Naval Postgraduate School
Dudley Knox Library

SECURE AND RESILIENT SPACE COMMUNICATIONS

Title	SECURE AND RESILIENT SPACE COMMUNICATIONS
Item Type	Thesis
Authors	Tian, Xisen
URI	https://hdl.handle.net/10945/75107
Publisher	Monterey, CA; Naval Postgraduate School
Date Issued	2025-12
Rights	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.
Download date	2026-06-28 14:06:47
Link to Item	https://hdl.handle.net/10945/75107

Downloaded from NPS Archive: Calhoun

Calhoun
Institutional Archive of the Naval Postgraduate School

Calhoun is the Naval Postgraduate School's repository for research materials and institutional publications created by the NPS community. Materials in Calhoun are openly accessible and will be preserved for future generations.

Dudley Knox Library, Naval Postgraduate School
411 Dyer Rd. Bldg. 339, Monterey, CA 93943
https://library.nps.edu/

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

SECURE AND RESILIENT SPACE COMMUNICATIONS

by

Xisen Tian

December 2025

Dissertation Supervisor: Britta Hale

Distribution Statement A. Approved for public release: Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

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

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** December 2025	**3. REPORT TYPE AND DATES COVERED** Dissertation

4. TITLE AND SUBTITLE SECURE AND RESILIENT SPACE COMMUNICATIONS	**5. FUNDING NUMBERS** 5252.00; 5172.01
6. AUTHOR(S) Xisen Tian	
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) NRO, Chantilly, VA 20151; ONR, Arlington, VA, 22217 | NCWDG, DC, 20395	**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 Distribution Statement A. Approved for public release: Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Through the success of commercial space-based internet service providers, information ranging from military secrets to general internet traffic now traverse space communications links. Traditional space security protocols are not optimized for security while common terrestrial security protocol alternatives degrade under space operational requirements. Moreover, as satellites frequently have long lifespans, systems launched today must be robust to quantum adversaries during their operational lifetime. This dissertation investigates space communications security and proposes methods for not only ensuring secure, resilient communication in delay and disruption prone environments, but optimizing efficiency of post quantum cryptography for such uses even under bandwidth constraints. The dynamic key establishment approaches further support self-healing of security in the face of adversarial attacks. Researched proposals are confirmed with results from cryptographic analyses and efficiency testing.

14. SUBJECT TERMS space, communications, satellite, Messaging Layer Security, MLS, networks, resilient, survivable, scalable, post-quantum, PQ, quantum, delay tolerant networks, DTN, dynamic key establishment	**15. NUMBER OF PAGES** 305
	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

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Distribution Statement A. Approved for public release: Distribution is unlimited.

SECURE AND RESILIENT SPACE COMMUNICATIONS

Xisen Tian
Lieutenant Commander, United States Navy
BS, United States Naval Academy, 2015
MS, Electrical Engineering, Naval Postgraduate School, 2020

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 2025

Approved by:

Britta Hale
Department of
Computer Science
Dissertation Supervisor
Dissertation Chair

James B. Michael
Department of
Academic Affairs

Scott Burleigh
The Johns Hopkins
University,
Applied Physics Lab

Wenschel D. Lan
Space Systems
Academic Group

Geoffrey G. Xie
Department of
Computer Science

Approved by:

Gurminder Singh
Chair, Department of Computer Science

James B. Michael
Vice Provost of Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Through the success of commercial space-based internet service providers, information ranging from military secrets to general internet traffic now traverse space communications links. Traditional space security protocols are not optimized for security while common terrestrial security protocol alternatives degrade under space operational requirements. Moreover, as satellites frequently have long lifespans, systems launched today must be robust to quantum adversaries during their operational lifetime. This dissertation investigates space communications security and proposes methods for not only ensuring secure, resilient communication in delay and disruption prone environments, but optimizing efficiency of post quantum cryptography for such uses even under bandwidth constraints. The dynamic key establishment approaches further support self-healing of security in the face of adversarial attacks. Researched proposals are confirmed with results from cryptographic analyses and efficiency testing.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AAD	Additional authenticated data
ACCE	Authenticated confidential channel establishment
AEAD	Authenticated encryption with associated data
AES	Advanced encryption standard
AKE	Authenticated key exchange
ALPN	Application-layer protocol negotiation
API	Application programming interface
BCB	Block confidentiality block
BIB	Block integrity block
BP	Bundle protocol
BPA	Bundle protocol agent
BPSec	Bundle protocol security
BRB	Bundle receipt block
CA	Certificate authority
CBC	Cipher block chaining
CBOR	Concise binary object representation
CGKA	Continuous group key agreement
ChaCha	ChaCha20 stream cipher
CKA	Continuous key agreement
COSE	CBOR object signing and encryption
CPU	Central processing unit
CTR	Counter mode
DAE	Deterministic authenticated encryption
DN	Destination node
DNS	Domain name system
DoD	Department of Defense
DTKA	Delay tolerant key administration
DTLS	Datagram transport layer security
DTN	Delay tolerant network

ECC	Elliptic curve cryptography
ECDSA	Elliptic curve digital signature algorithm
EMCON	Emissions control
ESA	European Space Agency
fACCE	Flexible ACCE
FS	Forward secrecy
FSC	Flexible secure channel
gACCE	Flexible group ACCE
GCGKA	Generalized continuous group key agreement
GCM	Galois/Counter mode
HMAC	Hash-based message authentication code
HNDL	Harvest now decrypt later
HPKE	Hybrid public key encryption
HTTP	Hypertext transfer protocol
HTTPS	HTTP secure
IETF	Internet Engineering Task Force
IN	Intermediate node
IoT	Internet of things
IP	Internet protocol
KDF	Key derivation function
KEM	Key encapsulation mechanism
KW	Key wrap
LTP	Licklider transmission protocol
MAC	Message authentication code
ML-DSA	Module-lattice digital signature algorithm
ML-KEM	Module-lattice key encapsulation mechanism
MLS	Messaging layer security
NASA	National Aeronautics and Space Administration
NIST	National Institute of Standards and Technology
NPS	Naval Postgraduate School
PCS	Post-compromise security
PKI	Public key infrastructure
PQ	Post-quantum

PQC	Post-quantum cryptography
PSK	Pre-shared key
QUIC	Quick UDP internet connections
RFC	Request for comments
RSA	Rivest-Shamir-Adleman cryptosystem
RTT	Round-trip time
SHA	Secure hash algorithm
SN	Source node
SNI	Server name indication
SSL	Secure sockets layer
TCP	Transmission control protocol
TLS	Transport layer security
UDP	User datagram protocol
USN	United States Navy

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

Having the opportunity to pursue a PhD, the pinnacle of academic study, has been a life long dream of mine. As a first generation college graduate in my family, I could not have gotten to this stage without the sacrifices from my parents, Ping Ren and Lun Tian, who raised me to value education and lifelong learning despite having only finished middle school and high school themselves. As an immigrant, I am grateful to the mentors I met along the way, especially to retired CDR Don Beatty who took me under his wing at an early age to show me the possibilities that my new world offered and inspired me to join the Navy. As a husband and father, I am eternally grateful to my amazing wife Sarah Tian who has supported me through late night submission deadlines, early morning transatlantic meetings, and a slough of week long travel events when she took care of our newborn, Elliot (see Figure 1), by herself. Sarah has been our family's anchor through this journey.

The experience of doing research the doctoral level has been highly rewarding. It has taken me to places I never anticipated research could go when I initially knocked on the door of Dr. Britta Hale as a new PhD student. Her guidance and commitment to excellence have helped to elevate my research a level above others. By her coattails, I have made inroads into the world of cryptographers and protocol standardization bodies which otherwise shun or are devoid of military officers. I will treasure the memories I've made traveling abroad for research collaboration, summer school, and attending the various *Crypto conferences for the rest of my life thanks to Dr. Hale.

Although the day-to-day PhD work has been fairly lonely, I have had the fortune to work with fantastic coauthors and have received support from my fellow students. I would like to thank my coauthors Elsie Mestl Fondevik, Benjamin Dowling, Bhagya Wimalasiri, Joël Alwen, Marta Mularczyk, Paul Westland, and Lee Wang for the enriching collaborations that have resulted in publications both in the academic setting and standards bodies throughout this journey. I would like to especially thank Elsie, Marta, and Ben for hosting when I visited Oslo, Zürich, and London for research collaboration. I am also thankful to my PhD cohort at NPS who have offered encouragement and helped me to chart a course for success.

Figure 1. Four month old Elliot R. Tian providing valuable research insight

Part I

Introduction

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Overview

Space-based communications, which used to be only available to governments, militaries, and large telecom companies, have now been made ubiquitous by advances in efficient rocket designs. Space-based internet service providers now deliver data ranging from internet traffic to sensitive military secrets via space links. Unfortunately for security-minded users, traditional space security protocols have not been optimized for security. Meanwhile, for those pushing to use terrestrial security protocols, the state-of-the-art most commonly used ones degrade in the space environment where delay and disruptions are common. Moreover, due to the long lifespans of space systems, space systems launched today will need solutions to withstand the coming of cryptographically relevant quantum adversaries that can break the traditional cryptography used now.

1.1 Challenges in Space Communications Security

Traditional space security architectures suffer from three fundamental inadequacies that render them insufficient for current and future operational requirements. First, the reliance on rigid networking stacks and preshared keys prevents interoperability, inhibits recovery from compromises, and cannot adapt to evolving mission requirements. Space systems are typically launched with hardware crypto devices that provide preshared keys for communication with ground systems through bent-pipe architectures. This rigidity makes it difficult or impossible to heal from compromises, hinders efficient scaling of networks, and requires mission planners to pre-negotiate all possible secure communication paths, a process that takes months of manual coordination and leaves systems brittle to any changes.

Second, protocols deployed in degraded environments lack formal security analysis confirming baseline guarantees. Decades-long export controls on both space communications technology and cryptographic protocols severely restricted public research, preventing independent security analysis and validation of deployed systems. Commercial entities driving network security advances have focused their resources on optimizing for terrestrial users with abundant bandwidth, low latency, and continuous connectivity—leaving the needs

of constrained environments unaddressed. The absence of rigorous computational analysis means that deployed protocols may be inappropriate for degraded environments or fail to provide the security properties they claim.

Third, the impending obsolescence of classical cryptography against quantum adversaries, coupled with the prohibitive overhead of naive post-quantum replacements, threatens the long-term security of space communications. Adversaries can harvest encrypted transmissions today and decrypt them once quantum computers become available. This is a particularly acute threat for space missions with operational lifetimes spanning decades. While post-quantum cryptographic algorithms exist, they impose substantial costs: public keys and ciphertexts are significantly larger (often an order of magnitude) than their classical counterparts and demand more computation. For bandwidth-constrained and power-limited space systems, these overheads are particularly problematic.

These challenges are not independent but rather compound one another. Rigid security architectures cannot be updated to address newly discovered vulnerabilities or incorporate quantum-resistant algorithms without physical access to spacecraft. Unanalyzed protocols deployed in the field may fail in unexpected ways when subjected to the stress of space environments. And even if quantum-resistant algorithms are deployed, their prohibitive overhead may make them operationally infeasible for resource-constrained missions.

1.2 A Unified Framework for Space Communications Resiliency and Security

This dissertation addresses the aforementioned challenges through an integrated framework comprising three interconnected pillars of contribution. Rather than treating these challenges independently, the framework recognizes their deep coupling and provides solutions that reinforce one another. The rigid, under-analyzed, and quantum-vulnerable systems of the past give way to a flexible, provably secure, and future-proof framework purpose-built for space environments while grounded in formal security foundations.

The three pillars are:

I **Analysis of Security for Degraded Environments**: Establishing formal computational security foundations through analysis of existing protocols and development of

enhanced security models.

II **Design, Analysis, and Implementation of Key Exchange for Space Settings**: Integration of continuous key agreement mechanisms in both Internet Protocol (IP) and Delay-Tolerant Network (DTN) secure channel protocols.

III **Design and Implementation for Post-quantum Optimization**: Reduce the computational and bandwidth cost of post-quantum operations to traditional cryptographic operations levels through amortization.

These pillars work synergistically to dismantle the inadequacies outlined above. The formal analysis foundations (Pillar I) provide the bedrock upon which practical innovations (Pillars II and III) are built, establishing theoretical security frameworks that enable protocol customizations while guaranteeing security properties transfer correctly. The dynamic key agreement mechanisms (Pillar II) address rigidity by replacing preshared keys and synchronous handshakes with asynchronous, self-healing protocols that maintain security even when devices cannot actively participate. The post-quantum optimizations (Pillar III) exploit the asynchronous nature of these mechanisms to achieve quantum resistance without prohibitive overhead, making future-proof security operationally feasible.

Concretely, the integrated framework enables scenarios previously impossible with traditional space security. Consider a future multi-agency space mission involving the National Aeronautics and Space Administration (NASA), the European Space Agency (ESA), and commercial partners operating a constellation of satellites, ground stations, and lunar infrastructure connected via Lunanet. Instead of pre-negotiating all possible secure communication paths and preloading hardware crypto modules, the framework enables dynamic security establishment. Satellites join groups using continuous key agreement for both IP links and DTN store-and-forward paths with a unified security layer ensuring cryptographic consistency across network boundaries. When a satellite is suspected of compromise, the system self-heals through updates initiated by other group members, containing the compromise without requiring the affected satellite to transmit. Throughout this process, post-quantum mechanisms maintain quantum resistance by periodically injecting quantum-resistant entropy during favorable link conditions and relying on efficient updates otherwise.

1.3 Pillar I: Analysis of Security for Degraded Environments

To address the inadequacy of unanalyzed protocols in degraded environments, Pillar I establishes rigorous security foundations through computational cryptographic analysis of existing protocols for key agreement and secure channel establishment. This pillar fills critical gaps through formal computational analysis that models adversarial capabilities, defines precise security properties, and proves security guarantees through rigorous reduction-based proofs. Specifically, the work under Pillar I provides the first comprehensive cryptographic analysis of the Bundle Protocol Security (BPSec), the standard security protocol for Delay Tolerant Networks deployed by space agencies worldwide. This analysis not only validates existing security claims but identifies vulnerabilities and proposes provably secure improvements that strengthen BPSec against realistic threat models. The proposed enhancements, formalized as StrongBPSec, maintain backward compatibility while closing critical security gaps in data authenticity during transit.

Beyond DTN-specific contributions, Pillar I extends the security model of Messaging Layer Security (MLS) to address a fundamental limitation: the inability of receive-only or emission-controlled devices to perform self-updates. In traditional MLS, post-compromise security (PCS) or the ability to sef-heal after a compromise (disgussed later in Section 2.2), requires that compromised parties actively update their own cryptographic state. For bandwidth-constrained users—including not only space systems but also military units operating under electromagnetic emissions control (EMCON) or Internet of Things (IoT) devices with limited uplink capacity—this requirement is often operationally infeasible. The work under Pillar I introduces a *guardianship* mechanism that enables PCS for passive participants through updates initiated by other group members, thereby extending the existing security guarantees to environments where active participation cannot be assumed. This advancement benefits the broader class of constrained devices and represents a foundational improvement to key agreement security models applicable far beyond the space domain.

1.4 Pillar II: Design, Analysis, and Implementation of Key Exchange for Space Settings

To address the rigidity problem of preshared keys and inflexible security architectures, Pillar II integrates modern dynamic and asynchronous key agreement mechanisms into the two dominant network architectures for space communications: Internet Protocol (IP) networks and Delay Tolerant Networks (DTN). While terrestrial security protocols like Transport Layer Security (TLS) and QUIC (formally, Quick UDP Internet Connections) excel in low-latency, continuously-connected environments, they degrade or fail entirely when confronted with the long propagation delays, intermittent connectivity, and limited bandwidth characteristic of space links. Conversely, while DTN architectures are explicitly designed to tolerate delay and disruption, they have historically lacked integrated, standards-based key agreement mechanisms—relying instead on out-of-band key distribution or hardware-based preshared keys that inhibit operational flexibility and security resilience.

For IP-based space communications, the works under Pillar II customize MLS to operate efficiently in multi-user, high-latency, and intermittently-connected scenarios. Specifically, the integration of MLS with QUIC (QUIC-MLS) transforms QUIC from a one-to-one synchronous handshake protocol into an asynchronous, many-to-many continuous group key agreement and secure channel protocol. This advancement not only provides stronger security properties (including post-compromise security and forward secrecy) but also enables new operational capabilities such as dynamic group membership changes and self-healing security without requiring all parties to be simultaneously online. The formal security analysis demonstrates that QUIC-MLS maintains QUIC's performance characteristics while extending its applicability to scenarios where participants cannot engage in synchronous handshakes—making it suitable for both space communications and terrestrial applications requiring resilient group security (e.g., tactical military networks, IoT mesh networks, and collaborative edge computing).

For DTN-based communications, Pillar II introduces the first standardized, in-band key agreement mechanism integrated directly into BPSec. Prior to this work, DTN security relied entirely on out-of-band key provisioning, requiring mission planners to preload all necessary keys before launch or to coordinate key distribution through separate, non-DTN channels. BPSec-MLS integrates MLS as the key agreement layer for BPSec, enabling

dynamic, asynchronous establishment and rotation of cryptographic keys entirely within the DTN infrastructure. This capability is transformative: space systems can now establish secure communications with previously unknown peers after launch, recover from key compromises through post-compromise secure updates, and adapt group membership dynamically as missions evolve. The implementation and performance evaluation demonstrate that BPSec-MLS operates efficiently even under severe bandwidth constraints, with empirical results showing acceptable overhead for realistic space communication scenarios. Together, these IP and DTN integrations represent the first comprehensive suite of standards-based, provably secure, and operationally validated key agreement solutions purpose-built for space environments.

1.5 Pillar III: Design and Implementation for Post-quantum Optimization

To address the quantum vulnerability of space systems, Pillar III develops practical mechanisms for deploying post-quantum cryptography in bandwidth-constrained, power-limited environments. The threat is particularly acute for space communications: adversaries can harvest encrypted transmissions today and decrypt them retroactively once quantum computers become available—a strategy known as "harvest now, decrypt later." For space missions with operational lifetimes spanning decades and handling information that must remain classified for extended periods, transitioning to quantum-resistant cryptography is not optional but imperative. However, post-quantum algorithms impose substantial costs: public keys and ciphertexts are significantly larger (often an order of magnitude) than their classical counterparts, and encryption/decryption operations demand more computation. These overheads are particularly problematic for space systems, where every byte of bandwidth is precious and power budgets are tightly constrained.

Pillar III introduces a novel amortization approach that dramatically reduces the cumulative overhead of post-quantum cryptography in continuous key agreement protocols. Rather than paying the full cost of quantum-resistant key exchanges at every update, the Amortized Post-Quantum MLS Combiner spreads quantum-resistant entropy across multiple updates over time. The key insight is that MLS's asynchronous ratcheted key schedule allows quantum-resistant key material to be injected periodically (via *full* updates) and then

combined with faster classical key agreement operations in intermediate (*partial*) updates. This approach maintains quantum resistance while achieving bandwidth and computational costs approaching those of classical-only implementations. Importantly, the amortization mechanism provides tunable security-efficiency tradeoffs: system designers can adjust the frequency of full post-quantum updates based on their threat model, operational constraints, and desired security margin.

The works under Pillar III validate this approach through implementation and empirical benchmarking. The amortized combiner provides post-quantum security guarantees comparable to (and in some respects stronger than) naive post-quantum MLS while incurring only a fraction of the overhead cumulatively over time. Testing confirms that the amortized approach reduces cumulative bandwidth consumption by up to 94% and cumulative computational overhead up to 85% compared to standard post-quantum MLS implementations. Furthermore, the amortized approach beats the leading hybrid post-quantum design, X-Wing, by similar margins while providing additional post-quantum authentication guarantee that is not provided by X-Wing.

The results demonstrate that quantum-resistant cryptography is not only feasible for space systems but can be deployed at costs roughly equivalent to traditional cryptography. Furthermore, the amortization mechanism is generalizable beyond the specific post-quantum algorithms currently standardized by NIST, providing a framework for integrating future quantum-resistant primitives as cryptanalysis and standardization efforts evolve and algorithms become more efficient. Lastly, as a strictly non-hybrid approach, the combiner makes transitioning to full-PQ cryptography, as posited by the National Security Agency's (NSA) Commercial National Security Algorithms (CNSA) 2.0 [1], trivial. Rather than reinstantiating a new session with new PQ ciphersuites, the combiner session method could be decoupled to full-PQ without interuption by simply terminating and shedding the traditional session half and leaving only the PQ MLS session. This would allow deployed space systems using the Amortized MLS Combiner to transition to full-PQ more seamlessly, if required so, after deployment.

1.6 Synergy of Pillars of Contributions

While each pillar independently addresses critical gaps in space communications security, their true transformative potential emerges through their synergistic integration. The formal analysis foundations established in Pillar I provide the bedrock upon which the practical innovations of Pillars II and III are built. By developing rigorous security models for degraded environments and proving that protocols like (Strong)BPSec and Guardian Messaging Layer Security (MLS) are secure under appropriate adversarial assumptions, Pillar I establishes theoretical security frameworks. These frameworks directly enable the protocol customizations in Pillar II: when QUIC-MLS and BPSec-MLS integrate MLS as their key agreement mechanism, the new security properties provided by Guardian MLS for key agreement provably transfer to both of the integrated protocols. Similarly, the stronger notions of channel authenticity from StrongBPSec also transfer to BPSec-MLS. This eliminates the guesswork and ad-hoc reasoning that historically plague space security—operators now have provably secure modifications that enhance security guarantees rather than inadvertently introducing vulnerabilities.

