testing, we ensured that MLS was implemented according to the specifications outlined in Chapter 5. We used WireShark to capture and analyze packets to confirm the encryption of plaintext. Once the implementation was verified, we compared the plaintext packet size to the ciphertext size, which corresponds to the chosen MLS ciphersuite to calculate protocol overhead.

We first sent an initial plaintext message of 15 bytes followed by the same message encrypted with each of the six ciphersuites available in MLS version 12. We also performed the same test with a plaintext message of 1000 bytes. Testing revealed that MLS encryption generated an overhead ranging from 171 to 277 bytes depending on the ciphersuite chosen, regardless of the plaintext size (15B or 1000B) 6.1. The major difference in overhead seems to be generated by the size of the cryptographic hash function used in the selected ciphersuite (SHA256 or SHA512).

Ciphersuite	Encryption size of 15 bytes of plaintext	Overhead in bytes	Encryption size of 1000 bytes of plaintext	Overhead in bytes
X25519 AES128GCM SHA256 Ed25519	186	171	1171	171
P256 AES128GCM SHA256 P256	192	177	1177	177
X25519 CHACHA20POLY1305 SHA256 Ed25519	186	171	1171	171
X448 AES256GCM SHA512 Ed448	268	253	1253	253
P521 AES256GCM SHA512 P521	292	277	1277	277
X448 CHACHA20POLY1305 SHA512 Ed448	268	253	1253	253

Figure 6.1. Table depicting the different overhead cost per ciphersuite

6.2 Phase Two: MLS ROS

There were a total of three areas of interest identified to test and measure in phase two:

1. MLS update intervals
2. Initialization benchmarks per MLS ciphersuite
3. MLS message handling metrics

6.2.1 Simulation Environment

Before actual unmanned systems (UxS) integration, the test environment for phase two consists of two virtual Ubuntu VMs installed with Robot Operating System (ROS) software. Each VM is configured to specifications discussed in Chapter 5 to simulate computing limitations common to small UxS. We then installed our MLS ROS application in each VM. Once the applications were installed and compiled per Section 5.4, we started the ROS master on each VM and the default ROS talker node modified to publish 1000 messages per second. We then ran our MLS ROS application on both VMs, one VM created the MLS group and the other joined the group. We tested the VMs connected to an 802.11n wireless network and military-grade mesh network radios separately.

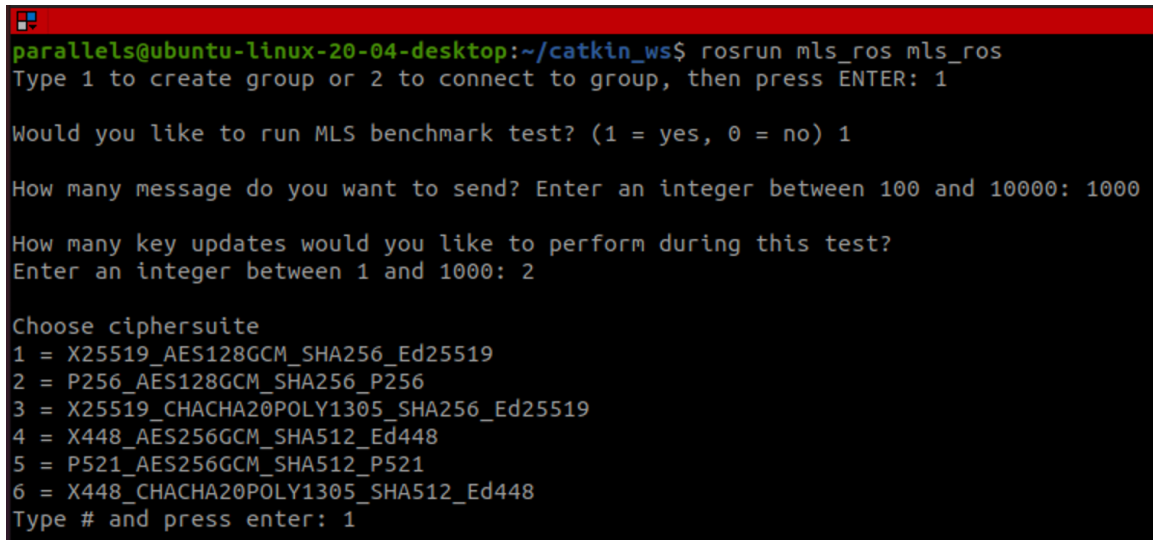
6.2.2 Testing and Results

For the initial metric of interest in this phase, we considered MLS update frequency. We varied MLS update frequency using a range of 2, 10, 50, and 300 updates per 1000 messages. At runtime, the user creating the group has the option to run a benchmark test. If “yes” is selected for testing, the user is prompted to enter the total messages to send and how many MLS key updates to perform. We entered 1,000 messages to send with 2 key updates during the message session as shown in Figure 6.2. On our first test we discovered that key updates were not being processed in sequence and therefore breaking the MLS session. We discovered that this error was due to group members receiving new messages while an **add** and **commit** of the new key update was being performed¹. To mitigate this issue we manually reduced message delivery rate of the default ROS talker.cpp file in our application. We found the optimal message transmission rate for all ciphersuites to be 100 messages per second.

After completing our initial test in this phase, we ran two benchmark tests to observe MLS protocol performance: initial group setup and message handling. We first tested and measured the time it took to set up and initialize an MLS group depending on the ciphersuite chosen; initializing a group requires the creation of user credentials and the group itself. For message handling, we tested and measured how long it took to encrypt and send 1000 messages of 15 bytes each while updating the encryption key two times during the sending

¹This issue appears to be from the absence of a Delivery Service (DS) 5.2 design inclusion into our application as discussed in the limitations section 6.4

period.



```
parallels@ubuntu-linux-20-04-desktop:~/catkin_ws$ rosrn mls_ros mls_ros
Type 1 to create group or 2 to connect to group, then press ENTER: 1

Would you like to run MLS benchmark test? (1 = yes, 0 = no) 1

How many message do you want to send? Enter an integer between 100 and 10000: 1000

How many key updates would you like to perform during this test?
Enter an integer between 1 and 1000: 2

Choose ciphersuite
1 = X25519_AES128GCM_SHA256_Ed25519
2 = P256_AES128GCM_SHA256_P256
3 = X25519_CHACHA20POLY1305_SHA256_Ed25519
4 = X448_AES256GCM_SHA512_Ed448
5 = P521_AES256GCM_SHA512_P521
6 = X448_CHACHA20POLY1305_SHA512_Ed448
Type # and press enter: 1
```

Figure 6.2. Screenshot of MLS ROS application testing prompt

We cycled through all ciphersuite options with the same parameters for baseline comparison. The results of the group initialization times per ciphersuite are shown in Figure 6.3. These tests reveal that the average setup time for five out of the six best performing ciphersuites was 5.1 milliseconds; with emph P521_AES256GCM_SHA512_P521 being the outlier performing 3.5 times slower than the average. Furthermore, the message handling results are shown Figure 6.4 indicated that following group initialization, X25519_AES128GCM_SHA256_Ed25519 and P256_AES128GCM_SHA256_P256 ciphersuites had the worst performance, although their initial group set up times were faster than average. These combined results established a connection that must be taken into consideration when implementing a particular ciphersuite. From this analysis, it is our recommendation that ciphersuites X448_AES256GCM_SHA512_Ed448 and X448_CHACHA20POLY1305_SHA512_Ed448 will provide the best balance between performance and protection. However, if there is a need for better performance, we recommend X25519_CHACHA20POLY1305_SHA256_Ed25519, which minimizes the encryption overhead at the cost of reducing the cryptographic hash functions from SHA512 to SHA256 .

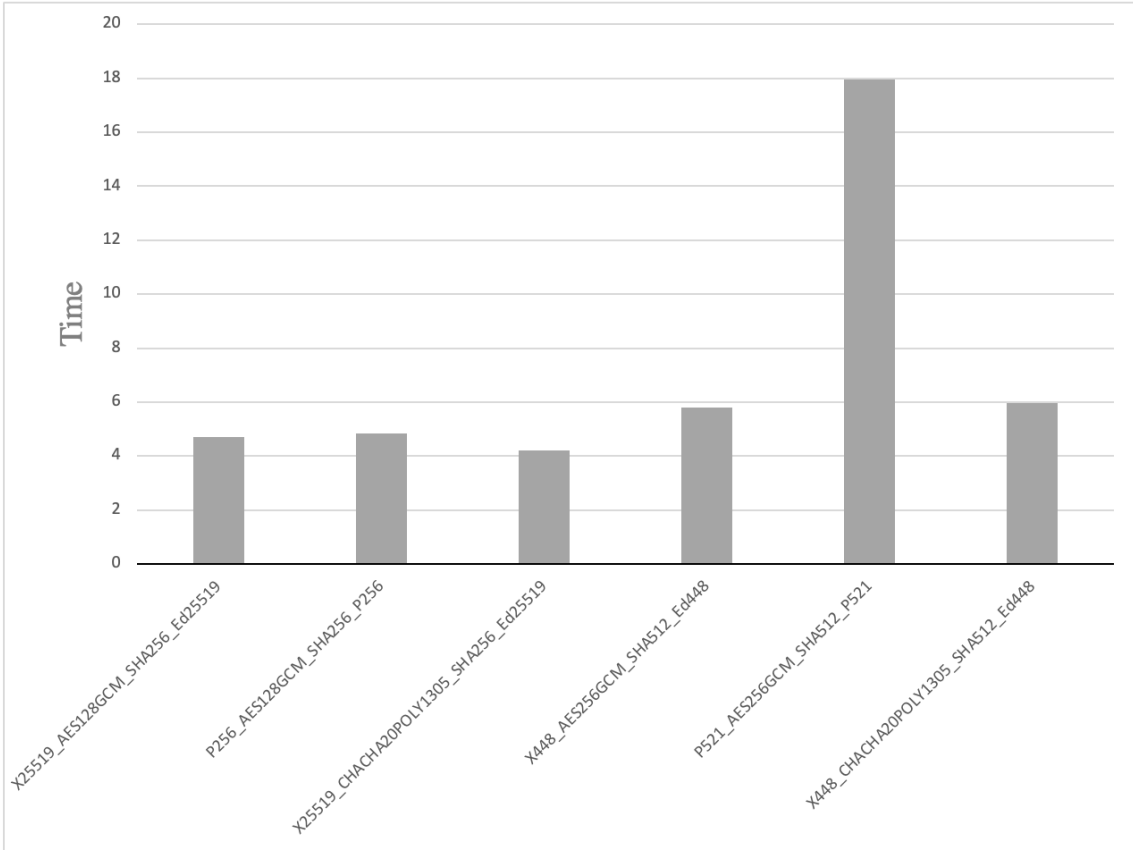


Figure 6.3. Ciphersuite setup time comparison in milliseconds

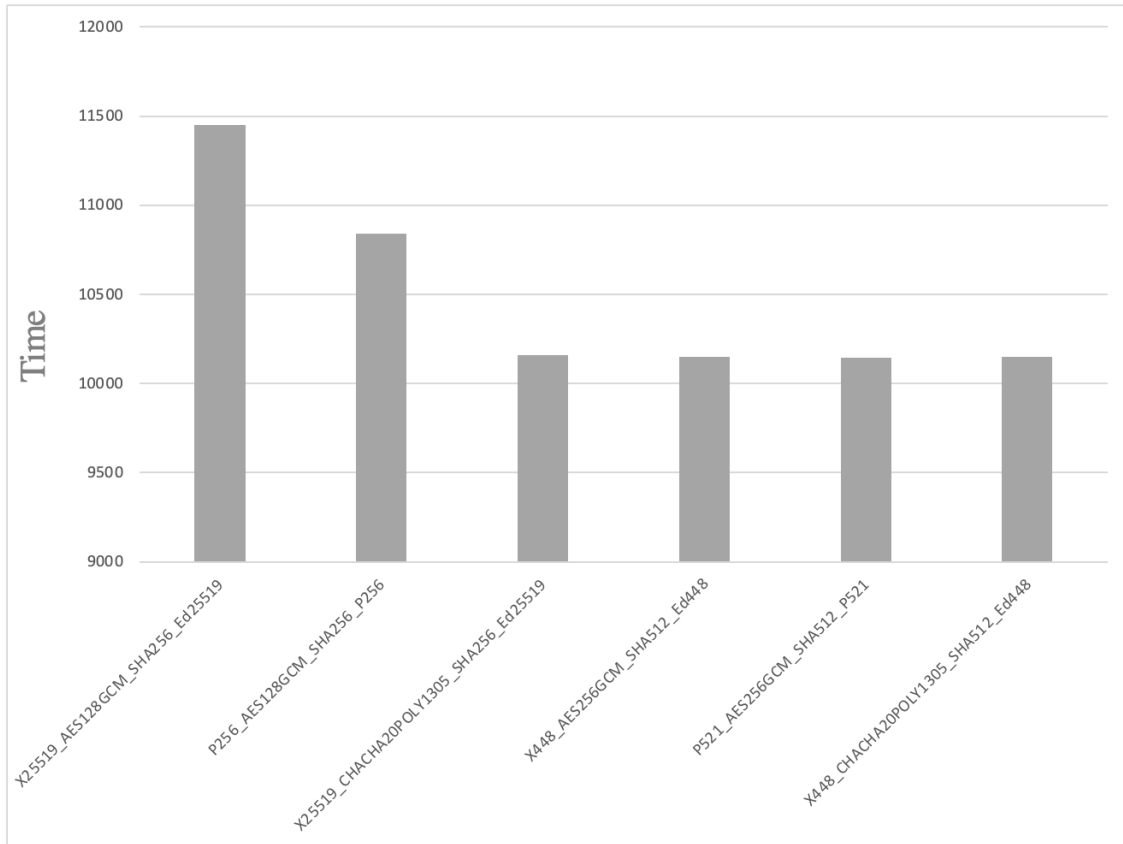


Figure 6.4. Message handling time comparison in milliseconds per 1000 messages

6.3 Phase Three: MLS C2

The section discusses the testing conducted for the proof-of-concept for this thesis use case. There were a total of three areas of interest identified:

1. Data exchange between ScanEagle and CASSMIR using MLS
2. C2 of ScanEagle using MLS
3. MLS Key Update

6.3.1 Simulation Environment