Pillar II's integration of MLS into both IP and DTN stacks addresses the rigidity problem by replacing synchronous key agreement and preshared keys, respectively, with an asynchronous and dynamic self-healing key agreement mechanism. Future space architectures, such as the National Aeronautics and Space Administration (NASA)'s Lunanet which will employ both IP and DTN protocols, will leverage this modularity to achieve unprecedented operational flexibility. Missions can establish secure communications with previously unknown peers after launch, recover from key compromises without requiring physical intervention or mission replanning, and adapt group membership dynamically as satellites, ground stations, and relay nodes come online or go offline. Critically, the unified use of MLS as the key agreement layer across heterogeneous network stacks (QUIC for IP, BPSec for DTN) provides interoperability: a satellite can seamlessly switch between IP and DTN modes while maintaining continuous security, and multi-stakeholder missions can conduct ad hoc multilateral operations without pre-coordinating cryptographic material. This flexibility is further amplified by Pillar I's guardianship mechanism, which ensures that even receive-only or emission-controlled devices—common in space due to power constraints and operational security requirements—can maintain post-compromise security through updates initiated by other group members.

Pillar III's amortized post-quantum combiner provides the final critical element: future-proofing against quantum adversaries without sacrificing the operational efficiency gains achieved by Pillars I and II. The amortization approach is specifically designed to exploit the asynchronous updates to the cryptographic state of MLS group members that underlies the QUIC-MLS and BPSec-MLS integrations. By spreading quantum-resistant entropy across multiple updates and through two modes of operation (PQ Conf or PQ Conf+Auth) and allowing system designers to tune the frequency of post-quantum operations based on threat models and bandwidth constraints, the combiner ensures that quantum resistance does not come at the cost of prohibitive overhead. This is particularly crucial for space systems: a satellite using BPSec-MLS can perform full post-quantum updates during high-bandwidth contact windows and rely on lightweight partial updates during constrained periods, maintaining continuous quantum resistance throughout its operational lifetime. Moreover, because the amortization mechanism is generalizable, any future advancements in post-quantum algorithms or discoveries of vulnerabilities in current standardized ciphersuites can be incorporated by simply adjusting the combiner's cipehrsuite configuration: the protocol infrastructure established by Pillar II remains intact, and the formal analysis frameworks from Pillar I provide the tools to validate new configurations.

This synergistic integration extends beyond operational benefits to broader advantages in protocol development. The modular separation of key agreement (MLS) from secure channel protocols (QUIC, BPSec) means that improvements to either component can be deployed independently. Advances in continuous key agreement (e.g., new security advancements and novel MLS extensions) can be incorporated into MLS and automatically benefit all integrated protocols. Conversely, optimizations to the channel components of QUIC or BPSec (e.g., better congestion control, improved custody transfer mechanisms) can proceed without disrupting the security layer. The flexible models developed in key agreement and secure channel analysis can be built upon to ensure that future modifications preserve security properties, accelerating the innovation cycle. Furthermore, the generalizable nature of the amortization mechanism means that as the National Institute of Standards and Technology (NIST) and other standardization bodies refine post-quantum algorithms or introduce new quantum-resistant primitives, the framework can accommodate these changes through ciphersuite modifications rather than requiring wholesale protocol redesign. Lastly, as organizations transition away from traditional ciphersuties altogether, the amortization

mechanism allows deployed systems to simply shed the traditional component for trivial transition to full post-quantum instead of by more disruptive means.

In summary, the three pillars collectively address the inadequacies of space security not merely through incremental improvements but through a fundamental paradigm shift. This integrated approach positions space communications to meet the security demands of current missions while maintaining the adaptability and resilience required for the evolving threat landscape and expanding operational requirements of future space endeavors.

CHAPTER 2:
Preliminaries

In this chapter, significant technical topics required for understanding the works contained are briefly introduced at a high level. This is followed by an overview of the organization of this dissertation and the and impacts of this research. Further details can be found within each of the works in subsequent chapters. The technical foundations to the three pillars aforementioned in Chapter 1 are as follows:

- **Cryptographic Analysis** (Section 2.1): The methodology for formalizing and proving security properties of protocols. This foundation directly enables rigorous analysis of protocols and upgrades to key agreement for degraded environments in Pillar I and Pillar II, establishing theoretical security baselines that inform the practical customizations in Pillars II and III.
- **Continuous Key Agreement** (Section 2.2): The paradigm shift from static preshared keys and handshake-based protocols to asynchronous and self-healing key agreement. Central to Pillar I's guardianship mechanism and Pillar II's MLS integrations, continuous key agreement provides the flexibility and post-compromise security essential for resilient space communications.
- **Space Network Architectures** (Section 2.3): The distinction between terrestrial IP proposed for space use and Delay-Tolerant Networks (DTN) and their implications for secure protocol design. This foundation shapes Pillar II's dual integration strategy, ensuring MLS-based solutions function across heterogeneous space communication stacks.
- **Post-Quantum Cryptography** (Section 2.4): The quantum threat landscape and hybrid approaches to quantum-resistant cryptography. This foundation underpins Pillar III's amortization mechanism, enabling practical deployment of post-quantum security in bandwidth-constrained environments.

These foundations are deeply interconnected: cryptographic analysis can help to validate and improve the security of continuous key agreement protocols (Sections 2.1 and 2.2), which must function across both IP and DTN network architectures (Section 2.3) while

incorporating post-quantum optimizations (Section 2.4). The following sections introduce each foundation and explicitly connect them to the dissertation's contribution pillars.

2.1 Cryptographic Analysis: Methodology for Security Claims

Security, in the context of this work, is focused on the topics of key agreement and secure channels. Key agreement is a process by which communicating partners establish a shared key to use in establishing a secure channel used to guarantee the confidentiality, integrity, and authenticity of their communications. Often, key agreement and secure channels go hand-in-hand but in space communications protocols one or both may be underdefined as seen in Chapter 3. When this occurs, security analysis can assist with finding vulnerabilities and improvements.

Security analysis can be approached from several directions yielding unique perspectives against a protocol in question. For example, side-channel analysis examines the physically observable outputs (e.g., signal to noise ratio, CPU temperature) of a protocol to find vulnerabilities in implementations. Other approaches such as modern cryptographic analysis methods abstract the protocol into mathematical terms to examine security claims of the protocol.

This work explores security through the lens of cryptographic analysis of protocols by abstracting them into mathematical terms and proving their (in)security.[1] Modern cryptanalysis, as opposed to classical cryptography, places emphasis on definitions, precise assumptions, and rigorous proofs of security. This allows for a systematic approach to studying and improving security as opposed to classical methods such as bruteforce attacks or systems which relied on the secrecy of their mechanics.

Two complementary strands of analysis underpin modern cryptographic analysis: computational (Turing Reduction based) analysis and symbolic (formal or mechanized) protocol verification. Computational analysis models adversaries as probabilistic, resource-bounded algorithms and seeks reductions that relate protocol security to well-studied hardness as-

[1]The reader is invited to refer to *Introduction to Modern Cryptography* by Jonathan Katz and Yehuda Lindell for a more thorough examination of cryptanalysis. An abridged introduction, based on its content, is presented herein.

sumptions; it provides concrete, quantitative guarantees about confidentiality, integrity, and authenticity under explicit adversarial models. Symbolic analysis, in contrast, abstract protocol logic to check for design-level flaws and logical inconsistencies using symbolic models and automated tools; these methods excel at finding subtle protocol interactions, replay or state-machining bugs, and incorrect message-handling that can evade purely algebraic reasoning.

Symbolic analysis, in its abstraction, assumes "perfect" cryptographic primitives: it treats them (authenticated encryption, signature schemes, etc.) as black-boxes. In doing so, the attacker is restricted by the primitives' definitions which may be imperfect or under-analyzed (see *deterministic authenticated encryption* in Chapter 3). On the other hand, computational analysis is less restrictive in that attackers are modeled as polynomial-time Turing machines and only limited by explicit assumptions of the security of primitives. In this sense, computational analysis allows for more flexibility and depth in modeling the attacker and the threat environment which is precisely what needs to be shown when analyzing space communications. This work utilizes computational analysis which allows the thorough consideration of the cryptographic primitives and algorithms in a more detailed examination of the underlying conditions and assumptions of cryptographic protocols, relevant to this work on tailoring security to space communications.

Computational analysis begins with formalizing the protocol to define a syntax which can be used to formulate definitions of correctness and security about the protocol. In defining security, adversarial powers and limits are developed alongside a security experiment based on reasonable assumptions of threats. Then a proof of security is formulated via a series of Turing reductions to demonstrate that the protocol meets the security definitions. This kind of analysis can help to establish baselines of security from which improvements to security can be made such as the work in Chapter 3 does or advance existing security models to incorporate additional security and functionality such as the work done in Chapters 4 and 7. Overall, this dissertation heavily relies on computational analysis to make and defend claims of security for new and existing space communications protocols. The security properties established through this methodology—particularly for continuous key agreement (Section 2.2) and its integration with network protocols (Section 2.3)—provide the formal guarantees necessary for deploying these protocols in mission-critical space systems.

2.2 Continuous Key Agreement: From Static to Dynamic Security

To address the rigidity problems outlined in Chapter 1, this dissertation employs continuous key agreement protocols that enable asynchronous, self-healing security. Traditional space communications rely on preshared (hardware) keys or session-based key agreement protocols. The commercial industry has been quick to adopt new classes of key agreement protocols. But the US government—especially the Department of Defense (DoD) and space agencies—with their low appetite for risk and lengthy acquisition timelines, have been slow to adopt modern protocols, resulting in the overreliance on preshared keys and inconsistent usage of key agreement protocols. Space systems are often launched with hardware crypto devices that provide preshared keys to communicate with ground systems through bent-pipe architectures. This makes it difficult or impossible to heal from compromises and hinders efficient scaling of networks.

Continuous key agreement fundamentally transforms this paradigm by providing end-to-end (E2E) security with dynamic, asynchronous key updates that enable systems to recover from compromises without requiring physical intervention. E2E security protocols are central to this resiliency, as they ensure that communications across untrusted networks remain private and secure for the end users, regardless of the state of intermediate nodes. Unlike rigid, static security architectures, E2E protocols with dynamic key agreement mechanisms allow for rapid recovery and self-healing after a compromise, limiting the impact of attacks and preventing cascading failures. The recent compromise [2] of US telecommunications and internet service providers (ISP) highlights the risks of relying on intermediaries and static security models. By pivoting to E2E secure protocols with continuous key agreement, space-based communications can achieve greater resiliency, privacy, and operational continuity for both commercial and government users.

2.2.1 Key Agreement Paradigms

Key agreement is the first step in establishing a secure communications channel between parties. The various types of key agreement methods are shown in Figure 2.1. Session-based authenticated key agreement (AKE) protocols like Transport Layer Security (TLS) must perform an explicit handshake every time to produce *fresh* keys. After initialization,

continuous key agreement (CKA) protocols like the Signal double-ratchet are able to ratchet (update) keys in a single step instead of renegotiating new ones. As an added benefit, CKA key updates are asynchronous, meaning that, so long as the update message is eventually received, the recipient is not required to be online when the update is sent, unlike session-based negotiations. This property is crucially important for the space network environments described in Section 2.3, where devices may be offline or unreachable periodically due to orbital dynamics and intermittent connectivity.

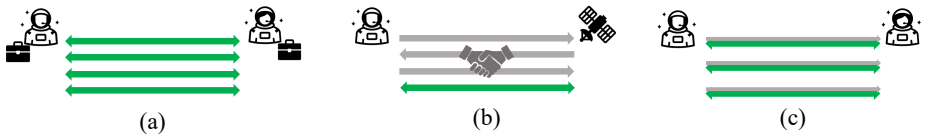


Figure 2.1. Key agreement paradigms: preshared-keys (a), session based (b), continuous key agreement (c). Grey lines indicate messages sent for key agreement and green lines indicate secure channel messages.

Although session-based protocols are also able to perform key updates after a handshake, their new key is derived from the old session key. Whereas the asymmetric handshake provides fresh keys, this symmetric key update only provides a property known as *forward secrecy* (FS). This means that a compromised new key does not allow the attacker access to the old key; however, the attacker would be able to follow the key updates as they continue in the session. CKAs can perform both symmetric and asymmetric key updates. The asymmetric key update provides a property known as *post-compromise security* (PCS) which protects future keys from being derivable by an adversary who has compromised the current key. This ability to *heal* security after a compromise is central to resilient security for space systems.

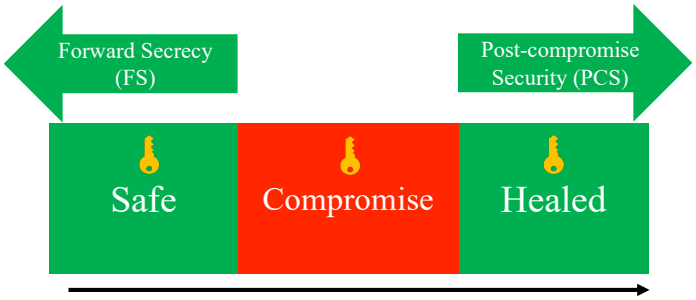


Figure 2.2. Forward secrecy (FS) and post-compromise security (PCS) from the perspective of a period of compromise during communications. FS ensures previous communications aren't affected by the present compromise. PCS allows for healing channel security after the compromise.

2.2.2 Messaging Layer Security (MLS)

As the only standardized CKA protocol by the Internet Engineering Task Force (IETF), Messaging Layer Security (MLS) [3] extends CKA to the group setting as a continuous group key agreement (CGKA) protocol. The MLS protocol is largely based on the concept of an ratcheting tree algorithm which generates group secrets through key encapsulation mechanisms (KEM) and key derivation functions (KDFs) from group members' public-private key pairs (see Figure 2.3). Each group member is represented via a leaf node in the tree and is able to compute the group secret through operations on their path to the root of the tree. Members are added as a new leaf in the tree and join through receiving a public key encrypted version of the new tree state. Group members recompute the tree as the group evolves over time through additions, removals, and updates to the leaf nodes. All operations to the tree can be sent asynchronously: so long as members are able to *eventually* receive the tree updates, they can maintain correct derivation of the group secret. This asynchronous property makes MLS particularly well-suited for the DTN environments detailed in Section 2.3.

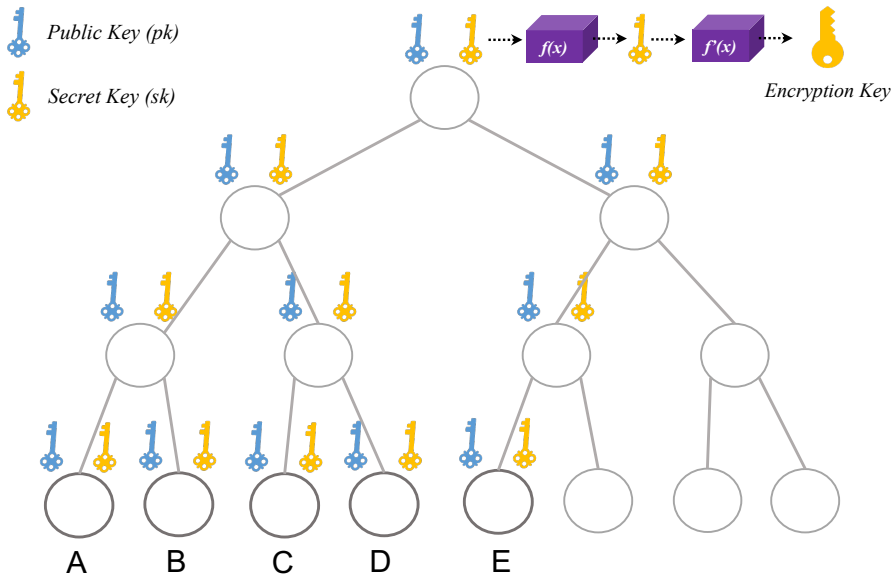


Figure 2.3. The MLS ratchet tree for group members A through E generating group encryption keys from group members' public-private key pairs.

As a CGKA, MLS offers FS and PCS security properties as well as logarithmic scaling in the number of key update operations to the total size of the group. Chapter 6 provides empirical analysis of MLS performance across various metrics and configurations across both IP and DTN networks (Section 2.3). As another benefit, MLS can also use quantum-resistant cryptographic algorithms (Section 2.4), which is a crucial property for space systems that may be flown for decades. Chapter 8 measures the efficiency of a novel amortized MLS construction that far exceeds the performance of naive and leading post-quantum MLS implementations.

MLS serves as the unifying key agreement mechanism throughout this dissertation: it is analyzed and extended in Pillar I (Chapter 4), integrated with both IP and DTN secure channel protocols in Pillar II (Chapters 6 and 7), and optimized for post-quantum deployment in Pillar III (Chapter 8). The security properties of MLS established through computational analysis (Section 2.1) transfer to these integrations, providing provable security guarantees for the protocol stacks.

2.3 Space Network Architectures: IP and Delay-Tolerant Networks

The security protocols employed in this dissertation must function across heterogeneous network stacks characteristic of space environments. Space communications operate differently from terrestrial Internet Protocol (IP) networks. Traditional IP networks generally assume that end-to-end paths exist between communicating nodes, that round-trip times are relatively short (milliseconds), and that connectivity is generally continuous. These assumptions enable protocols like Transmission Control Protocol (TCP) to provide reliable data delivery through mechanisms such as acknowledgments, retransmissions, and congestion control that depend on timely feedback. This, in turn, allows modern key agreement protocols like Transport Layer Security (TLS) and MLS to piggyback on TCP guarantees and make generous assumptions about timely and orderly delivery of setup messages (e.g., handshake and control messages).

In contrast, space environments face unique challenges: propagation delays can range from seconds to minutes or even hours depending on distance, link availability is often intermittent due to orbital dynamics and line-of-sight constraints, and data rates may be severely limited by power and bandwidth restrictions. While these constraints most severely impact deep space (i.e., lunar and beyond) communications, hereto simply referred to as 'delay tolerant environments', near-Earth communications still suffer from intermittent link availability and delays. Moreover, after launch, space systems do not have the luxury of terrestrial systems which have access to virtually unlimited power and they cannot be reset or upgraded on-the-fly. These constraints shape both the choice of continuous key agreement mechanisms (Section 2.2) and exacerbate the challenge of protecting space communications from quantum adversaries (Section 2.4). Regardless, due to the presence of these environmental challenges, to only consider applying TLS or MLS using the TCP/IP stack in space communications is folly.

Delay Tolerant Networking (DTN) addresses these environmental constraints through a store-and-forward architecture based on the Bundle Protocol (BP) [4]. Rather than assuming continuous connectivity, DTN nodes store data in persistent storage and forward it opportunistically when links become available. This custody-based approach tolerates long delays and intermittent connectivity by breaking the end-to-end path into a series of cus-

todial hops, where each intermediate node takes responsibility for reliable delivery to the next hop or ultimate destination.

The integration of modern security protocols designed for terrestrial networks into DTN environments presents unique challenges. As introduced in Section 2.2, MLS provides asynchronous key updates that tolerate delayed delivery, but even so, adapting MLS to function correctly and efficiently in high-latency, intermittently-connected environments requires careful consideration. This dissertation explores customization of the MLS protocol as well as the incorporation of MLS with prominent secure channel protocols to confront the operational realities of both IP and DTN environments. Pillar II addresses these challenges through dual integration: QUIC-MLS for near-Earth IP communications (Chapter 7) and BPSec-MLS for deep-space DTN communications (Chapter 6). Together, these integrations, which have been validated through the computational analysis methods described in Section 2.1, provide a comprehensive secure communications framework spanning the full spectrum of space network architectures.

2.4 Post-Quantum Cryptography: Future-Proofing Against Quantum Threats

Beyond adapting to the operational constraints of space network environments (Section 2.3), space systems must contend with long-term cryptographic threats that threaten the confidentiality and authenticity of communications over their multi-decade operational lifetimes. The security of most widely-deployed public-key cryptographic systems, including RSA and Elliptic Curve Cryptography (ECC), relies on the computational hardness of problems such as integer factorization and solving discrete logarithms. While these problems are intractable for classical computers, Shor's algorithm demonstrates that sufficiently large-scale quantum computers could solve them efficiently, thereby breaking the security foundations of current public-key infrastructure.

Although practical, cryptographically-relevant quantum computers do not yet exist, the consequences of their arrival would be severe. Of particular concern is the "harvest now, decrypt later" threat which attacks the confidentiality of all existing encrypted messages: adversaries can collect encrypted communications today and decrypt them once quantum computers become available. For space systems with operational lifetimes spanning decades

and handling information that must remain classified for extended periods, this threat is especially acute. Even without access to a cryptographically relevant quantum computer, adversaries can threaten data confidentiality by collecting communications today and storing it to be decrypted later by a quantum computer.

In response to this threat, the cryptographic community has developed post-quantum (PQ) cryptographic algorithms based on mathematical problems believed to be hard even for quantum computers. These include lattice-based cryptography (e.g., CRYSTALS-Kyber, CRYSTALS-Dilithium), hash-based signatures (e.g., SPHINCS+), code-based cryptography, and multivariate polynomial systems. The National Institute of Standards and Technology (NIST) has been conducting a multi-year standardization process to evaluate and standardize PQ algorithms, with the first standards published in 2024 that utilizes lattice-based cryptography [5].

However, transitioning to post-quantum cryptography is not simply a matter of replacing one algorithm with another. PQ algorithms typically have different performance characteristics than their classical counterparts: larger key sizes, longer ciphertexts and signatures, and higher computational requirements. For bandwidth-constrained and power-limited space systems, these differences matter. For space systems, the transition period that would normally be afforded to terrestrial systems, shown as *X* in Figure 2.4, shrinks down to simply the period between development and launch. Therefore, security mechanisms deployed today must be resilient against both traditional and quantum adversaries. Moreover, given the uncertainty about which mathematical problems will remain hard against future cryptanalysis (both classical and quantum), there is growing interest in *hybrid* approaches that combine classical and post-quantum algorithms to provide defense-in-depth.

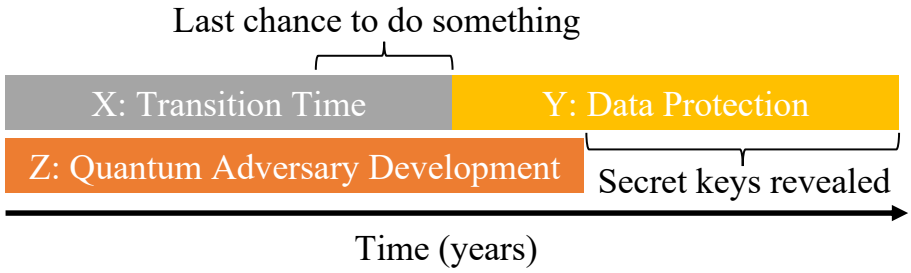


Figure 2.4. Depiction of Mosca's Inequality which states that, given X as the time it takes to transition to quantum resistant security, Y as the time data needs to be protected, and Z as the time it takes for a cryptographically relevant quantum computer to be developed, if $X + Y > Z$ then the security of the underlying data is vulnerable to a quantum adversary. Illustration adapted from Mosca's Inequality [6].

This dissertation investigates the practical integration of post-quantum cryptography in MLS (Section 2.2.2) for use in space communications, examining the tradeoffs between security guarantees, bandwidth efficiency, and computational overhead within the constraints of space network environments (Section 2.3). Particular attention is given to hybrid approaches that balance degrees of quantum resistance with operational efficiency. Pillar III addresses this challenge through a novel amortization mechanism (Chapter 8) that exploits the asynchronous nature of MLS group updates to spread post-quantum overhead across multiple updates, achieving quantum resistance at near-classical efficiency.

2.5 Organization of Contributions

This dissertation collects peer-reviewed and under-review works that together address the three pillars introduced in Chapter 1: (I) analysis of security for degraded environments, (II) design and implementation of key exchange and MLS customizations for space settings, and (III) post-quantum (PQ) optimization and evaluation. The volume is arranged thematically rather than chronologically to guide the reader from foundational analysis through applied design to practical deployment considerations. Concretely, the first part establishes analytic foundations in Delay-Tolerant Networking (DTN) and key rotations in degraded environments; the second part develops and evaluates Messaging Layer Security (MLS) integrations

and protocol customizations for operational use in BPSec and QUIC; and the final part implements and benchmarks post-quantum and hybrid approaches (notably the Amortized PQ Combiner) to make PQ cryptography feasible for constrained space platforms.

Where appropriate, chapters are accompanied by Internet Engineering Task Force (IETF) Internet-Drafts and implementation artifacts included in the appendices. These artifacts document standards-oriented outcomes and provide an engineering pathway for wider adoption and deployment beyond the academic literature.

Part II supports Pillar I, the analysis of existing security protocols for degraded environments. DTN security is scrutinized in Chapter 3 which presents the first formal (computational) security analysis of BPSec and makes improvements to the BPSec protocol. This work has also been adapted into a draft IETF standard document in [7], on track for adoption by the IETF DTN working group. Then, Chapter 4 dives into improvement of the MLS protocol for constrained users by extending post-compromise security to users who are unable to perform self-updates.

Part III supports Pillar II, the design, analysis, and implementation of MLS customizations. Chapter 5 explores integration of MLS as a key agreement mechanism for two leading space communications secure channel protocols proposed by the IETF: BPSec and QUIC. This paper marks a revolutionary shift in network security by modularizing key agreement to be tailorable to operational needs. This paper provides theoretical background for the subsequent works, which test the deployment of MLS into BPSec and analyzes MLS for QUIC in Chapter 6 and Chapter 7, respectively.

Part IV supports Pillar III, design and implementation for PQ optimization with testing of the Amortized PQ Combiner [8], now being standardized by the IETF, which seeks to reduce the cost of using post-quantum ciphersuites in MLS through amortization. Chapter 8 implements [8] and provides benchmark statistics which show that the Amortized PQ Combiner is more efficient while offering greater security than the leading approach for implementing quantum-resistant key agreement.

Finally, Part V concludes the dissertation with a discussion of of the major advances to scientific research and impacts of this work. Additionally, the futures of various submitted IETF drafts are discussed in this chapter. Lastly, directions for adoption and testing of the

research by various stakeholders is also outlined.

2.6 Impacts

The research contained within this dissertation has been presented to and scrutinized by both academic peer-reviewed journals, conferences, and workshops as well as by professionals from the Internet Engineering Task Force (IETF). With the exception of the paper presented in Chapter 4, all works have been accepted to Institute of Electrical and Electronics Engineers (IEEE) or International Association for Cryptologic Research (IACR) venues. Of the four affiliated IETF drafts in the Appendices, the APQ-MLS draft in Appendix A.4 is currently in working group last call for standardization. Along the way, this work has been refined and shaped by feedback from reviewers, engineers, and early adopters.

2.6.1 Advancements to Network Security State-of-the-Art

This work heralds a renaissance in securing space communications. Whereas there was previously a dearth of formal security analysis of space communications protocols, this work confirms the security understanding of existing and space protocols for the first time through computational analysis of BPSec, deployed by space agencies around the world. In the same vein, this work improves security guarantees and offers new features for BPSec which has been promulgated to major standardization bodies for space communications (CCSDS and IETF).

Additionally, this work provides improved security and performance in constrained environments, a superset of space, at both the fundamental key agreement level and as well as the secure channel level. At the key agreement level, MLS as a protocol has been tailored for offline and emission controlled devices (Chapter 4) which previously were unsupported. At the secure channel level, BPSec has been fitted with an asynchronous and dynamic key agreement mechanism through BPSec-MLS (Chapter 6) which, for the first time, provides BPSec users with an in-band standardized key agreement mechanism. This work also improves security guarantees and functional benefits for a major IP stack protocol, QUIC. The analysis and design of QUIC-MLS (Chapter 7 and appendix A.3) enables QUIC to be used as an asynchronous many-to-many continuous key agreement and secure channel protocol as opposed to being limited to a one-to-one synchronous authenticated key agreement

secure channel protocol. This not only improves the security for QUIC's traditional (internet) userbase but also enhances functionality and efficiency for non-traditional multilateral userbases such as those operating in space and other constrained devices.

Finally, this work demonstrates that post-quantum cryptography (PQC) can be deployed at significantly lower overhead than existing approaches, showing that it is attainable for space systems, while also outperforming the security offered by common PQC approaches. Furthermore, the amortization mechanism of APQ-MLS (Chapter 8) can be generalized to other asynchronous key agreement protocols as well as ciphersuites beyond those currently proposed for PQC.

2.6.2 Real World Impacts

Major internet and space standards bodies, academic institutions, government organizations, and industry have embraced the key findings of this work. The Consultative Committee for Space Data Systems (CCSDS) which promulgates communications standards for the majority of space agencies (NASA, ESA, etc.) are in the process of drafting standards based off works on StrongBPSec and BPSec-MLS from Chapters 3 and 6, respectively, which will also be introduced at the IETF for wider internet use. In terms of financial support, work in Chapter 8 was generously sponsored by the National Reconnaissance Office which designs and manages collections satellites for the U.S. government. Furthermore, the European Space Agency has extended invitations to test BPSec-MLS and QUIC-MLS on a system to be launched in 2026 based on Chapter 6 and Chapter 7. In the realm of industry adoption of this work, PhoenixR&D and individual members of the IETF MLS working group have implemented and tested the post-quantum MLS combiner in Appendix A.4. Their feedback from testing have been incorporated into the newest draft version currently in working group last-call for standardization. Furthermore, the US Navy's Program Executive Office for Command, Control, Communications, Computers and Intelligence will be testing APQ-MLS in their upcoming Rim of the Pacific Exercises (RIMPAC). Lastly, the Navy Engineering Duty Officer community has assigned LCDR Xisen Tian to the Naval Research Laboratory in a research capacity to support applying this research to military information operations.

Part II

Analysis of Security for Degraded Environments

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Cryptography is Rocket Science: Security Analysis of BPSec

Addressing the inadequacy of unanalyzed protocols

As established in Chapter 1, traditional space security architectures suffer from a critical inadequacy: protocols deployed in degraded environments lack formal security analysis confirming baseline guarantees. This vulnerability stems from decades of export controls that severely restricted public research on space communications technology and cryptographic protocols, preventing independent security analysis and validation of deployed systems. Meanwhile, commercial entities driving network security advances focused resources on optimizing for terrestrial users with abundant bandwidth, low latency, and continuous connectivity—leaving the needs of space and other constrained environments unaddressed. The absence of rigorous computational analysis means that deployed protocols may be inappropriate for degraded environments or fail to provide the security properties they claim. Space agencies worldwide have deployed Bundle Protocol Security (BPSec) as their secure channel protocol for Delay Tolerant Networks without formal cryptographic validation. Mission owners require confidence that their security protocols actually deliver the security guarantees claimed—not merely informal assurances but provable security against well-defined adversarial threats.

Pillar I: Establishing Formal Security Foundations. Recall from Chapter 1 that Pillar I addresses this inadequacy by establishing rigorous security foundations through computational cryptographic analysis of existing protocols for key agreement and secure channel establishment in degraded environments. Rather than accepting informal security claims, Pillar I applies formal computational analysis that models adversarial capabilities, defines precise security properties, and proves security guarantees through rigorous reduction-based proofs. Under Pillar I, the first comprehensive cryptographic analysis of BPSec validates existing security claims while identifying vulnerabilities and proposing provably secure improvements (StrongBPSec), and the guardianship mechanism extends MLS to enable post-compromise security for receive-only or emission-controlled devices. These formal

foundations provide the bedrock upon which the practical innovations of Pillars II and III are built: when protocols integrate MLS as their key agreement mechanism or adopt StrongBPSec's enhancements, the proven security properties transfer correctly to the integrated systems.

Chapter Contribution and Impact. This chapter provides the first formal cryptographic analysis of Bundle Protocol Security (BPSec), the IETF-standardized secure channel protocol used by space agencies worldwide including NASA and the European Space Agency. Space networking has become an increasing area of development with commercial satellite networks such as Starlink and Kuiper, and increased governmental space presence. Yet historically such network designs have not been made public, leading to limited formal cryptographic analysis of the security offered by them. BPSec is one of the few public protocols used in space networking, making this analysis critically important for establishing baseline security understanding.

At the highest level, this work transforms space security from informal assurance to provable guarantees. By developing formal computational security models and proofs, mission planners can now make informed decisions about protocol configurations based on mathematical certainty rather than intuition. The security properties established here directly enable the BPSec-MLS integration in Chapter 6: because this chapter proves BPSec secure under appropriate adversarial models, mission security designers can confidently combine BPSec with MLS knowing that the security guarantees of both components are preserved in the integrated system.

At the protocol level, this work builds a new security channel model based on the security goals stated in the IETF standard and proves BPSec secure under its default security context. BPSec utilizes authenticated encryption with associated data (AEAD) and message authentication codes (MAC) to provide confidentiality and integrity to bundle data. Although claims of security are made for BPSec in [9] and [10], no formal cryptanalysis had been conducted prior to this work. The analysis reveals issues with message loss detection and destination awareness—subtle vulnerabilities that could undermine security properties in operational deployments.

At the technical level, this chapter synthesizes all relevant IETF specifications [4], [9], [11] and published literature [10] on Bundle Protocol (BP) and BPSec to formally analyze the

protocol through computational cryptanalysis. As the first of its kind, this work develops a new Flexible Secure Channel (FSC) model specifically designed to capture the unique characteristics of DTN environments: intermittent connectivity, custody-based forwarding, and the separation of source, transit, and destination nodes. The proposed enhancements, formalized as StrongBPSec, maintain backward compatibility while closing critical security gaps in data authenticity during transit, ensuring that destinations can detect missing message components and attribute data to its true source.

Real-World Impact and Standardization. This publication has been peer-reviewed and published in the International Association for Cryptologic Research (IACR) Communications in Cryptology journal, representing the cryptographic community's validation of the formal analysis methodology and security results. The work has been presented to the Consultative Committee for Space Data Systems (CCSDS)—which promulgates communications standards for the majority of space agencies worldwide—and to the Internet Engineering Task Force (IETF) for consideration for standardization. As a result, the IETF draft specification in Appendix A.1 has been proposed to the Delay Tolerant Networks working group for adoption and standardization. This standardization pathway ensures that the security improvements identified in this work can be deployed across the global space communications infrastructure, benefiting NASA's Lunanet, ESA's deep space missions, and commercial satellite constellations alike.

Joint Work and Contributions. This chapter presents joint work with Benjamin Dowling (Kings College London), Britta Hale (Naval Postgraduate School), and Bhagya Wimalasiri (Kings College London). It has been reprinted here with permission from all authors. The 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. The candidate's specific contributions were in the background research for formalizing BPSec in Sections 1-2 with existing standards, alignment of the Flexible Secure Channel (FSC) model with BP limitations and constraints in Section 3, and constructing draft proofs for the FSC security for BPSec (Theorem 1) in Section 5 with further joint refinement with co-authors. General security analysis, and security improvements, dubbed StrongBPSec with Read Receipts, was joint work with the co-authors.

Chapter Abstract

Space networking has become an increasing area of development with the advent of commercial satellite networks such as those hosted by Starlink and Kuiper, and increased satellite and space presence by governments around the world. Yet, historically such network designs have not been made public, leading to limited formal cryptographic analysis of the security offered by them. One of the few public protocols used in space networking is the Bundle Protocol, which is secured by Bundle Protocol Security (BPSec), an Internet Engineering Task Force (IETF) standard. We undertake a first analysis of BPSec under its default security context, building a model of the secure channel security goals stated in the IETF standard, and note issues therein with message loss detection. We prove BPSec secure, and also provide a stronger construction, one that supports the Bundle Protocol's functionality goals while also ensuring destination awareness of missing message components.

3.1 Introduction

Originally developed for deep space communications, the Bundle Protocol Security (BPSec) [12] supports Delay Tolerant Networks (DTN) [10] as an application layer secure channel protocol to facilitate a store-and-forward paradigm for sending messages between nodes. It was specifically designed with space system security in mind, a use case where security protocols used on the internet today, such as IPSec [13] and TLS [14], are sub-optimal due to latency, delays, and bandwidth constraints. BPSec was designed to be more suitable for distressed environments where delivery is not guaranteed, bandwidth is a premium, and roundtrip times are on the order of minutes or even hours. This has made it suitable for use outside of deep space networks with applications extending to the Internet of Things (IoT) [15]. Such breadth of uses and interest has led BPSec to be standardized by the Internet Engineering Task Force (IETF). Yet, to date, there has been no formal cryptographic analysis of BPSec. This work provides a closer look at formalizing and strengthening space system channel security in a way that is appropriate under the intended use case constraints.

We give an example execution of BPSec in Figure 3.1. In this example, the satellites orbiting Earth act as bundle protocol agents (BPA) who create, forward and receive messages. The movement of these satellites and their availability to forward and receive messages are in a constant state of change which in turn creates a network infrastructure prone to high

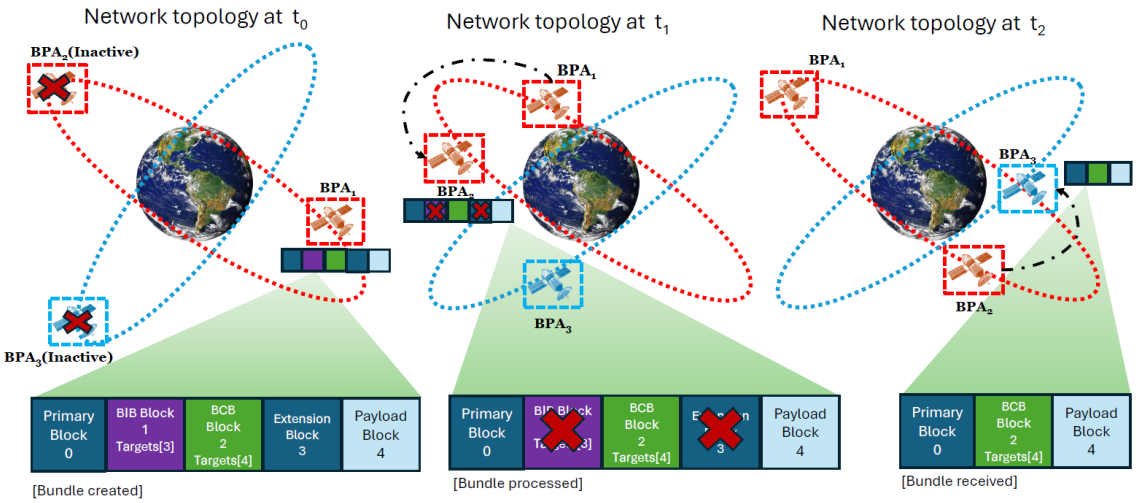


Figure 3.1. An execution of BPSec protocol. BPA_1 creates and forwards a bundle to BPA_2. BPA_2 processes blocks from the bundle, forwarding the modified bundle to BPA_3. This processing and discarding of bundle blocks by an intermediate BPA is legitimate behavior within a BPSec execution, provided that the corresponding intermediary has been authorized to process bundles by the relevant policies.

latency, connection disruptions and propagation delays. For instance in Figure 3.1, while BPA_1 creates the message at t_0, due to the unavailability of a viable path, it cannot be forwarded to the intended destination (BPA_3) immediately and thus stores the message until t_1. In this way, the unique network infrastructure BPSec operates within differs to the terrestrial internet. Within such DTN infrastructures, BPAs store messages for long periods and forward them once an appropriate link becomes available, until the message reaches its final destination.

Underlying BPSec is the Bundle Protocol, a DTN protocol analogous to other networking protocols such as TCP/IP. NASA has included Bundle Protocol version 7 in its three-phased approach for rolling out DTN-based communication links by 2030 [16]. The Bundle Protocol requires security to be handled separately, a feature that BPSec provides while being inherently bound to the functionality requirements of the Bundle Protocol. Furthermore, customizable security contexts that define the ciphersuite and parameters used by BPSec participants allow for fine-tuning of the level of BPSec's security. Authentication and/or a combination of confidentiality and authentication are applied through use of Block Integrity

Blocks (BIB) and Block Confidentiality Blocks (BCB). These blocks store the resultant tags of Message Authentication Code (MAC) and Authenticated Encryption with Associated Data (AEAD) calculations along with parameters used to generate them (i.e. nonces, SHA variant, wrapped key) [11]. This concept of security reference blocks is intended to provide a great deal of flexibility to securing data blocks, where an intermediate node (e.g., a satellite) can add or strip security layers.

Notably, BPSec does not aim to provide key agreement nor does it establish freshness of a session in entity authentication, aiming to instead offer a secure channel protocol under the assumption of preestablished keys. It therefore lacks the common handshake component. If BPSec only offers data authentication and/or confidentiality, one could ask why the protocol exists, e.g., instead of just applying ciphersuites to Bundle Protocol messages as needed. This question points towards the subtlety of BPSec, namely that it aims to achieve these properties among potentially several corresponding members even when intermediate nodes can add more data to the transmission and/or strip off corrupted data. The reader can here think of information distribution among satellites, where intermediate satellites must relay data or even add to a message, while it is also possible for data to be corrupted in transit, necessitating partial message discardment to preserve bandwidth. BPSec thus aims to define a channel security protocol that can modularly add and remove data, even while achieving its security goals.

3.1.1 BPSec: A Flexible Secure Channel

Typically, a secure channel is formed between two communication parties who have previously established a shared key. The security provided by a secure channel construction predominantly focuses on the *confidentiality* and *integrity* of transmitted data, i.e., only the sender and their respective receiver should be able to read and modify the data transmitted. To this extent, there is a rich body of work that formally captures the notions of secure channels as standalone primitives. The seminal works of [17] and [18] introduced the notions of *stateful authenticated encryption* and *authenticated encryption with associated data*, respectively, formalizing early ideas on secure channels. Protections against message replays, reordering, and dropping were early features in [17]. More recently, work has emphasized and differentiated the nature of data transmitted within secure channels, between fragmented streams of data and atomic messages, and formalized the notion of

stream-based secure channels [19]; hierarchies of how channel AEAD notions relate [20]; and multi-key security [21], enabling the analysis of secure channel protocols such as TLS 1.3 [14]. The TLS protocol has in fact played a key role in several works, motivating a better understanding of secure channels, with related works spanning robustness [22], channel termination [23], and alternative channel security notions [24].

While BPSec is a secure channel that aims to establish confidentiality and integrity of transmitted data, modelling BPSec within any existing model for secure channels is difficult due to its unique construction. A typical BPSec node simultaneously maintains a set of shared keys with multiple parties, including the source and destination of a bundle, as well as any intermediate node that might process the bundle on its way towards the final destination. Moreover, unlike a typical message transmitted within a secure channel, the length of a BPSec bundle may be constantly changing, which is an inherent part of its construction. Intermediate nodes that process a bundle may add or remove layers of security from a bundle as per their local policies for processing. This layering of security may sound similar to the design of Tor onion routing [25] but we highlight that these two protocols are strikingly dissimilar for various reasons; unlike a Tor circuit, BPSec paths cannot be pre-established and are subject to change hop-by-hop; message re-encryption is strictly prohibited in BPSec; BPSec integrity checking can be performed on a hop-by-hop basis and is not necessarily end-to-end; anonymity is not a property captured within the BPSec design. Furthermore, BPSec allows for the partial processing of bundles, i.e., intermediaries may only process blocks from a bundle for which they share a key with the respective source of that block, which may be the bundle source or an intermediate node. This rather "flexible" secure channel construction of BPSec cannot be successfully captured within the rigid formalisms of any existing secure channel frameworks. Thus, we introduce a novel Flexible Secure Channels (FSC) security definition that is capable of capturing the unique nature of security goals for the BPSec protocol: confidentiality and integrity guarantees of individual message blocks within a bundle. We note that our formalism is "flexible" both in the sense that it captures the fluid nature of BPSec security but can also be easily generalized to analyze the security of any channel protocol.