The test environment for phase three consists of A) one virtual Ubuntu VM serving as ground control for ScanEagle, B) the ScanEagle unmanned aerial vehicle (UAV) VM shown in Figure 6.5, and C) the physical Collaborative Autonomous Systems for Standoff Maritime Inspection and Response (CASSMIR) unmanned surface vehicle (USV) shown in Figure 6.6. We install our MLS C2 application in each environment. Once the applications were installed and compiled per Section 5.4, we ran our MLS C2 on all three environments in sequence, starting with the ScanEagle, then CASSMIR, and finally the ground station VM. The ScanEagle VM creates the MLS group, and all others join the group. The tested environments were connected to military-grade mesh network radios separately for physical layer transport.

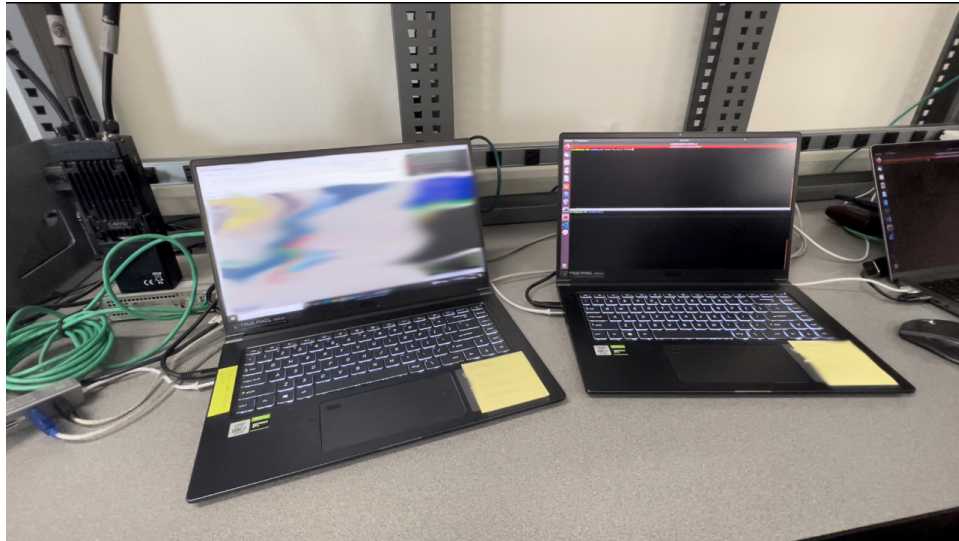


Figure 6.5. The following image is a depiction of the ScanEagle UAV virtual environment system testing setup used. The left display is blurred due sensitive platform data.



Figure 6.6. The following image is a visual depiction of the physical CASSMIR USV platform used for our thesis use case.

For the CASSMIR to join and communicate within the MLS group consisting of the ScanEagle and ground control, we added a second network adapter in the CASSMIR. This adapter was configured with a similar host IP address and network mask as the ScanEagle and the C2. Adding a dual network adapter allowed the CASSMIR to join the ad-hoc network without modifying its baseline internal network architecture.

6.3.2 Testing and Results

From the tests conducted in this final phase, the use of MLS to securely exchange data between the ScanEagle and CASSMIR was a success. The ScanEagle UAV shared its odometer data, with the CASSMIR USV which displayed the UAV positional data on the CASSMIR terminal. While the CASSMIR USV shared its GPS data, also displaying the USV positional data on the ScanEagle terminal 6.7.

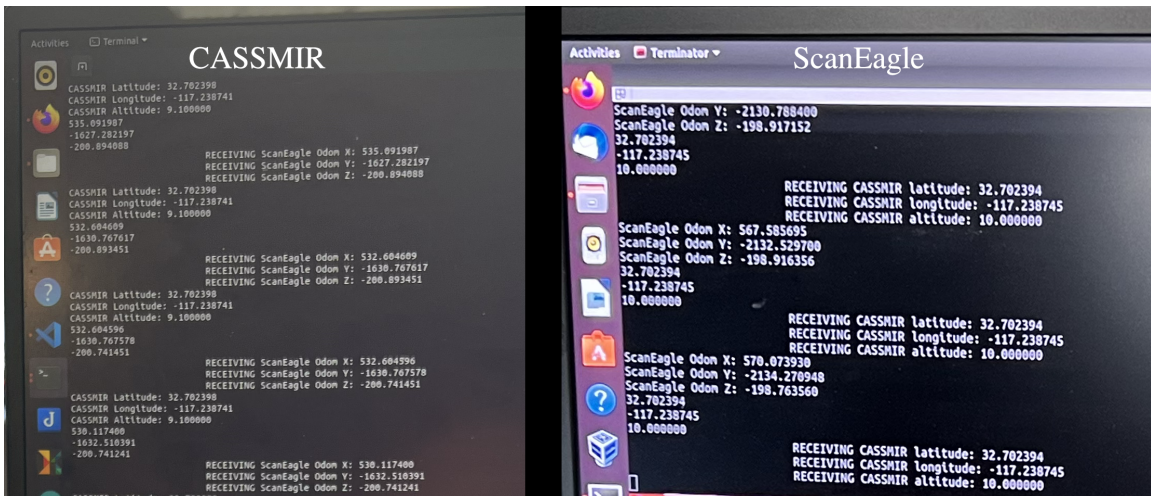


Figure 6.7. CASSMIR display (left) and ScanEagle display (right) exchanging GPS and ODOM data with MLS.