3.1.2 BPSec Overview

In this section we restate and clarify core components of BPSec RFC 9172 [12] and the underlying Bundle Protocol version 7 RFC 9171 [26]. This is a high-level description; a detailed algorithmic description of BPSec is presented later in Figure 3.7 which is based upon the mandatory minimum Default Security Context specified in [11].

Entities in a BPSec Channel The *Bundle Protocol Agent* (BPA) is described as a node component that offers the Bundle Protocol services and executes its procedures. We employ a slight abstraction of this for simplicity, and refer to the BPA as the node itself. Possible BPAs may include the *Source Node* (SN) that is the originator of the bundle and the *Destination Node* (DN) which is the intended recipient. *Intermediate Nodes* (IN) may also receive and process bundles – potentially adding or dropping component blocks – before forwarding the transmission. If a BPA $\in$ {SN, IN} adds a security block to a bundle (i.e., that it adds encryption or authentication) it is also called a *security source*. Similarly, if a BPA $\in$ {IN, DN} removes a security block (e.g. decrypting a BCB target) it is also called a *security acceptor*. If the same BPA does not remove the block (e.g. verifying the MAC of a BIB), it is called a *security verifier*.

We give the high-level components of a bundle below:

- *Primary Block* The primary block carries information about the SN, DN, and lifespan of the bundle among other basic bundle identification and forwarding values. There is exactly one primary block per bundle, followed by possible extension blocks and finally a payload block.
- *Extension Block* An extension block provides additional functionality for bundle processing through annotative information (e.g. previous node block, bundle age block, abstract security block, etc.). BIB and BCB are extension blocks that have identical structures as abstract security blocks – both delineating ciphers, parameters, etc. for the target blocks they apply to. INs may also add extension blocks.
- *Payload Block*: There is one payload bock per bundle, always the last block in the bundle, which carries the application data. Since the Bundle Protocol can be used as an encapsulating protocol for another protocol (e.g., an application layer protocol and data) a payload block may be a *partial payload* or *fragmented payload* containing only a segment of the overall payload.

- *Target Block*: The block within a bundle to which a security operation is applied.
- *Security Context*: A security context includes assumptions, algorithms, configurations, and policies that are used to implement security. [2]

Important BPSec Design Decisions through Example BPSec focuses on *block-level granularity and interactions*. Security operations are applied to individual blocks within a bundle according to the security context of a BPA. In this way INs between a SN and DN can add security operations just like a security source, or indeed remove security as if they were an acceptor for the bundle. Figure 3.2 shows an IN, Charlie, adding BCBs to the bundle it received, acting as a relay from Alice to Bob. While Alice only provided authentication to Block 2 and 4 (creating a BIB block 1 that contains the applicable MAC tags and algorithm information), Charlie decides to AEAD-encrypt certain blocks as well, namely *target blocks* 1, 2, and 4. These are Alice's original blocks including the BIB itself. We also demonstrate a separate BCB in Block 6 that targets only Block 3. It is not consolidated with BCB Block 5 because it uses a different parameter-set. Moreover, this AEAD encryption may have been realized with an encrypt-then-MAC mode vs e.g. GCM; in such cases, BPSec requires the MAC tag output from the BCB to be placed in the BCB block vs being considered as part of the ciphertext in Block 3.

If Charlie shares the same security context and keys with Alice and Bob, then Bob will be able to decrypt and verify the bundle (if the bundle is delivered honestly). Otherwise, if the DN Bob does not have the requisite keys, at least one other IN will be needed to process the BCB blocks for Bob, i.e., to act as a *security acceptor*. Bundle encapsulation, whereby a bundle is encapsulated as a payload of a wrapping bundle, is recommended when the bundle may arrive at the DN before being processed by a security acceptor.

3.1.3 BPSec Challenges

We note that the DTN threat model gives an attacker complete network access, affording them read/write access to bundles traversing the network. Eavesdropping, modification, topological, and injection attacks are all described in [12, Section 8]. Therein, these "on-path attackers" can be unprivileged, legitimate, or privileged nodes depending on their access to cryptographic material: unprivileged nodes only have access to publicly shared information, legitimate nodes have additional access to keys provisioned for itself, and

[2] Security contexts are user-defined but RFC 9173 specifies a mandatory-to-implement security context.

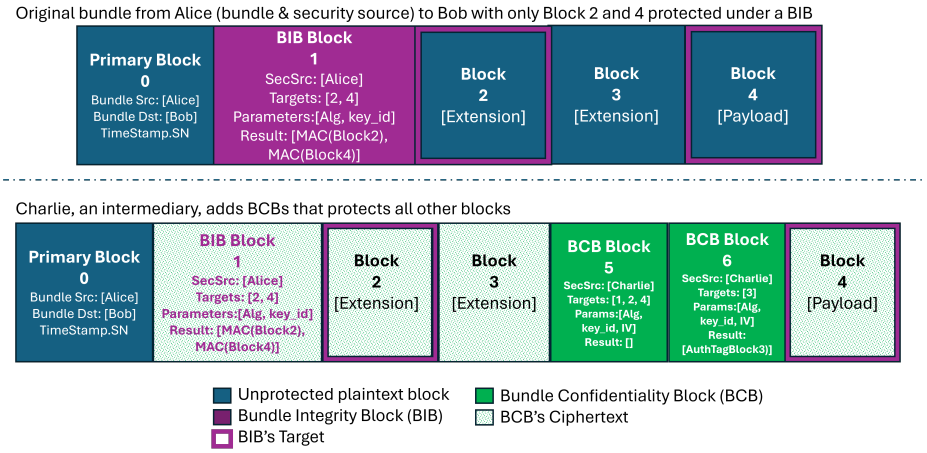


Figure 3.2. BPSec example showing block interactions between BCB and BIB made by different BPA security sources, Alice and Charlie, for a bundle intended for Bob.

privileged nodes have further access to keys (privately) provisioned for others. In our symmetric key based security context from [11], all BPSec participants are privileged while attackers may be unprivileged or privileged, but provide no guarantees against privileged attacks. In an effort to distinguish malice by INs, we further abstract these classes into *honest* and *dishonest* nodes in our analysis. Honest INs are privileged nodes that faithfully execute the role of a BPA as described in Section 3.1.2. Dishonest INs are unprivileged nodes that attempt to violate the integrity or confidentiality of blocks it processes (e.g. by dropping or modifying blocks), and captured by our adversary model. We observe a specific gap in guarantees BPSec provides through the BIB, BCB, and the default security context. Specifically, BPSec has no cryptographic auditing mechanism for detecting unprivileged modifications to a bundle between the SN and DN.

Claim: BPSec protections against plaintext modification are insufficient and can lead to a self-imposed denial-of-service. A similar argument can be made for authenticated encrypted BCB target blocks.

- Suppose an unprivileged dishonest IN strips the BIBs and/or BCBs from all bundles it receives and forwards and appends an additional bit to all their associated security targets. BPSec has no in-band mechanisms (i.e. cryptographically enforced) to detect or correct this kind of modification (aside from encrypting the BIB targets after they

are authenticated). According to [12, 5.1.2], other BPAs who receive this stripped bundle will use an out-of-band security policy mechanism to determine whether to drop, modify, or forward the bundle. Under the recommended security policy, BPAs will remove all target blocks that were supposed to be protected by a BIB. This could lead to dropping the entire bundle if the security policy specified that the primary block must be BIB protected.

- This paradigm sacrifices availability over authenticity. One could argue that availability was never guaranteed by BP and that it is out of scope of BPSec to prioritize availability over authenticity (resp. confidentiality) [3].
- Under a unprivileged dishonest attacker model the DN would not have a means to detect dropped target blocks. This may lead to the DN incorrectly assuming that they have a complete message and acting on it, even if core actionable information was in the dropped blocks.

The core impact of the issues highlighted above is that the destination BPA is unaware of what messages have been removed by the intermediate nodes. To address this, we provide a strong BPSec variant StrongBPSec that offers a *ledger* and *read receipts*. Assuming that SNs always add a *ledger* block and that DNs will not accept a bundle without one, read receipts are designed to provide a verifiable record for processed blocks. This ensures transparency while adding an additional layer of integrity protection between SN and DN within BPSec's symmetric key constraints. The maintenance of a *ledger* block through read-receipts guarantees the integrity of the original bundle created by SN, while simultaneously permitting honest intermediaries to process and discard SN-created security blocks. This should not be construed as the "return-receipt" from a Bundle Status Report [27] which is a out-of-band (i.e. separate from the bundle) policy-driven acknowledgment of receipt or change status. In fact, an on-path-attacker can simply drop all Bundle Status Reports whereas StrongBPSec offers a stronger in-band (i.e. to the bundle itself) cryptographically enforced audit log.

[3]We also note that, if only a basic integrity check – not authenticity – of the data is required, the use of the BIB to provide integrity protection is unnecessary as cyclic redundancy check (CRC) codes already exist in BP blocks. CRCs do not provide authenticity.

3.1.4 Contributions

As can be seen, the BPSec secure channel environment, security expectations, and functionality are quite unlike traditional secure channel protocols, creating a non-trivial environment for analysis. This work formalizes the complex goals of BPSec, provides an analysis of the protocol. This includes a model and proof corresponding to the type of security BPSec can be claimed to offer. We note issues in BPSec and outline normative security goals that it cannot assure. Furthermore, we offer recommendations to enhance security guarantees within the intended design constraints of BPSec by introducing StrongBPSec, which reinforces the integrity assurances of BPSec. StrongBPSec accomplishes this by maintaining a verifiable ledger of changes, allowing the intended recipient of a bundle to independently verify the integrity of modifications made throughout a bundle's journey. We also present a stronger model and analysis to match the improved security of StrongBPSec.

3.2 Cryptographic Assumptions

In this section we describe the cryptographic primitives that are used to build BPSec and define their security. Specifically, we use deterministic authentication encryption DAE to model the BPSec key wrap algorithm, authenticated encryption with associated data AEAD to model their use of symmetric encryption and message authentication codes MAC to model their use of authentication primitives. We begin by introducing the formalism and security for DAE by repeating the definitions of Rogaway and Shrimpton [28] who introduced this primitive.

Definition 1 (Deterministic Authenticated Encryption)**.** *A deterministic authenticated encryption scheme* DAE *is a tuple* $\mathsf{DAE} = \{\mathcal{K}, \mathsf{Enc}, \mathsf{Dec}\}$ *with associated with key space* $\mathcal{K}$*, message space* $\mathcal{M} \subseteq \{0,1\}^*$ *and headers* $\mathcal{H} \subseteq \{0,1\}^{**}$*. The key space* $\mathcal{K}$ *is a set of strings or infinite strings endowed with a distribution. For a practical scheme there must be a probabilistic algorithm that samples from* $\mathcal{K}$*, and we identify this algorithm with the distribution it induces. We denote by* $\mathsf{DAE.Enc}_K(H, M)$ *the deterministic* DAE *encryption algorithm that takes as input a key* $K \in \mathcal{K}$*, a header* $H \in \mathcal{H}$ *and a message* $M \in \mathcal{M}$ *and outputs either a ciphertext* $C \in \{0,1\}^*$ *or a distinguished failure symbol* $\perp$*. We denote by* $\mathsf{DAE.Dec}_K(H, C)$ *the deterministic DAE decryption algorithm that takes as input a key* $K \in \mathcal{K}$*, a header* $H \in \mathcal{H}$ *and a ciphertext* C *and returns a string* M'*, which is either in the message space* $\mathcal{M}$ *or a distinguished failure symbol* $\perp$*. We assume that* $M \in \mathcal{M} \implies \{0,1\}^{|M|} \subseteq \mathcal{M}$*. The*

ciphertext space is $C = \{\mathsf{DAE.Enc}_K(H, M) : K \in \mathcal{K}, H \in \mathcal{H}, M \in \mathcal{M}\}$. *Correctness of an* DAE *scheme requires that* $\mathsf{DAE.Dec}_K(H, (\mathsf{DAE.Enc}_K(H, M)) = M$ *for all* K, H, M *in the appropriate space.*

We note that in our construction presented in Figure 3.7 we have adopted the notation KW.Enc and KW.Dec instead of DAE.Enc and DAE.Dec, respectively, as defined above. We wanted our notation to adhere to the vocabulary of the BPSec Default Security Context [11] and thus WLOG we substituted DAE with KW within our construction.

Next, we describe the security of DAE schemes. On a high level a DAE scheme ensures confidentiality of the underlying plaintext and authenticity of the ciphertexts. We provide this in Definition 2.

Definition 2 (DAE Security)**.** *Let* $\mathsf{DAE} = \{\mathcal{K}, \mathsf{Enc}, \mathsf{Dec}\}$ *be a DAE scheme with header space* H*, message space* M*, and expansion function* e*. The advantage of adversary* $\mathcal{A}$ *in breaking* dae*-security is defined as*

$$\mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{A}}(\lambda) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{Enc}_K(.,.),\mathsf{Dec}_K(.,.)} \implies 1\right] - \Pr\left[\mathcal{A}^{\$(.,.),\perp(.,.)} \implies 1\right].$$

Upon query $H \in \mathcal{H},\ M \in \mathcal{M}$*,* $\mathcal{A}$*'s random oracle* $\$(.,.)$ *returns a random string of length* $|M| + e(H, M)$*. The* $\perp(.,.)$ *oracle returns* $\perp$ *on every input. We assume that* $\mathcal{A}$ *does not ask* (H, C) *of its right oracle if some previous left oracle query* (H, M) *returned* C*; does not ask* (H, M) *of its left oracle if some previous right-oracle query* (H, C) *returned* M*; does not ask left queries outside of* $H \times M$ *; and does not repeat a query.*

Definition 3 (Message Authentication Code (MAC) security)**.** *A message authentication code (*MAC*) scheme is a tuple of algorithms* $\mathsf{MAC} = \{\mathsf{KGen}, \mathsf{Tag}\}$ *where:*

- KGen *is a probabilistic key generation algorithm taking input a security parameter* λ *and returning a symmetric key* k*.*
- Tag *is a potentially probabilistic algorithm that takes as input a symmetric key* k *and an arbitrary message* m *from the message space* $\mathcal{M}$ *and returns a tag* τ*.*

Security is formulated via the following game that is played between a challenger C *and an algorithm* $\mathcal{A}$*:*

1. *The challenger samples* $k \xleftarrow{\$} \mathcal{K}$
2. *The adversary may adaptively query the challenger; for each query value* m_i *the challenger responds with* $\tau_i = \mathsf{Tag}(k, m_i)$
3. *The adversary outputs a pair of values* (m^*, τ^*). *such that* $(m^*, \tau^*) \notin \{(m_0, \sigma_0), \ldots (m_i, \sigma_i)\}$

We define the advantage of $\mathcal{A}$ *in breaking the strong unforgeability property of a MAC* MAC *under chosen-message attack to be:*

$$\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{A}}(\lambda) = \Pr\left((m^*, \tau^*) \notin \{(m_0, \sigma_0), \ldots (m_i, \sigma_i)\}\right)$$

We say that MAC *is classically* sufcma*-secure if, for all PPT algorithms* $\mathcal{A}$, $\mathsf{Adv}^{\mathsf{sufcma}}_{\mathsf{MAC},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ.

Our definition for AEAD-PRIV and AEAD-auth closely follow the work of Rogaway [18] which we describe next for completion.

Definition 4 (AEAD Security)**.** *We define an* authenticated-encryption scheme with associated-data (an AEAD-scheme*) as a three-tuple* $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$. *Associated to* Π *are sets of strings* $\mathsf{Nonce} = \{0,1\}^n$ *and* $\mathsf{Message} \subseteq \{0,1\}^*$, *the latter having a linear-time membership test and satisfying* $M \in \mathsf{Message} \implies M' \in \mathsf{Message}$ *for any* M' *of the same length as* M. *Further associated to* Π *is also a set* $\mathsf{Header} \subseteq \{0,1\}^*$ *that has a linear-time membership test. The key space* $\mathcal{K}$ *is a finite nonempty set of strings. The encryption algorithm* Enc *is a deterministic algorithm that takes strings* $K \in \mathcal{K}$ *and* $N \in \mathsf{Nonce}$ *and* $H \in \mathsf{Header}$ *and* $M \in \mathsf{Message}$. *It returns a string* $C = \mathsf{Enc}^{N,H}_K(M) = \mathsf{Enc}_K(N, H, M)$. *Decryption algorithm* Dec *is a deterministic algorithm that takes strings* $K \in \mathcal{K}$ *and* $N \in \mathsf{Nonce}$ *and* $H \in \mathsf{Header}$ *and* $C \in \{0,1\}^*$. *It returns* $\mathsf{Dec}^{N,H}_K(C)$, *which is either a string in* Message *or the distinguished failure symbol* $\perp$. *The correctness of the* AEAD *scheme requires that* $\mathsf{Dec}^{N,H}_K(\mathsf{Enc}^{N,H}_K(M)) = M$ *for all* $K \in \mathcal{K}$ *and* $N \in \mathsf{Nonce}$ *and* $H \in \mathsf{Header}$ *and* $M \in \mathsf{Message}$. *Some linear-time computable length function* ℓ *is given by* $|\mathsf{Enc}^N_K(M)| = \ell(|M|)$.

AEAD PRIV SECURITY: Let $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *be* AEAD*-scheme with length function* ℓ. *Let* $\$(\cdot,\cdot,\cdot)$ *be an oracle that, upon input returns* (N, H, M), *returns a random string of*

$\ell(|M|)$ *bits. The advantage of adversary* $\mathcal{A}$ *in breaking* AEAD PRIV*-security is defined as*

$$\mathsf{Adv}^{\mathsf{PRIV}}_{\mathsf{AEAD},\mathcal{A}}(\lambda) = \Pr\left[K \xleftarrow{\$} \mathcal{K} : \mathcal{A}^{\mathsf{Enc}_K(.,.,.)} \implies 1\right] - \Pr\left[\mathcal{A}^{\$(.,.,.)} \implies 1\right].$$

AEAD AUTH SECURITY: Let $\Pi = (\mathcal{K}, \mathsf{Enc}, \mathsf{Dec})$ *be* AEAD*-scheme, and let* $\mathcal{A}$ *be an adversary with access to an orcale* $\mathsf{Enc}_K(\cdot,\cdot,\cdot)$*. We say that* $\mathcal{A}$ forges *(for this key K and on some particular run) if* $\mathcal{A}$ *outputs* (N, H, C) *where* $\mathsf{Dec}_K^{N,H}(C) \neq \perp$ *and* $\mathcal{A}$ *did not query* $\mathsf{Enc}_K^{N,H}(M)$ *to the encryption oracle. We say that an AEAD scheme* AEAD *is* auth- *secure if for all PPT algorithms* $\mathcal{A}$*,* $\mathsf{Adv}^{\mathsf{auth}}_{\mathsf{AEAD},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ*.*

3.3 Related Work

We first summarize other related work on DTN protocols, BPSec security, and relevant security approaches. Standard definitions that we use in this work for e.g., AEAD and MAC security can be found in Appendix 3.2.

DTN Protocols A variety of DTN protocols for space communications exist and are managed by the Consultative Committee for Space Data Systems (CCSDS) [29], including the Space Packet Protocol [30] and CCSDS File Delivery Protocol (CFDP) [31], but relatively little cryptographic analysis has been done of them. The suite of DTN protocols are focused on a store-and-forward approach to make them more robust to environmental disruptions and message relay issues. Since ground stations may also require substantive planning in the order of days to send messages [32], it is essential that each transmission has the capability of aggregating message information along its path and processing as needed. Space link efficiency is a particular concern for DTN protocols.

A survey of DTN key management protocols [33] reveals that formal cryptographic analysis in the provable security sense is relatively rare. Of the ones that have received analysis in the areas of identity based cryptography, non-interactive key exchange, and group key management [34]–[39], none are documented or promulgated by public institutions such as the IETF or CCSDS as deployed in practice, including in space communications. Other DTN protocols that have known space uses, such as the Licklider Transmission Protocol, lack cryptographic security analysis [40]. Broadly speaking, the lack of cryptographic formal

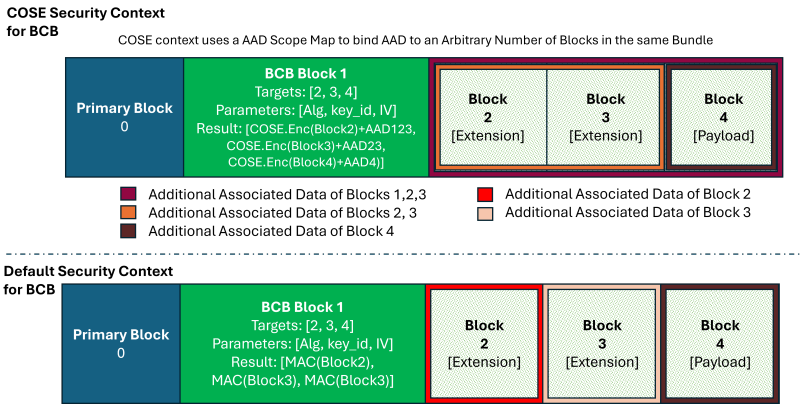


Figure 3.3. COSE Context vs Default Context treatment of AAD

or computational analysis in this field leaves claims of security by various DTN protocols inconclusive, threat models under-defined, and security assumptions on the underlying cryptographic primitives unknown. This presents a gap that we address in this work, providing a first rigorous cryptographic analysis of BPSec.

BPSec COSE Context BPSec's Default Security Context described in [11] was created for interoperability purposes (among space agencies), and provides the minimum level of security based on preshared symmetric keys. In an effort to interoperate with other networks using DTN protocols and achieve compatibility with asymmetric-keyed algorithms, the CBOR Object Signing and Encryption (COSE) Security context was proposed in the DTN IETF working group [41]. This draft standard defines how to incorporate signing and encryption to BPSec using PKI; expands the additional associated data (AAD) to support binding AAD to an arbitrary number of blocks in the same bundle; and introduces PKIX certificate for entity authentication.

Key Wrapping in BPSec BPSec [11] offers the option to incorporate AES Key Wrapping (AESKW) in security blocks, according to the AESKW standards outlined in [42]. In an early request for proposal [43] for ANS X9.102 standard that discuss *key-wrapping* as a primitive, the goal of key wrapping was highlighted as "to protect the confidentiality and integrity of cryptographic keys without the use of nonces". Accordingly, BPSec's use of AESKW aims to enable secure sharing of cryptographic keys used within BIB/BCBs with other nodes who have access to the correct key wrapping keys.

The AESKW wrapping algorithm takes as input a pre-established key encryption key (KEK), a message to encrypt which in this case is a key k, and optional associated data concatenated with a static integrity check vector (ICV) which are passed into a six-round non-standard Feistel-network [28]. This outputs a ciphertext which is sent along to a receiver with the plaintext authenticated data. The unwrapping algorithm takes as input the KEK, ciphertext, ICV and authenticated data and outputs either the shared key or error upon integrity check failure.

A serious caveat with AESWK and other *key-wrap* schemes discussed in ANS X9.102 is that their security has not been formally proven. Rogaway and Shrimpton in [28] likens the primitive to a secure enciphering scheme similar to a strong, variable-input-length pseudo-random permutation. However, they stop short of formally verifying the security of AESWK due to ambiguities regarding its construction in ANS X9.102. Instead, they introduce a novel framework called *deterministic authenticated encryption* (DAE) [28] that is capable of capturing the security goals of *key-wrapping*, which we leverage in the analysis of BPSec and our StrongBPSec improvements in (Section 3.6).

Key Management Key management (key derivation, key exchange, key revocation, key separation) policies are not explicitly defined by any of the three relevant RFCs for BPSec. Instead, it is assumed that these are handled separately as part of network management [12]. RFC9173 [11] stipulates that BP nodes using security contexts need to "establish shared key encryption keys (KEKs) with other nodes in the network using an out-of-band mechanism". BPSec *does not* provide key establishment or entity authentication mechanisms internally.

Symmetric keys were chosen over asymmetric keys (e.g. BIB with HMAC-SHA2) "in order to create a security context that can be used in all networking environments" [10]. RFC9173 stipulates that different keys must be used to perform different security operations (e.g. a separate key for data encryption vs integrity protection) and across different cipher-suites for the same operations (i.e. using a separate key for AES-GCM vs AES-CBC). Depending on how symmetric keys are distributed for a given security context (key exchange, pre-shared, out-of-band), the use of ephemeral keys through AES-Key wrapping or a key-rotation policy must be used to curtail key leakage. Initialization vectors must also not be reused with the same key across multiple encryption operations.

3.4 BPSec Formalization and Strong BPSec

We first present a formalization of notation for BPSec, with variables shown in Table 3.1. A general BPSec block is structured as shown in Figure 3.4.

$\vec{\vec{\ }}$	Global state $\{st_1, ..., st_n\}$ that holds particular key and parameter sets $\{st_i[p] = (\vec{k}_p, p)\}$ used by Party i participating in a BPSec channel.
aad	Authenticated associated data
bid	Block index which identifies the block written as a subscript of $\vec{\mathtt{M}}$, $\vec{\mathtt{F}}$, or $\vec{\mathtt{P}}$: B - Index of the Strong BPSec ledger block $\mathtt{BIB}_\mathsf{B}$ (always the second to last block in our construction to align with [26]) PB - Index of the Primary Block (always 0) PLD - Index of the bundle payload block (always B − 1) tar - Index of security operation's target block
$\vec{\mathtt{C}}$	Resultant array after operating on $\vec{\mathtt{M}}$ with a set of cryptographic operations specified by $\vec{\mathtt{F}}$ and parameterized by $\vec{\mathtt{P}}$.
c_k	Ciphertext of the key wrapped ephemeral symmetric key k
ctr	Counter for tracking index of security blocks
$\vec{\mathtt{F}}$	List of security operations {conf, int} for blocks in a bundle $\vec{\mathtt{M}}$
$\vec{\mathtt{F}}$.type	BPSec block type {"BCB", "BIB"} synonymous with operations {conf, int}
$\vec{\mathtt{F}}$.targets	Security operation targets $\{\mathsf{bid}_0, \ldots, \mathsf{bid}_\mathsf{N}\}$
$\vec{\mathtt{ID}}$	A global set of node identifiers consisting of $\mathsf{id} \in \{\mathsf{id}_1, \ldots, \mathsf{id}_n\}$ to uniquely identify each state st in global state $\vec{\vec{\ }}$
IV	Initialization vector
k	Ephemeral symmetric key (e.g. the key wrapped key)
k_p	Preshared symmetric key accessible to parties with access to $\vec{\vec{\ }}$
$k^{\mathtt{BB}}$	Ephemeral symmetric key for BIB_B authentication (i.e., the key wrapped BIB_B key)
$k_p^{\mathtt{BB}}$	Preshared symmetric key for BIB_B authentication
$k^{\mathtt{RR}}$	Ephemeral symmetric key for BRB authentication (i.e., the key wrapped BRB key)
$k_p^{\mathtt{RR}}$	Preshared symmetric key for BRB authentication
$\vec{\mathtt{M}}$	Plaintext blundle made of blocks $\{\mathsf{bid}_0, \ldots, \mathsf{bid}_\mathsf{n}\}$ where $\mathsf{n} = \lvert\vec{\mathtt{M}}\rvert$ such that $\lvert\vec{\mathtt{M}}\rvert \geq 1$ (mandatory primary and payload blocks)
$\vec{\mathtt{M}}_i.m$	The plaintext data in a block $\vec{\mathtt{M}}_i$
meta	Metadata ledger for all security operations performed by the security source that is authenticated by the $\mathtt{BIB}_\mathsf{B}$
$\vec{\mathtt{P}}$	Sets of ciphersuite parameters associated with security operations $\vec{\mathtt{F}}$ specified by a security context.
$\vec{\mathtt{P}}$.alg	Algorithms specified (e.g. AEAD modes, key generation) for a block.
$\vec{\mathtt{P}}$.id	Node identifier for a party that supports a particular parameter set
$\vec{\mathtt{P}}$.init	T/F flag: T if the security source is the bundle source; F otherwise.
$\vec{\mathtt{P}}$.params	Security parameters used for particular algorithms.
$\vec{\mathtt{P}}$.params.kid	Key identifier
st	The local state of a party within global state $\vec{\vec{\ }}$
st.id	The local state for party id
tar	The target block index: $\vec{\mathtt{M}}_{\mathtt{tar}}$ is the unprotected target block whereas $\vec{\mathtt{C}}_{\mathtt{tar}}$ is the protected target block.
valid	A helper function to check if the security operation $\vec{\mathtt{F}}_i$ conflicts with any existing security operations already applied to a block $\vec{\mathtt{M}}_i$.

Table 3.1. Abstracted variables used to formalize BPSec.

We assume a 1:1:1 relationship between $\vec{\mathtt{M}}$, $\vec{\mathtt{F}}$, $\vec{\mathtt{P}}$ meaning that for each block in a bundle $\vec{\mathtt{M}}$, a BPA $\in \{\mathrm{SN}, \mathrm{IN}\}$ can add a unique security operation $\vec{\mathtt{F}}$ governed by a unique parameter set $\vec{\mathtt{P}}$. The security policy of a BPA may be configured to ignore certain blocks which corresponds to a particular $\vec{\mathtt{F}}_i = \perp$. Otherwise, the BPA either adds plaintext integrity or authenticated encryption operations to the block yielding in a new BIB or BCB, respectively. We designate the BPSec protected bundle as $\vec{\mathtt{C}}$. For authenticity protection, the BPA performs HMAC

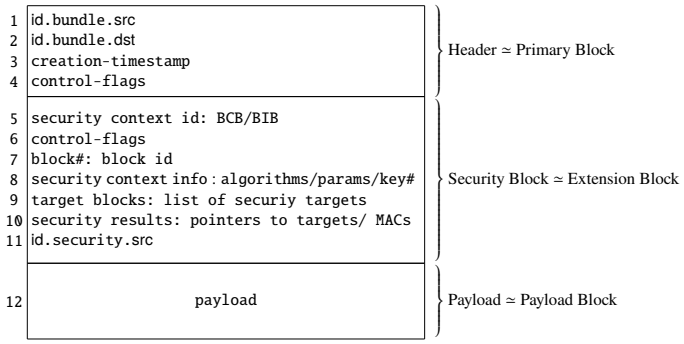


Figure 3.4. Generic BPSec bundle with cryptographically relevant fields.

operations using a secret key to the block and stores the tag, a security result in the BIB. For authenticated encryption, AEAD is applied to the block yielding either separate MAC and ciphertext or single combined result: in cases where the MAC is generated separately, it can be stored in the BCB as a result while the ciphertext replaces the data being encrypted in-place. When multiple blocks are protected by the same security operation using the same parameter set, they are consolidated under a single BIB or BCB respectively as seen in Figure 3.2.

3.4.1 Flexible Secure Channels

A Flexible Secure Channel (FSC) is distinguished by several features from a traditional secure channel. Specifically, an FSC includes not only a sender and receiver but also one or more intermediate nodes which share a common set of keys $\overrightarrow{k}$ and associated parameters $p \in \overrightarrow{\mathcal{P}}$ contained in their individual *states*. For BPSec, the keys are preinstalled symmetric keys and the parameters are dictated by the security context(s) supported by the participant. The FSC global state $\overrightarrow{\varkappa}$ is the union of all local states, st of FSC participants (i.e. $st \in \overrightarrow{\varkappa}$). Without loss of generality, in a symmetric key FSC, such as in BPSec under the Default Security Context, sender state st and receiver state st' must match (i.e. there exists $p \in \overrightarrow{\mathcal{P}}$ such that $st[p] = st'[p]$) to correctly process FSC messages.

Definition 5 (Syntax for flexible secure channels)**.** *A flexible secure channel* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Snd}, \mathsf{Rcv})$ *with associated state space* $\mathcal{S}$ *and error space* $\mathcal{E}$*, where* $\mathcal{E} \cap \{0,1\}^* = \emptyset$*, consists of three efficient algorithms:*

- Init. *On input of a security parameter array* $\vec{\mathtt{P}}$*, this probabilistic algorithm outputs initial state array* $\vec{\mathtt{st}} \in (\mathcal{S} \times \ldots \times \mathcal{S})$*. We write* $(\vec{\mathtt{st}}) \xleftarrow{\$} \mathsf{Ch.Init}(\vec{\mathtt{P}})$*. We note that global state* $\vec{\mathtt{st}}$ *constitute many states st, where a state st matches with multiple other states* $st' \in \{st_1, \ldots, st_n\}$*.*
- Snd. *On input of state* $st \in \vec{\mathtt{st}}$*, a message bundle* $\vec{\mathtt{M}} : |\vec{\mathtt{M}}| \in \mathbb{N}$ *and* $\forall m \in \vec{\mathtt{M}}, m \in \{0,1\}^*$*, a security flag array* $\vec{\mathtt{F}}$*, and a security parameter array* $\vec{\mathtt{P}}$*, this (possibly) probabilistic algorithm outputs an updated state array* $st' \in \vec{\mathtt{st}}$ *and a secured bundle* $\vec{\mathtt{C}} : |\vec{\mathtt{C}}| \in \mathbb{N}$ *and* $\forall c \in \vec{\mathtt{C}}, c \in \{0,1\}^*$*. We write* $(st', \vec{\mathtt{C}} \xleftarrow{\$} \mathsf{Snd}(st, \vec{\mathtt{M}}, \vec{\mathtt{F}}, \vec{\mathtt{P}}))$*.*
- Rcv. *On input of a state* $st \in \vec{\mathtt{st}}$ *and a secured bundle* $\vec{\mathtt{C}}$*, this deterministic algorithm outputs an updated state* $st' \in \vec{\mathtt{st}}$ *and a message bundle* $\vec{\mathtt{M}} : \forall\, m \in \vec{\mathtt{M}}$ *where* $m \in (\{0,1\}^* \cup \mathcal{E})$*. We write* $(st', \vec{\mathtt{M}} \leftarrow \mathsf{Rcv}(st, \vec{\mathtt{C}}))$*.*

Definition 6 (Flexible secure channel correctness)**.** *Let* $\mathsf{Ch} = (\mathsf{Init}, \mathsf{Snd}, \mathsf{Rcv})$ *be a flexible secure channel. We say that* Ch *provides correctness if for all state pairs* $(st_i, st_j) \in \vec{\mathtt{st}} \xleftarrow{\$} \mathsf{Ch.Init}(\vec{\mathtt{P}})$*, for all messages* $(m_0, \ldots, m_\ell) : \vec{\mathtt{M}} = \{m_0, \ldots, m_\ell\}$ *(where* $\ell \in \mathbb{N}$*) , for all flags* $(f_0, \ldots, f_\ell) : \vec{\mathtt{F}} = \{f_0, \ldots, f_\ell\}$*, for all parameters* $(p_0, \ldots, p_\ell) : \vec{\mathtt{P}} = \{p_0, \ldots, p_\ell\}$ *and for all message arrays* $\vec{\mathtt{M}}' \leftarrow \mathsf{Rcv}(st_j, \mathsf{Snd}(st_i, \vec{\mathtt{M}}, \vec{\mathtt{F}}, \vec{\mathtt{P}}))$*, we have that*

$$(m_0, \ldots, m_\ell) \in \vec{\mathtt{M}}' \iff \forall p_r \in \{p_0, \ldots, p_\ell\} : i, j \in p_r.id \ .$$

Thus, for a block to be processed correctly by a receiver, the receiver must support the particular parameter set embedded in the associated states for each of the message blocks in the bundle. Additionally, since an intermediate receiving node may not share a matching state (with appropriate parameter sets) for all security blocks in a bundle $\vec{\mathtt{C}}$*, the node may legitimately process only the blocks for which it shares the correct state. This* partial processing *is additionally captured as correct behavior via our definition.*

In our protocol construction for BPSec and StrongBPSec illustrated in Figure 3.7 we leverage the flexibility and modularity of our formalism for flexible secure channels. Notably, within the BPSec construction, the intermediary nodes process bundles differently to source and destination nodes, who only Snd and Rcv bundles respectively. In contrast, intermediary nodes both Snd and Rcv bundles; they Snd as they add new BCB/BIB blocks to a bundle and forward it to the next node; they Rcv bundles forwarded to them within which they

process and discard BIB/BCB blocks and add BRB read receipts. The versatility of our flexible channel design successfully captures all these use cases, enabling the formalization of uncommon constructions such as BPSec.

We next provide an intuition for StrongBPSec, before formalizing the construction of both BPSec and StrongBPSec, with a side-by-side clarification of differences, in Figure 3.7.

3.4.2 StrongBPSec with Read Receipts

As noted, the current BPSec specification [12] provides no security guarantees against arbitrary message dropping by an attacker (privileged or not) during the transit of a bundle between its source and the final destination. Particularly, an inherent feature of BPSec is to allow honest intermediary nodes to process and discard bundle blocks as governed by their internal policies with the exception of primary and payload blocks. However, BPSec [12] does not require intermediate nodes to provide verifiable evidence of identity in order to process or discard security blocks. Neither does the bundle maintain a record of changes made to it on way to its destination. Thus, an unprivileged attacker can arbitrarily and selectively drop blocks from a bundle en route without being detected. The proposed COSE Security Context for BPSec [41], as part of its integration into BPSec, includes a recommendation to increase the integrity of bundle blocks. In order to provide stronger integrity guarantees, they propose binding an arbitrary number of blocks together in the same bundle, using additional associated data (`aad`) for each block concerned. However, we argue that binding `aad` to an arbitrary number of blocks in this manner also restricts the ability of an honest intermediary to process and discard any individual block from such a group. In an attempt to find a fair middle ground between the somewhat relaxed original functionality goals of BPSec [12] and the overly rigid security guarantees of COSE [41], we propose our StrongBPSec protocol with *read receipts*.

StrongBPSec introduces two additional layers of integrity that aims to guarantee to the bundle destination one of two of the following:

1. All blocks added by the bundle source node SN has arrived unchanged at the bundle destination node DN, or
2. Some honest intermediary node IN has correctly processed and discarded block/s that were constructed by the bundle source SN.

Note that, for clarity, we will be exclusively distinguishing the roles of BPA as either SN, IN or DN for the rest of this discussion.

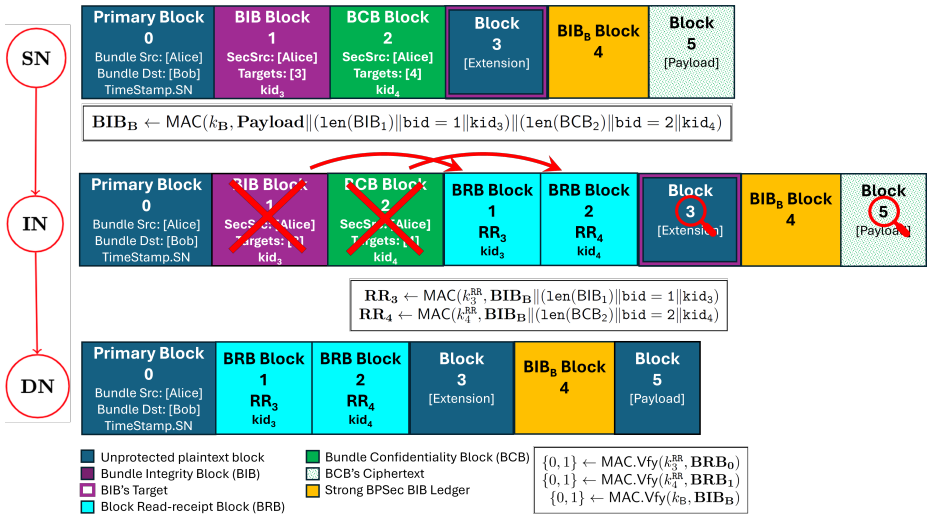


Figure 3.5. StrongBPSec with Read Receipts. Source integrity block MAC $\mathsf{BIB_B}$ is calculated over the payload and metadata for each target/security block ($\mathsf{BIB_1}$ and $\mathsf{BCB_2}$) added by SN. Read receipts BRB are calculated by INs every time they successfully process a security block added by SN, recording and verifying the details of the blocks they process. INs replace any such security blocks they process with a corresponding BRB before forwarding the bundle. DN first verifies any BRBs added by INs followed by the verification of $\mathsf{BIB_B}$.

We achieve these stronger integrity guarantees through the addition of two additional checks leveraging the already existing BIB format of BPSec. As illustrated in Figure 3.5, we introduce a $\mathsf{BIB_B}$ block, calculated and prepended to the payload block by the original source node SN. As the originator of the bundle, SN constructs $\mathsf{BIB_B}$, which includes a MAC tag calculated over the payload block and a set of uniquely identifying details for each security block added by SN. These identifying details include unique key/block identifiers (kid/bid), and the number of target blocks len(BIB/BCB) covered by a security block (BIB/BCB). Once constructed, this $\mathsf{BIB_B}$ is only verified by the destination node DN. [4] The successful verification of $\mathsf{BIB_B}$ with the corresponding key k^{BB}, which is either a pre-shared or a key-wrapped key, informs DN one of two things; either they have received the exact same

[4] We note that we only consider the use case for DN for simplicity but WLOG we say that any *security acceptor* authorized by the security policy may process $\mathsf{BIB_B}$.

bundle SN constructed; or an honest intermediary IN has processed and discarded some of the blocks but have replaced them with correct read receipts BRB. These BRB blocks duplicate the unique identifying details of a discarded block authenticated with MAC tag, which we discuss next. In line with the BPSec specification, we assume honest privileged intermediary nodes.

The second type of integrity check introduced by our StrongBPSec construction is the concept of a *read receipt* BRB. The addition of BRB allows an honest intermediary acting as a security acceptor to process and discard any block added by SN but still maintain a verifiable record of their details that was used in the calculation of BIB_B. To clarify, recall from Section 3.1.3, a read receipt BRB provides an in-band ledger of changes for processing at the DN; it should not be assumed that the BRB gets sent back to the SN as some sort of acknowledgment of receipt or changes as would be done by a "return-receipt" from a Bundle Status Report [27].

A BRB block contains a plaintext and a MAC tag calculated over it. The plaintext duplicates the identifying details of the corresponding discarded security block $(\mathsf{kid/bid}, \mathtt{len}(\mathsf{BIB/BCB}))$ that was used in the calculation of BIB_B, without which the verification of BIB_B at DN will fail. A MAC tag is then calculated using a unique key k^{RR}, which is either a pre-shared or a key-wrapped key, over the bundle BIB_B concatenated with this plaintext. These BRB blocks act as a transcript of the original bundle to the DN, who verifies all BRB blocks prior to the verification of BIB_B. The successful verification of BRB blocks within a received bundle informs DN that only honest intermediaries have discarded blocks from the original bundle. In Figure 3.6 we illustrate an expected execution of our StrongBPSec formalism, which we describe in detail in Figure 3.7.

While significant security gains could be achieved with inclusion of digital signatures in this improved protocol (in particular preventing impersonation within a group sharing the same parameter set and keys), the BPSec infrastructure does not assume asymmetric key management; pre-shared symmetric keys were favored for simplicity and broad implementation. Thus, in strengthening BPSec, we inherit the original key infrastructure and primitives assumed by the Default Security Context described in RFC9173 [11]. Thus, it is not possible to detect which IN has edited the bundle nor is it possible to protect against an insider threat. These issues are inherent to RFC9173 in absence of asymmetric key management.

Figure 3.7 shows the formal construction of BPSec under RFC9173 alongside our StrongBPSec. The construction abstracts details of BPSec [12] and its Default Security Context [11] into a FSC protocol (see Definition 5) with modifications needed for a StrongBPSec protocol. Below we explain the intuition behind our formal construction.