For our second metric of interest, the C2 testing of the ScanEagle was also a success. The ScanEagle received an MLS encrypted turn-rate message from the ground control VM, decrypted the message, and then published this command to the appropriate ROS topic as shown in Figure 6.8. The C2 commands were sent while simultaneously sharing data with the CASSMIR USV using MLS over an ad-hoc wireless mesh radio network.

For our final metric of interest, we repeated our second metric test and turned on the testing mode of our program to calculate setup times and the number of MLS updates sent. We attempted to conduct five MLS key updates within a single transmission of 1000 messages during this test. However, our program's lack of concurrency made it difficult for ground control to join the MLS group while updates were taking place. To solve this issue, we reduced the speed at which the application processed messages from the ScanEagle by putting the process to sleep for 1000 millisecond after every data message processed. This new setting gave us enough time to add the ground control to the MLS group before the key updates started. We were then able to successfully process 5 key updates while transmitting 1000 messages while ScanEagle and CASSMIR shared data, and the ScanEagle received C2 from the ground station. However, it is important to note that if we had implemented a DS or concurrency controls (data processing sequence scheme) there would not have been a need to put the process to sleep.

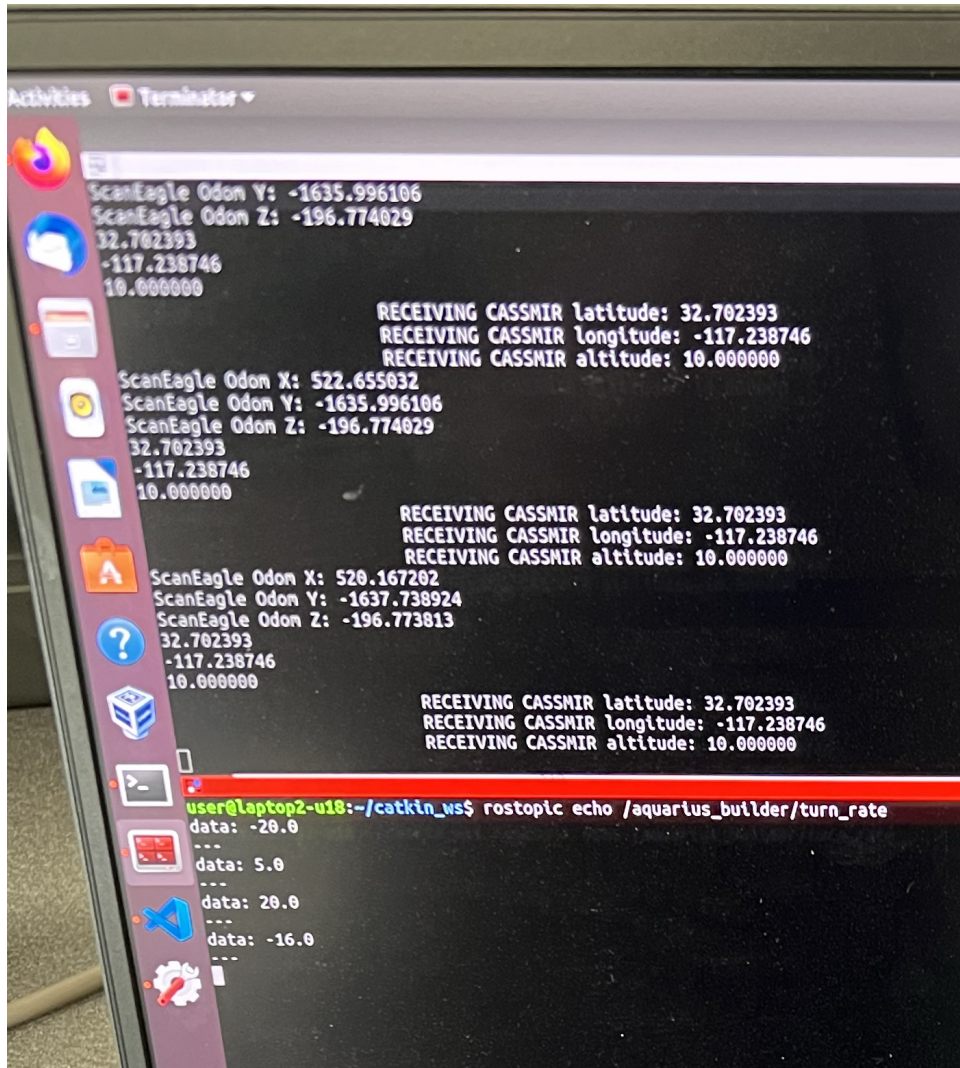


Figure 6.8. ScanEagle display exchanging data from CASSMIR and receiving C2 commands from ground station with MLS.

6.4 Limitations

The limitations covered in this section address restrictions in the thesis testing performed, where future work may extend. We observed the following limitations while implementing MLS C2: MLS group size larger than three, processing of MLS updates.

Based on our MLS application design approach, our tests can only manage three users

in an MLS group. We opted to use application threading to manage MLS group users for development simplicity. Therefore, we forfeited a more complex design that can dynamically set up, update, and track multiple TCP connections.

Per the error discovered in phase two testing, our application does not have the necessary exception handling and concurrency controls to manage out of sequence MLS maintenance messages. Therefore, our application is susceptible to the message receive rate. The use of a DS in combination with semaphores and locks can overcome this implementation limitation by blocking the MLS group members from sending new messages until all group members have processed the new group key [68].

6.5 Chapter Summary

This chapter covered the testing methods and results for integration of MLS into the ScanEagle and CASSMIR platforms. Testing of our MLS C2 and MLS ROS applications in the virtual environment provides evidence that the use of MLS can support encryption and decryption requirements to exchange data between disparate UxS systems operating in multiple domains. Tests of our MLS C2 application on a combination of virtual and physical devices demonstrates that MLS can serve as a multi-device security protocol that optimizes interoperability, agnostic to IP network and platform type in a multi-domain ad-hoc network configuration.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 7: Conclusion and Future Work

This chapter highlights our key findings in both the qualitative study and implementation of MLS and provides recommendations for future work.