- Init generates symmetric keys for matching BPSec states per parameter set. Parameters in a state are associated with a collection of party identifiers $\vec{\mathsf{id}}$ who all maintain the same symmetric keys. Init generates three keys per parameter set: a symmetric key k_p; a read receipt key k_p^{RR} and; a bundle verifier key k_p^{BB}.
- Snd creates security blocks either by bundle source or any intermediate node. For each message $m \in \vec{\mathtt{M}}$, Snd checks that the security operation is valid, and key wraps a new symmetric key if necessary. Snd adds the appropriate processing information to the security block, and either authenticates or encrypts the message m according to the security operation. In StrongBPSec, Snd also adds a meta-data array for its ledger block, which it authenticates, creating a verifiable ledger.
- Rcv processes security blocks either by bundle destination or any intermediate node. For each ciphertext $c \in \vec{\mathsf{C}}$, Rcv checks that the security acceptor has the correct keys to process the ciphertext, and key-wraps a new read-receipt key if necessary as intermediate node. Rcv adds read-receipts after processing the security block, and authenticates the associated meta-data. In StrongBPSec, the destination node also verifies all read-receipts, and constructs a meta-data array for the ledger block and verifies the $\mathsf{BIB}_{\mathsf{B}}$, rejecting the payload if any checks fail. Relevant keys are either extracted from state or decrypted from *keywrapped* ciphertexts.

We formally prove the respective security of BPSec and StrongBPSec in Section 3.6.

3.5 Flexible Secure Channel Models

In Figure 3.8 we formalize the security experiment for Flexible Secure Channels (FSC) that captures the BPSec security goals described by [12]. Furthermore, in Figure 3.8 we also capture the intended security goals added by StrongBPSec, namely through Strong FSC Security (SFSC) security, highlighting these as additional steps.

In the FSC experiment, the challenger begins by generating the full secret state for all parties, and randomly sampling a bit b. The adversary is then split into two phases. First, in

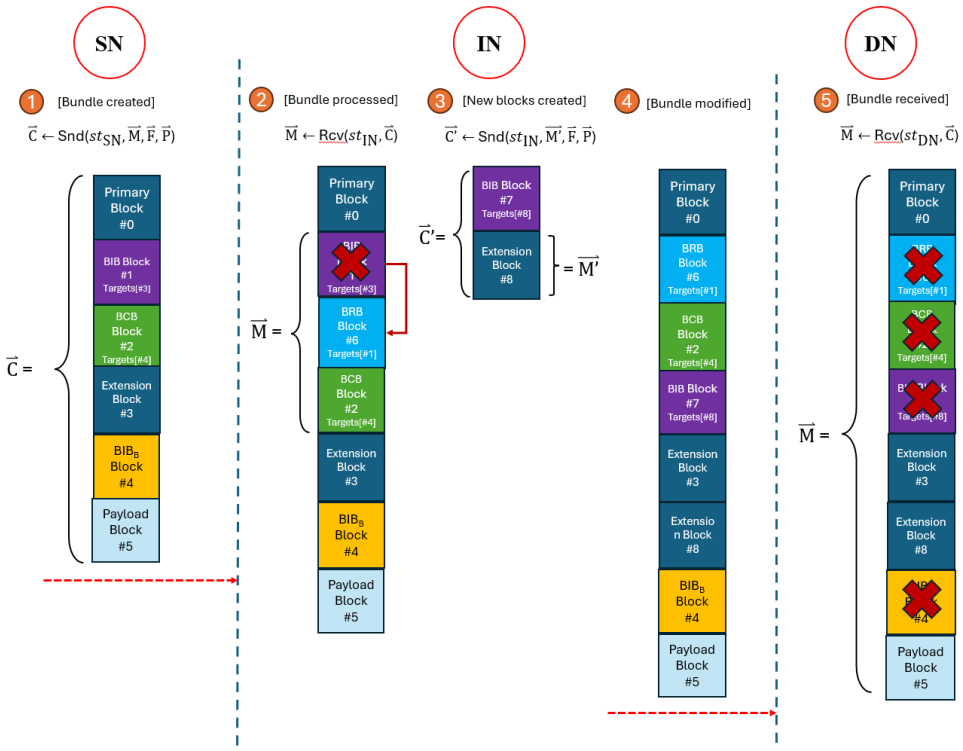


Figure 3.6. Flow of interactions during an expected execution of StrongBPSec. SN creates a bundle and forwards it to IN. IN processes and replaces BIB block #1 with read receipt block BRB #6. IN further adds new blocks #7 and #8 and forwards the modified bundle to DN. DN processes and discards all remaining security blocks and extracts the bundle message $\vec{M}$. All processed and discarded blocks are crossed out in red. The values for $\vec{P}$ and $\vec{F}$ are selected from respective sets which can be found in Table 3.1.

the Corrupt phase, the $\mathcal{A}$ is allowed to issue Corrupt queries for particular parameter sets[5]. In the next phase, $\mathcal{A}$ outputs some state, which is given as input to the next adversary $\mathcal{A}'$, which allows us to capture static corruption adversaries.

We argue that proving the security of our construction against a static (as opposed to adaptive) $\mathcal{A}$ is sufficient for the following reasons: each BPSec block is associated with a unique set of independent security parameters that remain unchanged for the duration of the message lifetime; there is no forward secrecy and it does not make a difference at what point

[5]This describes how $\mathcal{A}$ becomes a dishonest privileged node.

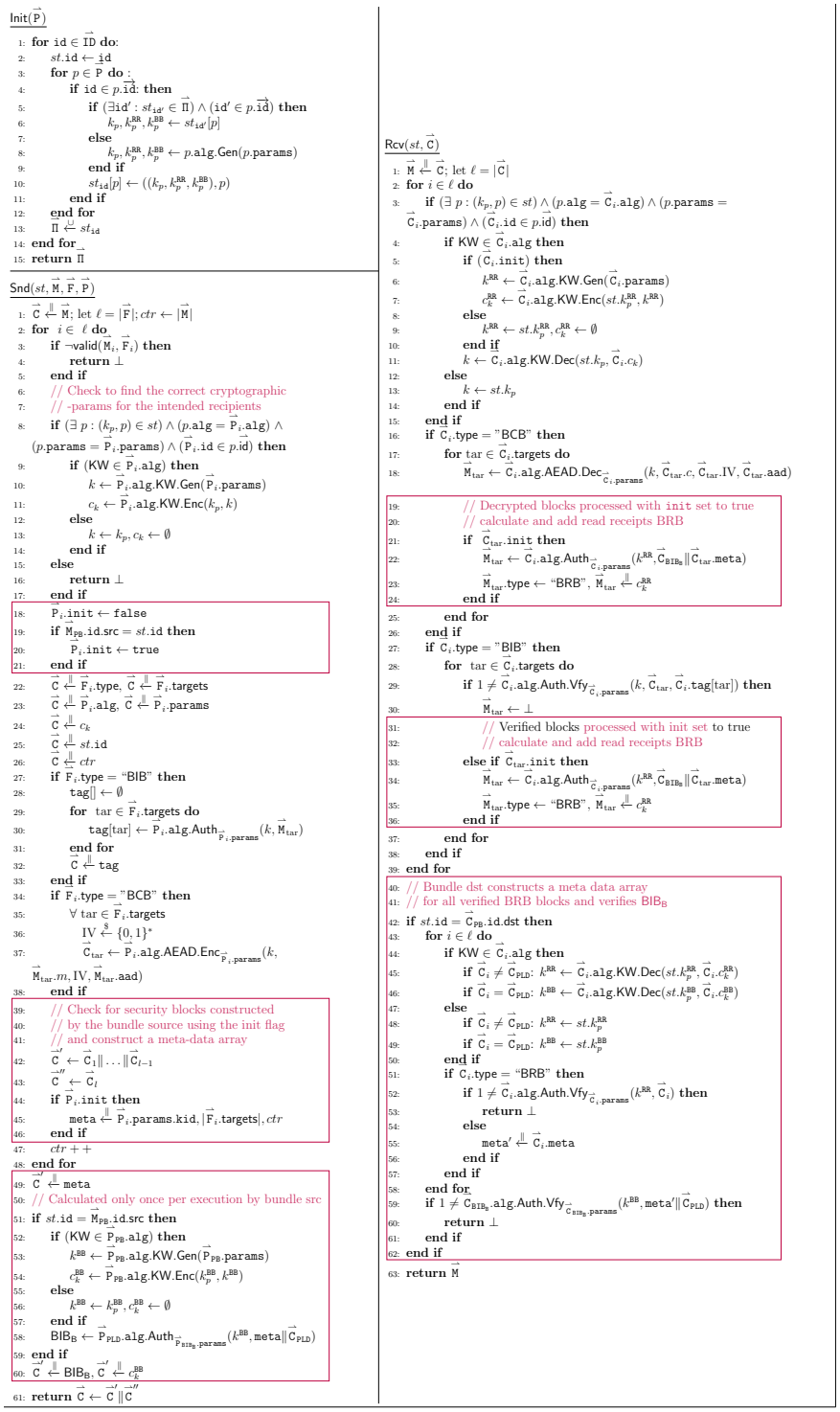


Figure 3.7. BPSec protocol construction. The additions needed for the StrongBPSec protocol are highlighted in boxes. Refer to Table 3.1 for notational and functional definitions. Additional notational clarifications: $X \xleftarrow{\|} Y$ denotes concatenating Y to X; $T[j]$ denotes the value accessed by key j in table T, which we assign no particular structure; subscripts are generally used for array indices or but are also used to associate variables (e.g. k_p to denote a key k associated with p) or to configure a function call (e.g. $\text{Auth}_{\vec{C}_i.\text{params}}$); superscripts are used to classify a variable type (e.g. k^{RR} refers to a read-receipt key); p.alg.Gen refers to invoking a key generation function from a family of algorithms such as Key-Wrap (KW) in p but we also overload alg as an identifier (e.g. used for boolean comparison).

of a message's lifetime $\mathcal{A}$ compromises these security parameters as they do not update; and adversary $\mathcal{A}$ compromising of set of security parameters for a specific block does not affect the security of any other blocks in the same bundle. $\mathcal{A}$ now has access to two oracle queries: $\mathcal{O}$Snd and $\mathcal{O}$Rcv:

- $\mathcal{O}\mathsf{Snd}(i, \vec{\mathtt{M}}, \vec{\mathtt{F}}, \vec{\mathtt{P}}) \rightarrow \vec{\mathtt{C}}$: allows the adversary to indicate that party i should protect the plaintext bundle $\vec{\mathtt{M}}$ using security operations $\vec{\mathtt{F}}$ using security parameters $\vec{\mathtt{P}}$. If the bit b sampled by the challenger is 1, the associated parameters have not been corrupted, and the security operation is encryption (signified by $\mathsf{type} = \mathsf{BCB}$) then the challenger records and replaces the associated ciphertext with random strings from the same length. $\mathcal{O}$Snd also records all integrity protected blocks (signified by $\mathsf{type} = \mathsf{BIB}$), and their associated tags in the AUTH register (to detect integrity wins later in the experiment).
- $\mathcal{O}\mathsf{Rcv}(i, \vec{\mathtt{C}}) \rightarrow \vec{\mathtt{M}}$: allows the adversary to indicate that party i should process the ciphertext $\vec{\mathtt{C}}$. $\mathcal{O}$Rcv also checks if the adversary has caused any win events to trigger. Specifically, if the associated security parameters are not corrupted, and the party i outputs a valid block which no honest party produced, then the adversary has forged this block, and the challenger sets $\mathtt{win} \leftarrow \mathsf{true}$.

At some point the adversary terminates and outputs a bit b', and the output of the experiment is $(b = b') \vee \mathtt{win}$. We say an adversary $\mathcal{A}$ wins the FSC security game if they succeed in achieving one of the following: forges an integrity-protected block; forges an authenticated ciphertext within a block or; distinguishes the real-or-random ciphertext within a block. Below we briefly summarize additional notations used in our security experiment illustrated in Figure 3.8 for our security experiment.

- AUTH: A register for honestly authenticated messages and their MAC tags.
- CORR: A register used to track corrupted parameters for determining wins.
- CTXT: A register that maintains either real ciphertext and associated parameter pairs or uniformly random strings from the same space.
- Extr: A helper function to extract a parameter set from a block and state.
- Proc: A binary function that checks if a ciphertext bundle $\vec{\mathtt{C}}$ can be processed.
- tsid: A state identifier that identifies the parameter set used by source.
- TXT:A register for all messages accessible by the global state $\vec{\ }$.

We now formally define FSC security.

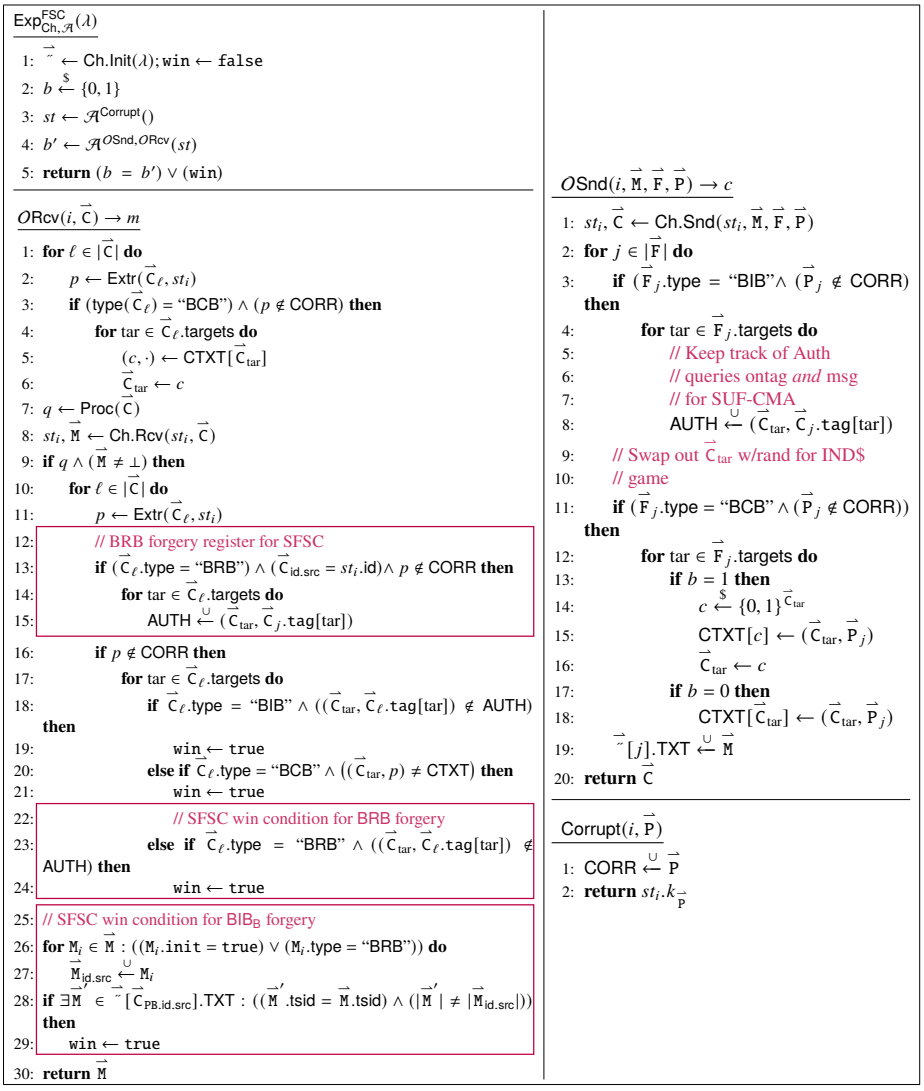


Figure 3.8. Security Definition for Flexible Secure Channels (FSC). The modifications needed for SFSC experiment for the StrongBPSec with Read Receipts is presented in boxes. Refer to Table 3.1 and Section 3.5 for notational definitions.

Definition 7 (FSC Security)**.** *Let* Ch *be a channel protocol* $\mathsf{Ch} = \{\mathsf{Init}, \mathsf{Snd}, \mathsf{Rcv}\}$. *Let* $\mathsf{Exp}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *be the* FSC *security experiment without red-boxed lines defined in Figure 3.8. We define the advantage* $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *that the adversary* $\mathcal{A}$ *wins the* FSC *game as*

$\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = |2 \cdot \Pr(\mathsf{Exp}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = 1) - 1|$. *We say that* Ch *is* FSC*-secure if* $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ.

Next, we turn to defining the Strong FSC game (SFSC), which adds the boxes in Figure 3.8. The SFSC game adds two additional win conditions in the Rcv query, allowing the adversary to win by dropping $\vec{\mathsf{C}}$ blocks without being detected, or by forging so-called read receipts, which indicate to the receiving party that an honest party has "processed" a block from the sender. As before, the adversary at some point will terminate and outputs a bit b' and the output of the experiment is $(b = b') \vee \texttt{win}$. Thus, in addition to the FSC win conditions, the SFSC adversary $\mathcal{A}$ wins if it can forge either BRB or $\mathsf{BIB}_{\mathsf{B}}$ with non-negligible probability. We now formally define SFSC security.

Definition 8 (SFSC Security)**.** *Let* Ch *be a channel protocol* $\mathsf{Ch} = \{\mathsf{Init}, \mathsf{Snd}, \mathsf{Rcv}\}$. *Let* $\mathsf{Exp}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *be the* SFSC *security experiment defined in Figure 3.8 (with all lines included). We define the advantage* $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *that the adversary* $\mathcal{A}$ *wins the* SFSC *game as* $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = |2 \cdot \Pr(\mathsf{Exp}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda) = 1) - 1|$. *We say that* Ch *is* SFSC*-secure if* $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{Ch},\mathcal{A}}(\lambda)$ *is negligible in the security parameter* λ.

3.6 Security Analysis

In this section, we provide a security analysis of the BPSec construction given in Figure 3.7 under Flexible Secure Channel security described in Definition 7. Next, we provide a security analysis of the StrongBPSec construction given in Figure 3.4 under Strong FSC security in Definition 8. We begin with our analysis of BPSec.

Theorem 1 (FSC Security for BPSec)**.** *Let* n_a *be the total number of parameter sets used in the experiment, and let* n_m *be the total number of key-wraps keys in the experiment. The* BPSec *protocol presented in Figure 3.7 is* FSC*-secure. That is, for any PPT algorithm* $\mathcal{A}$ *against the* FSC *security experiment (described in Definition 7)* $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A}}(\lambda)$ *is negligible under the* sufcma*,* aead*, and* dae *security of the* MAC*,* AEAD*,* DAE *primitives, respectively.*

Proof. We note that in the base FSC game there are three main ways the $\mathcal{A}$ can win. First, by forging a BIB tag, secondly by forging a BCB ciphertext, and thirdly by guessing the bit b. Thus we divide our proof into three cases: in the first case the adversary has caused

$\texttt{win} \leftarrow \mathsf{true}$ by generating a forged MAC tag, in the second case the adversary has caused $\texttt{win} \leftarrow \mathsf{true}$ by forging an AEAD ciphertext, and finally the third case the adversary guesses the challenger bit b. In each case we upper-bound the probability of the adversary causing the winning event to occur, and thus prove that the BPSec construction is FSC secure. We thus split our analysis into the following cases:

- C_1 : $\mathcal{A}$ sets $\texttt{win} \leftarrow \mathsf{true}$ when a party π^i accepts $(\vec{\texttt{M}}'_t, \vec{\texttt{C}}_j.\texttt{tag}[t])$ such that $\vec{\texttt{M}}'_t \neq \perp$, but $(\vec{\texttt{M}}'_t, \vec{\texttt{C}}_j.\texttt{tag}[t]) \notin \mathsf{AUTH}$ for $\vec{\texttt{C}}_j.\mathsf{type} = \mathsf{BIB}$ and $\mathsf{Corrupt}(\cdot, \vec{\texttt{C}}_j.\vec{\texttt{P}})$ was not issued, i.e. $\mathcal{A}$ has successfully forged a valid message MAC tag pair for a BIB that verifies correctly.
- C_2: $\mathcal{A}$ sets $\texttt{win} \leftarrow \mathsf{true}$ when a party π^i accepts $\vec{\texttt{C}}$ such that $\vec{\texttt{M}}_t \neq \perp$, $\vec{\texttt{C}}_t.\mathsf{type} = \mathsf{BCB}$, $p \leftarrow \mathsf{Extr}(\vec{\texttt{C}}_t, \mathsf{st}_i)$ and $\mathsf{Corrupt}(\cdot, \vec{\texttt{C}}_j, \vec{\texttt{P}})$ was not issued, i.e. $\mathcal{A}$ has successfully forged a valid AEAD ciphertext.
- C_3 $\mathcal{A}$ does not set $\texttt{win} \leftarrow \mathsf{true}$, and has terminated the experiment and output a bit b', i.e. $\mathcal{A}$ has guessed the challenge bit b.

We proceed via a sequence of games with each game focusing on a specific win condition (or lack thereof) in the three cases. We bound the difference in the adversary's advantage in each game with the underlying cryptographic assumptions until the adversary reaches a game where the advantage of that game equals 0, which shows that adversary $\mathcal{A}$ cannot win with non-negligible advantage.

Case 1 : π^i **accepts a forged message tag pair for a** BIB.

Game 0 This is the initial FSC security game. Thus $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 In this game, for any key-wrapping keys associated with parameter sets that have not been Corrupted, we abort if $\mathcal{A}$ forges a DAE ciphertext, and replace honestly generated key-wrapped keys with uniformly random values. To do so, by a hybrid argument we introduce a series of reductions $\mathcal{B}_1$ as follows: At the beginning of the experiment, $\mathcal{B}_1$ will initialize a DAE challenger $C_{\mathsf{DAE},i}$ for each parameter set $\vec{\texttt{P}}_i$ such that $\mathsf{Corrupt}(\cdot, \vec{\texttt{P}}_i)$ has not been issued (where i is the index into the hybrid argument). Whenever the challenger is required to key wrap ephemeral key k' using k_i, the challenger instead queries $(\emptyset, k')$ to the associated $C_{\mathsf{DAE},i}$ and replaces the honestly generated key-wrap ciphertext with the

output from $C_{\mathsf{DAE},i}$. Additionally, each time $\mathcal{B}_1$ generates a ciphertext from $C_{\mathsf{DAE},i}$, they maintain a table $\mathsf{DAE}[i] \leftarrow (C, k')$ which they update with each honestly generated key-wrap ciphertext. Whenever $\mathcal{B}_1$ is required to decrypt a DAE ciphertext using the key-wrap key associated with parameter set $\vec{\mathtt{P}}_i$, they first check if $(C, k') \in \mathsf{DAE}[i]$. If so, they use k' as the key-wrapped key. If not, they submit $(\emptyset, C)$ to $C_{\mathsf{DAE},i}$ challenger.

Note that by definition of the DAE security game, if $\mathcal{A}$ forges a DAE ciphertext and the bit b sampled by DAE is 0, then DAE will return the correct decryption of C, otherwise DAE returns $\perp$. Thus, any adversary that can forge a DAE ciphertext can be used to break the security of the DAE game. Additionally, we note that if the bit b sampled by $C_{\mathsf{DAE},i}$ is 0, then the $C_{\mathsf{DAE},i}$ encrypts the ephemeral key-wrapped keys k' honestly, and we are in **Game 0**. Otherwise, if the bit b sampled by the $C_{\mathsf{DAE},i}$ is 1, then $C_{\mathsf{DAE},i}$ simply samples a ciphertext uniformly at random, and now the key k' is uniformly random and completely independent of the protocol flow and we are in **Game 1**. Any $\mathcal{A}$ that could distinguish this change can be directly used to break the DAE security of the DAE primitive.

We note that $C_{\mathsf{DAE},i}$ samples keys identically to the security experiment, and thus this replacement is sound. Finally, we note that $\mathcal{A}$ can never later call $\mathsf{Corrupt}(\cdot, \vec{\mathtt{P}}_i)$, so all internal values to DAE will never need to be revealed. As a result of these changes we know that any key-wrap output from a parameter set that has not been Corrupted is uniformly random and independent of the protocol flow, and $\mathcal{A}$ has no information on this key. There are at most n_a total parameter sets and thus n_a hybrid arguments and thus we find: $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda)$.

Game 2 In this game, we introduce an abort event that triggers if any un-Corrupted party "accepts" any of the MAC tags $\sigma_t \in \vec{\mathtt{C}}_j.\mathtt{tag}[t]$ for $t \in \vec{\mathtt{C}}_j.\mathsf{targets}$ of a block $j \in |\vec{\mathtt{C}}|$ such that $\vec{\mathtt{C}}_j.\mathsf{type} = \mathsf{BIB}$, $\vec{\mathtt{M}}_t \neq \perp$, and no honest party generated a message $\vec{\mathtt{M}}_t$. Specifically, this occurs if $\mathcal{A}$ is able to successfully forge BIB using a key associated with a parameter set $p = \mathsf{Extr}(\vec{\mathtt{C}}_t, \mathsf{st}_i)$ such that $\mathsf{Corrupt}(\cdot, p)$ was not issued. We do so by introducing a series of reductions $\mathcal{B}_2$ as follows: At the beginning of the experiment, for each parameter set $\vec{\mathtt{P}}_i$ that does not support key-wrapping $\mathcal{B}_2$ initialises a MAC challenger $C_{\mathsf{sufcma},i}$. Additionally, during the experiment if a DAE ciphertext is generated or decrypted using k_{p_i} (where $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p_i)$), $\mathcal{B}_2$ initialises a MAC challenger $C_{\mathsf{sufcma},i}$. Next, whenever $\mathcal{B}_2$ is required to generate a MAC tag over a message m using k_{p_i} (or the ephemeral key-

wrapped key k' encrypted under k_{p_i}) for p_i, $\mathcal{B}_2$ instead queries m to $C_{\mathsf{sufcma},i}$. If $\vec{\mathtt{C}}_t$ verifies correctly but was not produced by an honest security source, then $(\vec{\mathtt{M}}_t, \sigma_t) \notin \mathsf{AUTH}$ and $\mathcal{A}$ has broken the sufcma security of the MAC. We note that by **Game 1** and the definition of the security experiment, all MAC keys replaced in this way were already uniformly random and independent of the protocol flow, and $\mathcal{A}$ cannot learn these by issuing a Corrupt query. We have at most n_a parameter sets that may be used directly as MAC keys, and at most n_m key-wrapped MAC keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$.

Case 1 Analysis: By definition of **Case 1** we know that $\mathcal{A}$ has caused $\mathtt{win} \leftarrow \mathtt{true}$ by forcing some party π^i to accept a message block $\vec{\mathtt{M}}_t$ such that $\vec{\mathtt{C}}_t.\mathsf{type} = \mathsf{BIB}, \mathsf{Corrupt}(\cdot, \mathsf{Extr}(\vec{\mathtt{C}}_t, st_i))$ has not been issued. By **Game 1** we replaced the ephemeral key k' used to generate the BIB if p is a key-wrapping algorithm. By **Game 2** we added an abort query that prevents $\mathcal{A}$ from forging MAC tags and thus BIB blocks. Since we abort if the adversary forges their own valid BIB, by **Game 2** π^i aborts and thus $\mathcal{A}$ cannot win the game. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A},C_1}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$ We now proceed to **Case 2**.

Case 2 : π^i **accepts a forged message, ciphertext pair for a** BCB

Game 0 This is the initial FSC security game. Thus $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A},C_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 This game proceeds identically to **Game 1** of **Case 1**, with the same reductions and bounds. Thus we have $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_3}(\lambda)$.

Game 2 In this game, we introduce an abort event that triggers if any un-Corrupted party "accepts" a ciphertext block $\vec{\mathtt{C}}_t$ such that $\vec{\mathtt{C}}_t.\mathsf{type} = \mathsf{BCB}$, $\vec{\mathtt{M}}_t \neq \perp$, and no honest party generated a message $\vec{\mathtt{M}}_t$. Specifically, this occurs if $\mathcal{A}$ is able to successfully forge BCB using a key associated with a parameter set $p = \mathsf{Extr}(\vec{\mathtt{C}}_t, \mathsf{st}_i)$ such that $\mathsf{Corrupt}(\cdot, p)$ was not issued. We do so by introducing a series of reductions $\mathcal{B}_4$ as follows: At the beginning of the experiment, for each parameter set $\vec{\mathtt{P}}_i$ that does not support key-wrapping, $\mathcal{B}_4$ initialises an auth-security AEAD challenger $C_{\mathsf{auth},i}$. Additionally, during the experiment if a DAE ciphertext is generated or decrypted using k_{p_i} (where $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p_i)$), $\mathcal{B}_4$ initialises an auth AEAD challenger $C_{\mathsf{auth},i}$.

Next, whenever $\mathcal{B}_4$ is required to generate a ciphertext over a message, nonce and header tuple (m, N, H) using k_{p_i} (or the ephemeral key-wrapped key k' encrypted under k_{p_i}) for p_i, $\mathcal{B}_4$ instead queries (m, N, H) to $C_{\mathsf{auth},i}$'s encryption oracle. Similarly, if $\mathcal{B}_4$ is required to decrypt a ciphertext, nonce, header tuple (C, N, H) for k_{p_i} (or the ephemeral key-wrapped key k' encrypted under k_{p_i}) for parameter set p_i, $\mathcal{B}_4$ instead queries (C, N, H) to $C_{\mathsf{auth},i}$'s decryption oracle. If $\vec{\mathsf{C}}_t$ decrypts correctly but C was not produced by an honest security source, then $(\vec{\mathsf{C}}_t, p_i) \notin \mathsf{CTXT}$ and $\mathcal{A}$ has broken the auth security of the AEAD scheme. Submitting (C, H, N) to $C_{\mathsf{auth},i}$'s decryption oracle then allows $\mathcal{B}_4$ to win the auth-security game. We note that by **Game 1** and the definition of the security experiment, all AEAD keys replaced in this way were already uniformly random and independent of the protocol flow, and $\mathcal{A}$ cannot learn these by issuing a $\mathsf{Corrupt}$ query. We have at most n_a parameter sets that may be used directly as AEAD keys, and at most n_m key-wrapped AEAD keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{auth}}_{\mathcal{B}_4,\mathsf{AEAD}}(\lambda)$.

Case 2 Analysis: By definition of **Case 2** we know that $\mathcal{A}$ has caused $\mathtt{win} \leftarrow \mathtt{true}$ by forcing some party π^i to accept a message block $\vec{\mathsf{C}}_t$ such that $\vec{\mathsf{C}}_t.\mathsf{type} = \mathsf{BCB}$, and $\mathsf{Corrupt}(\cdot, \mathsf{Extr}(\vec{\mathsf{C}}_t, st_i))$ has not been issued. By **Game 1** we replaced the ephemeral key k' used to generate the BCB if p is a key-wrapping algorithm. By **Game 2** we added an abort query that prevents $\mathcal{A}$ from forging ciphertexts and thus BCB blocks. Since we abort if the adversary forges their own valid BCB, by **Game 2** π^i aborts and thus $\mathcal{A}$ cannot win the game. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A},C_2}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{auth}}_{\mathcal{B}_2,\mathsf{AEAD}}(\lambda)$ We proceed to **Case 3**.

Case 3 : **$\mathcal{A}$ has terminated the experiment and output a bit b', i.e. $\mathcal{A}$ has guessed the challenge bit b.**

Game 0 This is the initial FSC security game. Thus $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A},C_3}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 This game proceeds identically to **Game 1** of **Case 1**, with the same reductions and bounds. Thus we have $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_5}(\lambda)$.

Game 2 In this game, all ciphertexts produced by un-$\mathsf{Corrupt}$ed party generating a ciphertext block $\vec{\mathsf{C}}_t$ such that $\vec{\mathsf{C}}_t.\mathsf{type} = \mathsf{BCB}$, using a key associated with a parameter set $p = \mathsf{Extr}(\vec{\mathsf{C}}_t, \mathsf{st}_i)$ such that $\mathsf{Corrupt}(\cdot, p)$ was not issued are replaced with uniformly random

and independent strings. We do so by introducing a series of reductions $\mathcal{B}_6$ as follows: At the beginning of the experiment, for each parameter set $\vec{\mathsf{P}}_i$ that does not support key-wrapping $\mathcal{B}_6$ initialises an PRIV-security AEAD challenger $C_{\mathsf{PRIV},i}$. Additionally, during the experiment if a DAE ciphertext is generated or decrypted using k_{p_i} (where $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p_i)$), $\mathcal{B}_6$ initialises an PRIV AEAD challenger $C_{\mathsf{PRIV},i}$.

Next, whenever $\mathcal{B}_6$ is required to generate a ciphertext over a message, nonce and header tuple (m, N, H) using k_{p_i} (or the ephemeral key-wrapped key k' encrypted under k_{p_i}) for p_i, $\mathcal{B}_6$ instead queries (m, N, H) to $C_{\mathsf{AUTH},i}$'s encryption oracle. Similarly, if $\mathcal{B}_6$ is required to decrypt a ciphertext, nonce, header tuple (C, N, H) for k_{p_i} (or the ephemeral key-wrapped key k' encrypted under k_{p_i}) for parameter set p_i, $\mathcal{B}_6$ instead queries (C, N, H) to $C_{\mathsf{PRIV},i}$'s decryption oracle. We note that by **Game 1** and the definition of the security experiment, all AEAD keys replaced in this way were already uniformly random and independent of the protocol flow, and $\mathcal{A}$ cannot learn these by issuing a Corrupt query.

If the bit b sampled by $C_{\mathsf{PRIV},i}$ is 0, then the $C_{\mathsf{PRIV},i}$ encrypts the message m honestly, and we are in **Game 1**. Otherwise, if the bit b sampled by the $C_{\mathsf{PRIV},i}$ is 1, then $C_{\mathsf{PRIV},i}$ simply samples a ciphertext uniformly at random, and now the ciphertext is uniformly random and completely independent of the protocol flow and we are in **Game 1**. Any $\mathcal{A}$ that could distinguish this change can be directly used to break the PRIV security of the PRIV primitive. We have at most n_a parameter sets that may be used directly as AEAD keys, and at most n_m key-wrapped AEAD keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{auth}}_{\mathcal{B}_2,\mathsf{AEAD}}(\lambda)$.

Case 3 Analysis: By **Game 1** we replaced the ephemeral key k' used to generate the BCB if p is a key-wrapping algorithm. By **Game 2** we replaced all ciphertexts with uniformly random strings regardless of the challenge bit b. Since the behavior of the security experiment no longer relies of the challenge bit b the $\mathcal{A}$ has no advantage in guessing and thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{FSC}}_{\mathsf{BPSec},\mathcal{A},C_3}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_5}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{PRIV}}_{\mathcal{B}_6,\mathsf{AEAD}}(\lambda)$.

□

We now turn to analyzing the SFSC security of the StrongBPSec construction.

Theorem 2 (SFSC Security for StrongBPSec)**.** *The* StrongBPSec *protocol presented in*

Figure 3.7 is SFSC*-secure. That is, for any PPT algorithm $\mathcal{A}$ against the* SFSC *security experiment (described in Definition 7)* $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A}}(\lambda)$ *is negligible under the* sufcma *and* dae *security of the* MAC *and* DAE *primitives respectively.*

Proof. We note that in the strong FSC game there are five main ways $\mathcal{A}$ can win. In addition to those described in Theorem 1, the SFSC game introduces two new ways for $\mathcal{A}$ to set `win` $\leftarrow$ true. In particular, if $\mathcal{A}$ has successfully dropped message blocks sent by the security source without being detected when the bundle is processed by the security target, or by forging a read receipt for a given message block. The analysis for the three original cases from the FSC game apply here, and so we focus on the other two cases:

C_1: Adversary $\mathcal{A}$ has successfully dropped blocks $\vec{\mathtt{M}}_i$ from the source-constructed bundle $\vec{\mathtt{C}}$ without detection and the bundle payload integrity block $\mathsf{BIB_B}$ verifies correctly;

C_2: Adversary$\mathcal{A}$ has successfully forged a read receipt BRB that verifies correctly.

We proceed via a sequence of games, and bound the difference in the $\mathcal{A}$'s advantage in each game with the underlying cryptographic assumptions. We conclude when the adversary reaches a game where the advantage of that game for that case equals 0, which shows that $\mathcal{A}$ cannot win with non-negligible advantage. We begin by dividing the proof into two separate cases corresponding to each win condition (and denote with $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A},C_i}(\lambda)$ the advantage of the adversary in winning the strong integrity game in Case i). It follows that $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A},C_1}(\lambda) + \ldots + \mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A},C_5}(\lambda)$.
We define with $\mathsf{Adv}^{\mathcal{A}}_{G_i}(\lambda)$ the advantage of $\mathcal{A}$ in Game i. We begin with Case 1.

Case 1: Adversary $\mathcal{A}$ wins by causing some party π^i to `accept` bundle $\vec{\mathtt{C}}$ from a security source $\pi^{\vec{\mathtt{C}}_{\mathtt{PB.id.src}}}$ with missing blocks $\vec{\mathtt{M}}_i$. By the definition of the case, we know that the adversary $\mathcal{A}$ has not been able to Corrupt any keys $(k_p, k_p^{\mathtt{RR}}, k_p^{\mathtt{BB}})$ such that $k_p^{\mathtt{BB}}$ was used to verify (or key-wrap keys to verify) the $\mathsf{BIB_B}$. We proceed via the following series of games.

Game 0 This is the SFSC security game in Case 1: $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A},C_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 This game proceeds identically to **Game 1** of **Case 1** of Theorem 1, with the same reductions and bounds. Thus we have $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda)$.

Game 2 In this game, we introduce an abort event that triggers if any un-Corrupted party "accepts" a message $\vec{\mathtt{M}}$ and no honest party generated a message with the same payload and of the same length. Specifically, this occurs if $\mathcal{A}$ is able to successfully forge $\mathsf{BIB_B}$ using a key associated with a parameter set $\vec{\mathtt{P}}_i$ such that $\mathsf{Corrupt}(\cdot, \vec{\mathtt{P}}_i)$ was not issued. This game proceeds identically to **Game 2** of **Case 1** of **Theorem 1**, with the same reductions and bounds. We have at most n_a parameter sets that may be used directly as MAC keys, and at most n_m key-wrapped MAC keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$.

Case 1 Analysis: By definition of **Case 1** we know that $\mathcal{A}$ has caused $\mathtt{win} \leftarrow \mathtt{true}$ by forcing some party π^i to accept a message with "missing blocks" $\vec{\mathtt{M}}$. Specifically, if π^i accepts a message whilst incorrectly believing that the original bundle had a different amount of blocks than generated. By definition of the security experiment we know that $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p)$ such that π^i used $k_p^{\mathtt{BB}}$ to verify the $\mathsf{BIB_B}$ block. By **Game 1** we replaced the ephemeral key k' used to generate the $\mathsf{BIB_B}$ if p is a key-wrapping algorithm. By **Game 2** we added an abort query that prevents $\mathcal{A}$ from forging $\mathsf{BIB_B}$ blocks. We note that the $\mathsf{BIB_B}$ is computed over each key identifier that was used by the source node SN and the number of target blocks that the key identifier was used for. Thus, if π^i believed that SN generated an incorrect number of blocks, then the bundle integrity block $\mathsf{BIB_B}$ for $\vec{\mathtt{M}}_{\mathtt{PLD}}$ would not verify correctly. Since we abort if the adversary forges their own valid $\mathsf{BIB_B}$, by **Game 2** π^i aborts and thus $\mathcal{A}$ cannot win the game. Thus we have: $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A},C_1}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$. Now we transition to **Case 2**.

Case 2: $\mathcal{A}$ wins by causing a party π^i to accept a read receipt BRB under key $k_p^{\mathtt{RR}}$ but no read receipt was generated by an honest party.. By the definition of the case, we know that the adversary $\mathcal{A}$ has not issued $\mathsf{Corrupt}(\cdot, p)$. We proceed via the following series of games.

Game 0 This is the SFSC security game in Case 2: $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A},C_2}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda)$.

Game 1 Let n_a be the total number of parameter sets used in the experiment. In this game, for any key-wrapping keys associated with parameter sets that have not been Corrupted, we abort if $\mathcal{A}$ forges a DAE ciphertext, and replace honestly generated key-wrapped keys with uniformly random values. This proceeds identically to the reduction described in **Game 1**

of **Case 1** and we find: $\mathsf{Adv}^{\mathcal{A}}_{G_0}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) + n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda)$.

Game 2 In this game, we introduce an abort event that triggers if any un-Corrupted party "accepts" a message $\overrightarrow{\mathtt{M}}$ with a read receipt BRB (generated using k_p^{RR} and no honest party generated BRB. Specifically, this occurs if $\mathcal{A}$ is able to successfully forge BRB using a key associated with a parameter set $\overrightarrow{\mathtt{P}}_i$ such that $\mathsf{Corrupt}(\cdot, \overrightarrow{\mathtt{P}}_i)$ was not issued. This game proceeds similarly (up to a change in notation) to **Game 2** of **Case 1** of **Theorem 1**, with the same reductions and bounds. We have at most n_a parameter sets that may be used directly as MAC keys, and at most n_m key-wrapped MAC keys, thus $\mathsf{Adv}^{\mathcal{A}}_{G_1}(\lambda) \leq \mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$. *Case 2 Analysis:* By **Case 2** we know that a party π^i that verifies a BRB using k_p^{RR} (or a key-wrapped key encrypted under k_p^{R}) that sets $\mathtt{win} \leftarrow \mathtt{true}$ such that $\mathcal{A}$ has not issued $\mathsf{Corrupt}(i, p)$ will abort. Thus $\mathcal{A}$ cannot win the game in **Case 2** and thus we have $\mathsf{Adv}^{\mathcal{A}}_{G_2}(\lambda) = 0$, and it follows that $\mathsf{Adv}^{\mathsf{SFSC}}_{\mathsf{StrBPSec},\mathcal{A},C_2}(\lambda) \leq n_a \cdot \mathsf{Adv}^{\mathsf{dae}}_{\mathsf{DAE},\mathcal{B}_1}(\lambda) + (n_a + n_m) \cdot \mathsf{Adv}^{\mathsf{sufcma}}_{\mathcal{B}_2,\mathsf{MAC}}(\lambda)$.

□

3.7 Conclusion

Space networking follows different restrictions than standard, terrestrial Internet-style networking, which as consequently motivated work in developing and deploying suitable networking protocols over the years. However, the cryptographic security of such, including DTN protocols like BPSec, has largely escaped formal cryptographic analysis, leading to a lack of understanding of how well the security intentions of such protocols are realized. This work provides the first formal cryptographic analysis and provable security treatment of one such protocol and lays a ground work for further analyses in this area. Furthermore, we propose StrongBPSec, a strong alternative to BPSec with additional integrity guarantees. We note that integrating our StrongBPSec with read receipts into existing BPSec instantiations and implementations can be achieved through multiple approaches: the existing standard could be extended with a new read receipt block type (e.g. as an Other Security Block [12, Section 10]); or, more conveniently, read receipts may also be introduced as an extension to the existing BIB block type in the form of a specialized BIB block. We leave exploring the specifics of how StrongBPSec can be incorporated within the existing

BPSec paradigm to future work. We highlight, regardless of how it is integrated, the true value of StrongBPSec lies in adopting its read receipts with additional integrity guarantees within the current BPSec and BPSec Default Security Context standards. We further leave finding more efficient *key-wrapping* solutions and reducing the security overhead of BPSec to future work.