7.1 Conclusion

This research examined aspects of the challenges UxS stakeholders face across the DOD and DON. It also outlined our implementation of the MLS library version 12 developed by Cisco for UxS within virtual and physical environments. Our research identified challenges in DOD and DON procedures, and gaps in UxS strategies that point to a lack of security protocol specificity to enhance the integration of UxS in JADC2 and Project Overmatch. The results from our qualitative study revealed key protocol security properties and features needed for DOD and DON UxS. Viable security protocol options were then matched to these needs and analyzed. We find MLS particularly suited to address these needs from our analysis and selected it for our research use case. Our use case testing results show that the MLS protocol is a viable solution for our multi-domain proof of concept for secure data exchange and C2 of distributed and dissimilar UxS agnostic to IP network architecture.

Achieving these results is a significant step toward UxS security and interoperability enhancements needed to address DOD and DON UxS shortfalls.

7.2 Future Work

There are a number of possibilities for future work based on the qualitative study in Chapter 3 and the MLS C2 application in Chapter 5.

7.2.1 Qualitative Study

The NPS IRB restricted the participant pool used to collect data for the qualitative study conducted in this thesis to DON military and DON civilian personnel, as well as a limited number of DON contractor and industry participants. Future work should consider utilizing

a similar qualitative study across all service branches within the DOD, with a much higher number of DOD contractor and industry participants. It should also consider conducting the qualitative study with allies and partners in the realm of UxS for a much broader understanding of strategic and procedural challenges for UxS. Future studies that examine cross-service and international real-world UxS experience will diversify the overall participant pool. This new aggregate can yield further understanding of the C2 link security protocol properties and features required for UxS security and interoperability in this modern era.

7.2.2 Additional Testing

To further examine MLS' effect within distributed multi-domain UxS environments requires more extensive testing. A small number of controlled tests supported our research, involving tailored simulations to exchange data and conduct command and control (C2) actions on virtual and physical UxS for a singular proof of concept. Even though, MLS by design is a group protocol, a future evaluation of MLS should include testing beyond a controlled lab environment modeled after real-world DOD and DON UxS tactics with more than three members in an MLS group.

7.2.3 Authentication Service

The MLS applications developed for this thesis use case concentrated on technical aspects of our use case for UxS within an ad-hoc wireless network, and did not take into consideration mechanisms to authenticate MLS group membership. The MLS C2 application developed for our proof of concept does not verify members joining the MLS group. Future work in this area should include the development of either a centralized or decentralized authentication service, pursuant to the MLS standard. Developer assessment and alignment to the threat models respective to the UxS mission requirements should be conducted before employment of MLS on UxS.

7.2.4 Delivery Service

Our MLS C2 application does not have built-in concurrency controls for message handling. Therefore, if messages are received out of sequence, they will prevent additional members from joining the MLS group or breaks the MLS session. Future work should include developing a centralized or decentralized delivery service that can manage MLS members

and messages within each group, pursuant to the MLS standard. As a result, this added service would provide the robustness needed to handle MLS group updates and members effectively.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX

A.1 Recruitment Sample Email

This recruitment email was used to request voluntary participation of subject matter experts for the qualitative study interview.

Calling for unmanned systems or cybersecurity experts to assist in a study relating to JADC2 and Project Overmatch needs at Naval Post Graduate School (NPS). This study will help in gaining a deeper understanding of the current state of communications security of unmanned systems and sensors, and associated cybersecurity and accreditation processes for the Department of Defense and the Department of the Navy.

Your insight is needed and appreciated in assessing and guiding security and network resiliency development options for autonomous devices and sensors – aggregating a vast spectrum of real-world experience!

The following CHIPS article (<https://www.doncio.navy.mil/chips/ArticleDetails.aspx?ID=15380>) and attached documents provide further context into the area being examined.

Who: Those with experience in security, autonomous device and sensor networking, acquisition, or overlaps among these What: A 60 to 90 min interview consisting of 33 questions (Sent NLT 24 hours in advance) When: At your convenience (We are on your time) Where: MS Teams / Telcon / Fact-to-Face (Your Choice) Why: To assess and guide security, and network resiliency development options for autonomous devices and sensors

For more information or to participate in this study please send an email to christopher.britt@nps.edu and andre.leon@nps.edu.

Thank you for your time and consideration.

A.2 Interview Questionnaire

These qualitative study questions formed the foundation of the case study research conducted in Chapter 3 of this thesis. The data collected from a wide array of subject matter experts assisted in the recommended protocol adaptations also provided in this research.

Privacy Act Statement

Authority to request this information is granted under 5 U.S.C. 301, Departmental Regulations; 10 U.S.C. 5031 and 5032. License to administer this survey is granted per OPNAVINST 5300.8C under OPNAV Report Control Symbol: NSP13011 which expires 01/06/2024. This survey has received IRB Approval.

Line of questions

We request your input on the following questions. Some questions may not be relevant to all respondents; please answer all relevant questions, including ‘No’ answers or “I do not know,” where appropriate.

Reference definitions / interpretations as used in this survey.

- ‘Security protocol’ refers to the cryptologic protocol comprised of algorithmic components leading to cyber security protections on C2 links. Examples of standardized security protocols may include MLS, TLS, QUIC, etc.
- ‘COTS’: Commercial of the Shelf
- Security Criteria and Integration
- ‘UxS’: Unmanned Systems
- ‘Asynchronous protocol’ refers to a communication between two or more parties that does not require for them to be authenticated and vetted in real-time allowing for one party to send and the appropriate parties to receive the information when available.

Demographics:

What is your role/rank within organization?

How is your role linked to UxS/cybersecurity?

UxS C2 Link Security

1. To your knowledge, what are the Navy's UxS C2 security protocol requirements for designated UxS platforms, if any?
2. How can or should advances in security protocol technologies be used to modernize and streamline UxS security techniques to account for
 - a. denied environments?
 - b. potential compromise during mission (due to contested environments)?
3. Are you aware of the distinction between security algorithms and protocols for C2 links?
4. Are you aware that neither NIST nor the RMF cover the definitions or security considerations for the security protocols used, instead defining component algorithms and vulnerability mitigations?
5. What new cybersecurity C2 link protocol features or capabilities would be ideal for enabling current or future UxS use if they could be developed? Please elaborate your answer.

ATO

6. What organization or person holds the authority to dictate Authority to Operate (ATO) requirements and approval for UxS platforms?

Please answer the following question on a scale from 1 to 10, where 1 = Very Inefficient and Irrelevant and 10 = Most Relevant and Efficient.

7. How do you feel the current ATO process for UxS platforms in terms of efficiency and relevancy is considering modern-day cyber threats and vulnerabilities?
8. Are standardized (vs. proprietary) security protocols currently incorporated in UxS cyber security technology?
9. How would the use of C2 link security protocols that are standardized (in domestic or international standards) effect efficiency of the UxS ATO process?
10. In your opinion, how does use of standardized security protocols compare to proprietary

protocols? Which one should be used for UxS and why?

11. In your opinion, at what point in the UxS development process should ATO cyber security requirements for certification be considered?

Security Criteria and Integration

12. What documents or instructions – formal or informal – are UxS authentication and security requirements derived from?

13. How do C2 link cybersecurity requirements fit within the overall operational requirements?

14. What are the evaluation criteria and processes to approve the security of a UxS platform for military use?

15. In your opinion, should C2 link security choices be integrated into UxS platforms before, during, or after development? What is the reasoning for this?

16. In your opinion, should protocol security requirements fall under one executive agent who controls and manages all cyber security requirements for UxS across the DON and DoD? Please elaborate on your answer.

17. Who in your opinion should be responsible for keeping C2 link security protocols current and in accordance with operational requirements?

18. What are third parties (e.g: contractors or industry) doing to enhance C2 link cybersecurity requirements for DoD UxS platforms?

19. Should third party, contractor, or commercial entities be responsible for developing cybersecurity for DoD UxS fleets?

20. How can security protocols for UxS platforms be integrated into the design, development, and acquisition process? Please elaborate your answer.

Trade-offs

21. Given the options of COTS use (and accompanying limited security requirements)

vs. contracted platforms (with accompanying controllable security and interoperability procurement requirements), how would you prioritize?

22. When do benefits of C2 link security outweigh the cost of development and implementation?

23. How important is it for C2 link cybersecurity requirements (which define communication link interoperability) to be standard across all services?

Desired Security Features

Imagine an ISR UAV flying in a conflict zone. The UAV is recording video and conducting SIGINT activities. This information is transmitted over a C2 link for the appropriate stakeholders to view and analyze for real-time decision making. While this is taking place, a nation-state adversary is also collecting on the transmission, i.e. the encrypted C2 link aiming to conduct a full take-over. Suppose that a cyberattack is eventually successful, giving data access to the adversary at the exact moment of the attack.

Please answer the following questions on a scale from 1 to 10, where 1 = Not Important at All and 10 = Very Important.

24. How important is the protection of the data that was sent on the UxS C2 link before the moment of cyberattack?

25. How important is the protection of the data that is sent on the UxS C2 link after the moment of a cyberattack?

26. How important is it to be able to actively detect a cyberattack occurring in real time?

27. How important are UxS C2 link protocols that function in denied environments without requiring consistent connectivity? (I.e., that support asynchronous communication.)

Security protocol guidance and integration

28. In UxS development, is C2 link security protocol integration part of initial design, incorporated in later stages, or not at all? How does this relate to your company's business plan and strategy?

29. If integrating security protocols into your products, are these protocols proprietary or standards based? Can you give an example?

30. What are the advantages and disadvantages of using standardized and proprietary security protocols from your view?

31. Is there discrete requirements or guidance given by DoD on what security protocol to use for UxS development or is this left to the discretion of the developer?

32. Would you prefer to have a specific security protocol requirement to follow, or guidance on what will be accepted on devices used in the DoD? Why or why not?

Conclusion

33. Do you have any other comments, questions, or insights you would like to share regarding UxS security and in particular the security of C2 links?

A.3 Repository Links

The GitHub repository containing the source code for MLS Chat, MLS ROS, and MLS C2 can be found at <https://github.com/brosito/> [62].

List of References

- [1] E. Gamboa, "SECNAV meets with project overmatch experts; discusses way ahead for connected future fleet," United States Navy, Apr. 26, 2021 [Online]. Available: <https://www.navy.mil/Press-Office/News-Stories/Article/2584424/secnav-meets-with-project-overmatch-experts-discusses-way-ahead-for-connected-f/>
- [2] J. R. Hoehn, "Joint all-domain command and control (JADC2)," CRS Report No. IF11493, Washington, DC: USA, 2022 [Online]. Available: <https://sgp.fas.org/crs/natsec/IF11493.pdf>
- [3] *Department of the Navy Unmanned Campaign Framework*, Unmanned Campaign Plan, U.S. Dept. of the Navy, Washington, DC, USA, 2021 [Online]. Available: https://www.navy.mil/Portals/1/Strategic/20210315%20Unmanned%20Campaign_Final_LowRes.pdf?ver=LtCZ-BPIWki6vCBTdgtDMA%3D%3D
- [4] *Department of the Navy Science and Technology Strategy for Intelligent Autonomous Systems*, Intelligent Autonomous Systems, U.S. Dept. of the Navy, Washington, DC, USA, 2021 [Online]. Available: <https://www.onr.navy.mil/-/media/Files/Our-Research/Naval-IAS-Strategy.ashx>
- [5] NSA, "Mission combat support," Accessed Apr. 18, 2022 [Online]. Available: <https://www.nsa.gov/About/Mission-Combat-Support/>
- [6] R. B. Thompson and P. Thulasiraman, "Confidential and authenticated communications in a large fixed-wing UAV swarm," *IEEE Explore*, Oct. , 2016 [Online]. doi: <https://doi.org/10.1109/NCA.2016.7778644>. Available: <http://ieeexplore.ieee.org/document/7778644/>
- [7] M. Ozmen and A. Yavuz, "Dronecrypt - an efficient cryptographic framework for small aerial drones," *IEEE Explore*, Jan. 3, 2019 [Online]. doi: <https://doi.org/10.1109/MILCOM.2018.8599784>. Available: <https://ieeexplore.ieee.org/document/8599784>
- [8] G. Samid, "Drone targeted cryptography," *International Association for Cryptologic Research*, 2016 [Online]. doi: https://doi.org/10.1007/978-3-642-23822-2_18. Available: <https://eprint.iacr.org/2016/499.pdf>
- [9] B. Dieber, S. T. Benjamin Breiling, S. Kacianka, S. Rass, and P. Schartner, "Security for the robot operating system," *Elsevier*, vol. 98, Dec., 2017 [Online]. doi: <https://doi.org/10.1016/j.robot.2017.09.017>. Available: <https://doi.org/10.1016/j.robot.2017.09.017>

- [10] F. Xiong, A. Li, H. Wang, and L. Tang, "An SDN-MQTT based communication system for battlefield UAV swarms," *IEEE Communications Magazine*, vol. 57, no. 8, Apr. 2019 [Online]. doi: <https://doi.org/10.1109/MCOM.2019.1900291>. Available: <https://www.ieeeexplore.ieee.org/document/8808160/>
- [11] X. Chen, J. Tang, and S. Lao, "Review of unmanned aerial vehicle swarm communication architectures and routing protocols," *Applied Sciences*, vol. 10, no. 10, May 25, 2020 [Online]. doi: <https://www.mdpi.com/2076-3417/10/10/3661>. Available: <https://www.mdpi.com/2076-3417/10/10/3661>
- [12] J.-P. Yaacoub, H. Noura, O. Salman, and A. Chehab, "Security analysis of drones systems: Attacks, limitations, and recommendations," *Internet of Things*, vol. 11, Sep. , 2020. Available: <https://doi.org/10.1016/j.iot.2020.100218>
- [13] Insitu, "Demonstrating true interoperability with RQ-21A blackjack." Available: <https://www.insitu.com/newsroom/story/interoperability-with-rq-21a>
- [14] Northrop-Grumman, "Tactical data link explained," pp. 148–167, 2014. Available: <https://dl.icdst.org/pdfs/files/e90d37a9b93e2e607206320ea07d7ad2.pdf>
- [15] J. R. Hoehn, "Joint all-domain command and control: Background and issues for Congress," CRS Report No. R46725, Washington, DC: USA, 2022 [Online]. Available: <https://sgp.fas.org/crs/natsec/R46725.pdf>
- [16] NAVWAR, "Navy continues revitalization momentum for NAVWAR facilities in San Diego," U.S. Navy, Mar 11, 2021 [Online]. Available: <https://www.navy.mil/DesktopModules/ArticleCS/Print.aspx?PortalId=1&ModuleId=523&Article=2533500>
- [17] DOD, *Fact Sheet: 2022 National Defense Strategy*, NDS, 2022 [Online]. Available: <https://media.defense.gov/2022/Mar/28/2002964702/-1/-1/1/NDS-FACT-SHEET.PDF>
- [18] *NIST Risk Management Framework*, NIST SP800-37, U.S. Department of Commerce, Washington, DC, USA, 2018 [Online], pp. 1–183. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-37r2.pdf>
- [19] USN, "CNO NAVPLAN 2021," CNO NAV Plan, U.S. Dept. of the Navy, Washington, DC, USA, 2021 [Online]. Available: <https://media.defense.gov/2021/Jan/11/2002562551/-1/-1/1/CNO%20NAVPLAN%202021%20-%20FINAL.PDF>
- [20] E. Gamboa, "NAVWAR meets milestone transition to risk management framework; achieves improved cyber readiness navy-wide," United States Navy, Mar. 1, 2021 [Online]. Available: <https://www.navy.mil/Press-Office/News-Stories/Article/>

2520059/navwar-meets-milestone-transition-to-risk-management-framework-achieves-improve/

- [21] NIST, “NIST timeline,” Accessed Mar. 6, 2022 [Online]. Available: <https://www.nist.gov/timeline#event-774416>
- [22] NIST, “Federal information security modernization act (FISMA) background,” Accessed Apr. 14, 2022 [Online]. Available: <https://csrc.nist.gov/Projects/risk-management/fisma-background>
- [23] *Control Baselines for Information Systems and Organizations*, NIST SP800-53B, U.S. Department of Commerce, Washington, DC, USA, 2020 [Online], pp. 1–85. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53B.pdf>
- [24] *Risk Management Framework for Information Systems and Organizations*, NIST SP800-37r2, U.S. Department of Commerce, Washington, DC, USA, 2018 [Online], pp. 1–183. Available: <https://doi.org/10.6028/NIST.SP.800-37r2>
- [25] *Standards for Security Categorization of Federal Information and Information Systems*, FIPS Publication 199, U.S. Department of Commerce, Gaithersburg, MD, USA, 2004 [Online], pp. 1–13. Available: <https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.199.pdf>
- [26] *Minimum Security Requirements for Federal Information and Information Systems*, FIPS Publication 200, U.S. Department of Commerce, Gaithersburg, MD, USA, 2006 [Online], pp. 1–17. Available: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.200.pdf>
- [27] *National Security Directive 42*, NSD-42, The White House, Washington, DC, USA, 1990 [Online]. Available: https://www.nsa.gov/Portals/75/nsd42_pdf.pdf
- [28] *Cryptographic Standards and Guidelines Development Process*, NIST, Visiting Committee on Advanced Technology, 2014 [Online]. Available: <https://www.nist.gov/system/files/documents/2017/05/09/VCAT-Report-on-NIST-Cryptographic-Standards-and-Guidelines-Process.pdf>
- [29] *Committee on National Security Systems (CNSS) Glossary*, (CNSS) Glossary, Committee on National Security Systems, Washington, DC, USA, Apr. 6, 2015 [Online]. Available: <https://rmf.org/wp-content/uploads/2017/10/CNSSI-4009.pdf>
- [30] *What is Commercial Solutions for Classified (CSfC)?*, [NSA], Washington, DC, USA, 2021 [Online], pp. 1–11. Available: [https://www.nsa.gov/Portals/75/documents/resources/everyone/csfc/Coming%20Soon/CSfC%20PMO%](https://www.nsa.gov/Portals/75/documents/resources/everyone/csfc/Coming%20Soon/CSfC%20PMO%20)

20Customer%20Handbook_07222021.pdf?ver=eESCodGD30fGBF13mvZkDg%3d%3d

- [31] *Commercial Solutions for Classified (CSfC)*, NSA/CSS, NSA, Washington, DC, USA, 2022 [Online]. Available: [https://www.nsa.gov/Portals/75/documents/resources/everyone/csfc/capability-packages/\(U\)%20MA%20CP%202_5_1.pdf?ver=9cS-6Gc5aDkg8IFi9zXDeQ%3d%3d](https://www.nsa.gov/Portals/75/documents/resources/everyone/csfc/capability-packages/(U)%20MA%20CP%202_5_1.pdf?ver=9cS-6Gc5aDkg8IFi9zXDeQ%3d%3d)
- [32] E. Rescola, *The Transport Layer Security (TLS) Protocol Version 1.3*, Aug., 2018 [Online]. Available: <https://www.rfc-editor.org/rfc/pdf/rfc/rfc8446.txt.pdf>
- [33] C. Boyd, A. Mathuria, and D. Stebila, *Protocols for Authentication and Key Establishment* (Information Security and Cryptography). Berlin, Heidelberg: Springer Berlin / Heidelberg, 2019.
- [34] B. Dowling and B. Hale, “Secure messaging authentication against active man-in-the-middle attacks,” 2021 IEEE European Symposium on Security and Privacy (EuroSP), pp. 54–70, 2021.
- [35] *Digital Identity Guidelines*, NIST SP800-63-3, U.S. Department of Commerce, Washington, DC, USA, 2017 [Online], p. 39. Available: <https://doi.org/10.6028/NIST.SP.800-63-3>
- [36] T. Perrin and M. Marlinspike, *The Double Ratchet Algorithm*, Nov. 20, 2016 [Online]. Available: <https://signal.org/docs/specifications/doubleratchet/doubleratchet.pdf>
- [37] B. Hale, “Authenticated Continuous Key Agreement: Active MitM Detection and Prevention,” Private Communication, May 2022.
- [38] IETF, “The internet engineering task force (IETF) - about,” 2022 [Online]. Available: <https://www.ietf.org/about/>
- [39] J. Calls, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer, *OpenPGP Message Format*, Nov., 2017 [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc4880>
- [40] *Recommendation for Key Management*, NIST SP800-57, U.S. Department of Commerce, Washington, DC, USA, 2020 [Online], pp. 1–85. Available: <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r5.pdf>
- [41] S. Frankel and S. Krishnan, *IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap*, 2011 [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc6071>

- [42] C. Kaufman, *Internet Key Exchange (IKEv2) Protocol*, 2005 [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc4306>
- [43] R. Atkinson, *Security Architecture for the Internet Protocol*, Aug., 1995 [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc1825>
- [44] P. Karn, P. Metzger, and W. Simpson, *The ESP DES-CBC Transform*, Aug., 1995 [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc1829>
- [45] C. Cremers, “Key exchange in IPsec revisited: Formal analysis of IKEv1 and IKEv2,” *CISPA*, 2011. Available: <https://people.cispa.io/cas.cremers/downloads/papers/Cr2011-IKE.pdf>
- [46] T. Dierks and E. Rescola, *The Transport Layer Security (TLS) Protocol Version 1.2*, Aug., 2008 [Online]. Available: <https://www.ietf.org/rfc/rfc5246.txt>
- [47] C. Cremers, M. Horvat, J. Hoyland, S. Scott, and T. van der Merwe, “A comprehensive symbolic analysis of TLS 1.3,” Oct., 2017 [Online]. doi:<https://doi.org/10.1145/3133956.3134063>. Available: <https://doi.org/10.1145/3133956.3134063>
- [48] C. Brzuska, E. Cornelissen, and K. Kohbrok, “Cryptographic security of the MLS RFC, draft 11,” *Cryptology ePrint Archive*, Paper 2021/137, 2021. Available: <https://eprint.iacr.org/2021/137>
- [49] C. W. Chuah, E. Dawson, and L. Simpson, “Key derivation function: The SCKDF scheme,” 2013 [Online]. Available: <https://link.springer.com/content/pdf/10.1007/978-3-642-39218-4.pdf>
- [50] R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon, *The Messaging Layer Security (MLS) Protocol - Draft 13*, Mar. 7, 2022 [Online]. Available: <https://datatracker.ietf.org/doc/pdf/draft-ietf-mls-protocol-13>
- [51] B. Beurdouche, E. Rescorla, E. Omara, S. Inguva, A. Kwon, and A. Duric, *The Messaging Layer Security (MLS) Architecture*, Oct. 4, 2021 [Online]. Available: <https://datatracker.ietf.org/doc/pdf/draft-ietf-mls-architecture-07>
- [52] Signal, “Group chats,” 2022 [Online]. Available: <https://support.signal.org/hc/en-us/articles/360007319331-Group-chats>
- [53] C. Howell, T. Leavy, and J. Alwen, “Wickr’s messaging protocol,” 2017 [Online]. Available: https://wickr.com/wp-content/uploads/2019/12/WhitePaper_WickrMessagingProtocol.pdf

- [54] J. Alwen, “Comparing the wickr messaging protocol to the double-ratchet protocols,” Wickr, Feb. 25, 2020 [Online]. Available: <https://wickr.com/comparing-the-wickr-messaging-protocol-to-the-double-ratchet-protocols/#:~:text=What%20We’ve%20Looked%20At,device%20compromise%20should%20remain%20secret>.
- [55] J. Alwen, “The messaging layer security protocol,” Wickr, Feb. 7, 2020 [Online]. Available: <https://wickr.com/the-messaging-layer-security-protocol/>
- [56] CISCO, “Zero-trust security for webex white paper,” Jul. 12, 2021 [Online]. Available: <https://www.cisco.com/c/en/us/solutions/collateral/collaboration/white-paper-c11-744553.html>
- [57] Northrop-Grumman, “Data link processing and management.” Available: <https://www.northropgrumman.com/what-we-do/land/data-link-processing-and-management/>
- [58] R. Barnes, A. S. Nandakumar, O. Roques, C. Jennings, and J. Idicula, “mlspp,” 5 2022. Available: <https://github.com/cisco/mlspp>
- [59] “Introduction,” *ROS Wiki*. Accessed May 06, 2022 [Online]. Available: <http://wiki.ros.org/ROS/Introduction>
- [60] “Master,” *ROS Wiki*. Accessed May 06, 2022 [Online]. Available: <http://wiki.ros.org/Master>
- [61] “Nodes,” *ROS Wiki*. Accessed May 06, 2022 [Online]. Available: <http://wiki.ros.org/Nodes>
- [62] A. Leon and C. Britt, “Brosito repository,” 5 2022. Available: <https://github.com/brosito/>
- [63] A. Leon and C. Britt, “mls_chat,” 5 2022. Available: https://github.com/brosito/mls_chat
- [64] USC, *Transmission Control Protocol*, Sep., 1981 [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc793>
- [65] J. Postel, *User Datagram Protocol*, Aug. 28, 1980 [Online]. Available: <https://www.ietf.org/rfc/rfc768.txt>
- [66] “ROS tutorials writing publisher subscriber (c++),” *ROS Wiki*. Accessed May 08, 2022 [Online]. Available: <http://wiki.ros.org/ROS/Tutorials/WritingPublisherSubscriber%28c%2B%2B%29>

- [67] “Ubuntu install of ROS noetic,” *ROS Wiki*. Accessed May 08, 2022 [Online]. Available: <http://wiki.ros.org/noetic/Installation/Ubuntu>
- [68] P. J. Denning and C. H. Martell, *Great Principles of Computing* (The MIT Press). MIT Press.

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California