CHAPTER 4:
Guardian-MLS: MLS Under Restricted Key Updates

Extending security to constrained users

As established in Chapter 1, the inadequacy of protocols in degraded environments extends beyond validating existing channel protocols to identifying and addressing fundamental limitations in their key agreement components. While the commercial sector has driven advances in continuous key agreement protocols like Messaging Layer Security (MLS), these protocols have been optimized for well-connected, low-latency terrestrial networks where all participants can actively engage in key updates. For bandwidth-constrained users—including space systems, military units operating under electromagnetic emissions control (EMCON), and Internet of Things (IoT) devices with limited uplink capacity—the requirement that devices perform self-updates to achieve post-compromise security is often operationally infeasible. In both space-to-ground and space-to-space communications, devices frequently appear offline due to link unavailability (orbital mechanics), link disruptions (weather), or operationally-enforced radio silence. Standard MLS requires that users who cannot self-update be removed from the group and re-added—a sequence of operations that may be impractical or create operational barriers that curtail the very post-compromise security (self-healing ability) that makes continuous key agreement valuable.

Pillar I: Advancing Security Models for Degraded Environments. Recall from Chapter 1 that Pillar I establishes rigorous security foundations not only by analyzing existing protocols but also by extending security models to address operational realities previously neglected by the commercial sector's focus on abundant bandwidth and continuous connectivity. The guardianship mechanism introduced in this chapter represents a fundamental advancement to continuous group key agreement (CGKA) security models, enabling post-compromise security for passive participants through updates initiated by other group members. This extension directly addresses the gaps in protocol scrutiny caused by the historical focus on terrestrial use cases, providing formal security guarantees for environments where active participation cannot be assumed. The security properties established through this rigorous analysis transfer directly to the protocol integrations in Pillar II: when BPSec-MLS (Chap-

ter 6) and QUIC-MLS (Chapter 7) incorporate Guardian MLS mechanisms, they inherit proven post-compromise security even for receive-only or emission-controlled devices.

Chapter Contribution and Impact. This chapter extends the security model of Messaging Layer Security (MLS) to address a fundamental limitation: the inability of receive-only or emission-controlled devices to perform self-updates while maintaining post-compromise security. In traditional MLS, post-compromise security (PCS), or the ability to self-heal after a compromise, requires that compromised parties actively update their own cryptographic state. Even though constrained devices may be in restricted settings, adversaries may not be similarly constrained. Thus, a compromise may prevent those devices and their communication groups from recovering security, creating a persistent vulnerability.

At the highest level, this work transforms continuous key agreement from a technology accessible only to well-connected devices into a security mechanism viable for the full spectrum of operational environments. By enabling post-compromise security for passive participants, this work removes a critical barrier to deploying modern cryptographic protocols in space systems, tactical military networks, and IoT deployments. The operational flexibility gained by allowing devices to maintain group membership and security even when unable to transmit directly supports the resilient, adaptive security architectures envisioned in Chapter 1. For space missions involving constellations of satellites with intermittent ground contact or military operations under EMCON, this capability is transformative: compromised devices can be healed by other group members without requiring the compromised device to break radio silence or wait for favorable link conditions.

At the protocol level, this chapter presents a novel analysis of how to attain post-compromise security (PCS) under restricted key update settings for MLS, enabling members who cannot update themselves to achieve PCS without needing to leave and rejoin the group. A thorough analysis of various protocol constructions, which range from naive adaptations to novel designs, scrutinizes eight candidate architectures for their ability to achieve PCS for constrained group members. These designs are systematically compared across security features (PCS strength, resistance to adversarial disruption) and architectural features (bandwidth overhead, implementation complexity, compatibility with existing MLS infrastructure). Through this rigorous comparative analysis, the work downselects to a single guardianship construction recommended for deployment that balances security guarantees

with operational feasibility.

At the technical level, the guardianship mechanism enables PCS for passive participants through updates initiated by other group members ("guardians"), thereby extending the security guarantees of continuous key agreement to environments where active participation cannot be assumed. The formal security analysis proves that guardianship preserves the core security properties of MLS—including forward secrecy and post-compromise security—while accommodating operational constraints that would render standard MLS unusable. This advancement benefits the broader class of constrained devices far beyond space applications and represents a foundational improvement to CGKA security models applicable to tactical military communications, industrial IoT deployments, and any scenario where bandwidth asymmetry or operational requirements limit uplink capacity.

Real-World Impact and Standardization. The final guardianship design has been proposed as an IETF Internet-Draft to the MLS working group and is in the process of gaining working group adoption for standardization. This standardization pathway ensures that the security model extensions developed for space and military use cases can benefit the broader internet community, enabling MLS deployment in constrained environments previously considered unsuitable for continuous key agreement protocols. The work directly enables the BPSec-MLS and QUIC-MLS integrations in Chapters 6 and 7 by providing proven mechanisms for maintaining security even when satellites, ground stations, or relay nodes cannot actively participate in key updates during critical mission phases.

Joint Work and Contributions. This section presents Guardian MLS which was joint work with Elsie Fondevik (Kongsberg Aerospace & Defense) and Britta Hale (Naval Postgraduate School). This work has been reprinted here with permission from all authors.[6] The candidate's contributions to this paper include writing of use case discussion, writing clarification of the design rationale, and general writing and editing. The candidate collaborated on with the constructions design, security modeling, and the co-constructing the security proofs, which were predominantly drafted by Elsie Fondevik and Britta Hale.

[6]The 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.

Chapter Abstract

Post-compromise security (PCS) has been a core goal of end-to-end secure messaging for many years, both in one-to-one continuous key agreement (CKA) and for groups (CGKA). PCS relies on a compromised party to perform a key update in order to 'self-heal'. However, due to bandwidth constraints and various other environmental demands of the growing number of use cases for such CGKA protocols, a group member may not be able to issue such updates. In this work, we address the issue of devices functioning in limited mode through the introduction of *guardianship*, where a designated guardian can perform key updates on the behalf of its paired edge device. This work introduces Guardianship PCS (GPCS) security, investigates constructions and architectural designs in the pursuit of GPCS, and describes trade-offs.

4.1 Introduction

One classic scenario discussed in the context of messaging protocol security involves a user who temporarily surrenders their device while traveling across a country border. The challenge is to maintain maximum security during the temporary loss of control over the device. This challenge has been closely tied to the exploration of forward secrecy (FS) and post-compromise security (PCS) guarantees in instant messaging and advancements in continuous key agreement (CKA) protocols. Through CKA, keys are continually updated between communicating parties, ensuring the security of past keys even if exposed in a later epoch (FS). The introduction of new keying material at each update can further enhance PCS, allowing the connection to "self-heal" after an honest update from a compromised device.

The concept of CKA evolved into continuous group key agreement (CGKA) to facilitate secure communication among multiple devices [44] and the standardization of the Messaging Layer Security (MLS) protocol (RFC9420), published by the Internet Engineering Task Force (IETF) in 2023 [3]. MLS provides both forward secrecy and post-compromise security, allowing for the party in the above traveling scenario to "heal" from compromise by updating their keys before the adversary is able to inject an update of their own. However, another challenge arises when the scenario is realistically expanded: limited bandwidth, poor service, or low device battery power may impede key updates in addition to non-technical constraints. This necessitates a solution for restricted users in a MLS group – but one that also does not introduce security weaknesses to other group members under regular

operation.

Limited mode security is especially relevant for some user groups such as political activists and journalists. For example, a remote report member of a journalist team may have long periods of inactivity that undermines the group's security. Current recommendations suggest evicting and re-adding the reporter regularly, as per the organization's security policies but this may be impractical for the reporter. Tangentially, this approach discourages the group membership of devices or users operating routinely in limited environments, such as low-power devices or remote sensors.

While extensive work exists on MLS and CGKAs, research on methods for addressing PCS security when devices have limited ability to send updates is lacking. One intuitive solution is if the offline user "edge" device is paired with another "guardian" device of the user's that could update on its behalf. For instance, if the reporter has a mobile device, laptop, tablet, smart watch, etc., one of those could be left in a secure and reliable bandwidth location allowing it take on the key update action. We call the concept of PCS key updates on behalf of another device *Guardian PCS (GPCS)*. Of course, GPCS raises questions about what information a guardian has or does not have access to, whether the guardian device can impersonate the edge device to the group (or vice versa), whether the other group members have awareness of the guardian existence and its relationship to the edge, and many more core security considerations.

In this work, we delve into the solution space for the posed scenario and guardianship options, initializing a study of the trade-offs, construction options, and how architectural design considerations affect the end security. This work expands the state of the art in security and privacy in practical messaging. It contributes to current efforts in the IETF MLS working group [45]. We cross-compare several cryptographic models to guide down-selection by GPCS implementations with security analysis and detailing foreseeable issues pertaining to cost, storage, bandwidth, and additional hardware.

Contributions. We initialize a study of guardianship and its use within continuous group key agreement. Specifically, we I. introduce a formal definition for Guardian Continuous Group Key Agreement (GCGKA) and associated security model GCGKA. II. provide a cross-comparison of the design space of guardian extension protocols for the MLS protocol.III. analyze selected GCGKA protocols over MLS for architectural design considerations and

GCGKA security.

4.2 Related Work

This work is based on continuous key agreement protocols, which have historically evolved from one-to-one key agreement to include group key agreement protocols. End-to-end encryption [46] is a fundamental goal in all cases considered.

Continuous Key Agreement and Security. The Signal protocol [47] marked a pivot point in the design of secure messaging due to a variety of security properties supported beyond confidentiality and data authenticity. Research on CKA protocols has gained considerable traction, with use in messaging applications such as Signal [48], Whatsapp [49], and Facebook Messenger [50], and a variety of modeling techniques for assessing security [51]–[55]. CKAs offer a single, long-lived session with unspecified termination. Instead of periodic handshakes to establish keying material, CKA continuously evolve the secret state through asynchronous *updates*. Due to the asynchronicity of the updates, CKA has the ability to support offline devices and even devices in receive-only mode.

The pursuit of FS and PCS in group messaging protocols can sometimes be at odds with efficiency. After all, a single long-lived shared group state with no updates to the keying material supports good application performance under limited bandwidth or computational power – it also comes at the cost of FS and PCS security.[7] Meanwhile, performing frequent CKA updates on one-to-one channels, such as in the Signal protocol [48] supports FS and PCS but imposes costs in computational processing time and bandwidth costs for sending updates. This has led to an effort for the design and standardization of a CGKA protocol, Messaging Layer Security (MLS) [57], and has since inspired extensive analysis throughout its development cycle [58]–[62], including under continuous group key agreement (CGKA) model [44]. Notably, MLS security has matured as a result under the contexts of active attackers, man-in-the-middle, and cross-group effects scenarios.

Guardianship vs Ghosts. One may wonder about the distinction between guardianship and

[7]It should be noted that traditional definitions for PCS do not account for authentication issues or impersonation from the compromise of signature keys [56], instead focusing on *confidentiality* properties only. We similarly focus on confidentiality in this paper, but provide an overview comparison of entity authenticity properties in Section 4.7.

ghost users as have been proposed [63] and are often controversial [64]. First, a fundamental goal of guardianship is preservation of the FS and PCS guarantees often found in current end-to-end encrypted messaging protocols (this work notes intuitive guardianship designs that would break this foundation and should be avoided). While some strategies remove [3] or quarantine [65] inactive users, we look to preserve their ability to participate through use of the guardian. Unlike ghost users, where a third party is silently added to the group, guardianship is an additional device under the current user's ownership, with varying degrees of visibility to other group members. The guardian's role is to enforce strong FS and PCS guarantees even if its corresponding edge is unable to send updates, and as such there is a fundamental assumption on the guardian of being held in a more secure location with less risk of compromise. In short, guardianship is an intended feature under the user's control.

4.3 Prerequisites

We build upon the basis of a continuous group key agreement (CGKA) protocol, and leverage the underlying notation from [44] as found in Definition 9. The eventual constructions will use MLS as a CGKA example with guardianship capabilities added.

Definition 9 (CGKA, adapted from [44] for clarity)**.** *A CGKA scheme* $\Pi = (\mathsf{init}, \mathsf{create}, \mathsf{add}, \mathsf{rem}, \mathsf{upd}, \mathsf{proc})$ *consists of the following algorithms:*

- *Initialize:* init *takes as input a group member identity,* ID*, and outputs an initial shared group state* $\gamma[\mathsf{ID}]$.
- *Create Group:* create *takes as input a state* $\gamma[\mathsf{ID}]$ *and a list of identities for group members* $G = (\mathsf{ID}_1, \ldots, \mathsf{ID}_n)$*, and outputs a new state* $\gamma[\mathsf{ID}]$ *and a welcome message* $Welcome$.
- *Add Group Member:* add *takes a shared group state* $\gamma[\mathsf{ID}]$ *and ID,* ID*, and outputs a new state* $\gamma[\mathsf{ID}]$*, welcome message* $Welcome$*, and control message (public update value)* T.
- *Remove Group Member:* rem *takes a state* $\gamma[\mathsf{ID}]$ *and an ID,* ID*, and outputs a new group state* $\gamma[\mathsf{ID}]$ *and a control message (public update value)* T.
- *Update Generation:* upd *takes a shared group state,* $\gamma[\mathsf{ID}]$*, and outputs a new state* $\gamma[\mathsf{ID}]$ *and a control message (public update value)* T.
- *Process Received Update:* proc *takes a state* $\gamma[\mathsf{ID}]$ *and a control message (public update value)* T *and outputs a new state* $\gamma[\mathsf{ID}]$ *and an update secret* I.

Definition 10 (CGKA Security (adapted from [44])). *Let* Π *be a CGKA protocol and let* $\mathcal{A}$ *be a PPT adversarial algorithm against* Π *as defined in* blue *in Figure 4.2. We define the adversarial advantage of* $\mathcal{A}$ *as*

$$\mathsf{Adv}^{cgka}_{\Pi-\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{CGKA}_{\Pi-\mathcal{A}}(()\lambda)\right] - \frac{1}{2}\right|$$

and say that the protocol Π *is* CGKA-secure *if* $\mathsf{Adv}^{cgka}_{\Pi-\mathcal{A}}(\lambda)$ *is negligible for all* $\mathcal{A}$. [8]

Some of our constructions also utilize the continuous key agreement (CKA) protocol for a secure pairwise channel between the guardian and edge. We utilize the CKA definition and security notions from [66], with Definition 11 provided as reference.

$\mathrm{Exp}^{\mathsf{CKA}}_{\mathcal{A}}$:

$b \leftarrow\$ \{0,1\}$
$\mathsf{k} \leftarrow\$ \mathcal{K}$
$\gamma^A \leftarrow \mathsf{Init\text{-}A}(\mathsf{k})$
$\gamma^B \leftarrow \mathsf{Init\text{-}B}(\mathsf{k})$
$t_A, t_B \leftarrow 0$

send-A

$t_A{+}{+}$
$(\gamma, T_{t_A}, I_{t_A}) \leftarrow\$ \mathsf{S}(\gamma)$
return (T_{t_A}, I_{t_A})

corr-A

req allow-corr or fin_A
return γ^A

send-A'(r)

$t_A{+}{+}$
req allow-corr
$(\gamma, T_{t_A}, I_{t_A}) \leftarrow\$ \mathsf{S}(\gamma; r)$
return (T_{t_A}, I_{t_A})

recieve-A

$t_A{+}{+}$
$(\gamma^A, *) \leftarrow\$ \mathsf{R}(\gamma^A, T_{t_A})$

chall-A

$t_A{+}{+}$
req $t_A = t^*$
$(\gamma, T_{t_A}, I_{t_A}) \leftarrow\$ \mathsf{S}(\gamma)$
if $b = 0$
  return (T_{t_A}, I_{t_A})
else
  $I \leftarrow\$ \mathcal{I}$
  return (T_{t_A}, I)

CKA safety predicate

allow-corr$_p$: $\Longleftrightarrow$ $\max t_A, t_B \leq t^* - 2$
finished$_p$: $\Longleftrightarrow$ $t_p \geq t^* + \Delta_{\mathsf{CKA}}$

Figure 4.1. The CKA experiment from [66].

Definition 11 (CKA [66]). *A CKA scheme is a quadruple of algorithms CKA = (Init-A, Init-B, S, R), where*

- *Init-(A)/(B) takes a key and produces an initial state* $\gamma^A \leftarrow$ *Init-A(k)* ($\gamma^B \leftarrow$ *Init-B(k)*).
- *S takes a state, and produces a new state, message, and key* $(\gamma', T, I) \leftarrow\$ S(\gamma)$.
- *R takes a state and message and produces a new state and key* $(\gamma', I) \leftarrow R(\gamma, T)$.

[8] For CGKA [44] the safety predicate Section 4.5 aligns to $X \in \{FS, PCS\}$ and $\iota = \mathsf{ID}$. [44]

Denote by $\mathcal{K}$ the space of initialization keys k and by $\mathcal{I}$ the space of CKA keys I.

Definition 12 (CKA Security (adapted from [67])). *Let CKA be a continuous key agreement protocol, and let $t^* \in \mathbb{N}$ be an epoch index and $\Delta_{CKA} \in \mathbb{N}$ be the number of epochs until an epoch no longer contains secret information pertaining to a challenge. For a PPT algorithm $\mathcal{A}$, we define the advantage of $\mathcal{A}$ in the CKA security experiment (see Figure 4.1) to be*

$$\mathsf{Adv}^{cka,t^*,\Delta_{cka}}_{CKA-\mathcal{A}}(\lambda) = \left|\Pr[Exp^{CKA,t^*,\Delta_{CKA}}_{CKA-\mathcal{A}}(\lambda) = 1] - \frac{1}{2}\right| .$$

We say that CKA is CKA-secure if, for all $\mathcal{A}$, $\mathsf{Adv}^{cka,t^*,\Delta_{cka}}_{CKA-\mathcal{A}}(\lambda)$ is negligible in the security parameter λ.

Our constructions also utilize a Key Derivation Function. Here we provide the definition for PRF security of the KDF.

Definition 13 (Key Derivation Function (KDF) (adapted from [68])). *A* key derivation function, *$KDF(sk, \mathtt{id}) \rightarrow sk'$, is a pseudorandom function $\mathcal{K} \times \mathcal{N} \rightarrow \mathcal{K}$ that takes as input keying material sk and an optional key identifier* `id` *and outputs a key sk'.*

4.4 Terminology and GCGKA Definition

The intuition behind GCGKA is to allow users to enter an operational mode in which their ability to send updates is restricted – for example due to bandwidth, power, or other limitations – while still retaining full security. We treat the underlying group key exchange as a CGKA blackbox, without modification, and enhance it to allow guardians to update on behalf of another specified user. As a requirement, guardianship should not reduce the security of the underlying CGKA protocol.

4.4.1 Terminology

Edge Device: an *edge* device is an original user equipment device. Such a device may operate in receive-only mode or in another limited fashion such that sending regular keying updates is impractical or even impossible.

Guardian: the *guardian* device operates from a secured space with reliable network access. The intent is for the guardian device to be paired with, and provide keying updates on behalf

of the edge device. This supports FS in the event the edge device is compromised. When the edge device is in an active state that allows for it to perform keying updates of its own, the guardian device may be placed in offline mode.

Edge Device Operational Modes An edge device has two modes of operation. Before an edge device enters a new state the MLS delivery service (DS), which facilitates group state consensus by ensuring in-order delivery of MLS messages, must be informed. The operational states are:

Online mode: The edge device is available and running the group protocol as per specification. In this state it does not have need of a guardian.
Limited mode: The edge device can *receive* application and key update messages but cannot send key updates messages. Depending on limitations, the edge may still send application messages.

Guardian Operational Modes A guardian may be in one of two modes as well, contingent on the mode of the edge.

Offline mode: When an edge device is online the guardian may be set to be inactive or offline (depending on the guardianship construction).
Online mode: When the edge device enters limited mode, it becomes reliant on a guardian. Therefore the guardian status must be set to online to send key updates.

4.4.2 Notation and State Variables

Independent of construction a *Guardian Protocol* (GP) consists of an edge device and a guardian that interchangeably access the underlying group to update the group session key. These access point(s) into the underlying group used by the guardian and edge device can be viewed as a group member in the underlying group and has its own unique ID inherited from CGKA. We call this CGKA group member node an *anchor*[9]. In the experiment and definition we notationally differentiate between anchor ID's (ID) and guardian/edge device ID's (id_G/id_E). Depending on protocol design, the guardian and edge device may share an anchor or they may have distinct anchors.

To the underlying protocol an anchor is indistinguishable from a regular group member.

[9]For example, in MLS, an anchor would be a leaf node in the MLS tree.

The anchor maintains the same variables as any CGKA group member: an identity id and a state $\gamma[\mathsf{ID}]$. However, the anchor is a logical construct node vs. a real device; its state is accessed and maintained by the edge device and/or guardian. Meanwhile, the edge device and guardian each consist of an identity, id_E and id_G respectively, together with an expanded state $\mathrm{S}[id_E]$ and $\mathrm{S}[id_G]$ respectively. For an edge device id_E with anchor ID_E, the state $\mathrm{S}[id_E] = (\gamma[id_E], \gamma[\mathsf{ID}_E])$ is a two tuple consisting of the state of edge device-specific information together with the state of the anchor node. This inherently includes the associated signing key of the anchor. If the edge device does not have direct access to the anchor, the $\gamma[\mathsf{ID}_E]$ component is set to ϵ. The guardian state is similarly constructed using the guardian's id, e.g., $\mathrm{S}[id_G] = (\gamma[id_G], \gamma[\mathsf{ID}_G])$.

Moreover, the edge and guardian sessions contain encoded information. For an edge or guardian id, $mode[id]$ is a binary flag specifying the instances' operational mode: $\{isOnline, isOffline\}$ for guardians and $\{isOnline, isLimited\}$ for edges. The variable $anchor[id]$ stores the id's anchor id. Additionally have the session state variable $getGuardian[id]$ and $getEdge[id]$, which respectively specifies the guardian or edge of a particular id.

4.4.3 GCGKA Protocol Definition

A guardian protocol consists of eight algorithms in addition to those inherited form CGKA. The first three algorithms focus on the addition and removal of guardian or edge devices. In the case of addition, it is assumed that anchor(s), if needed, already exist. If a user wishes to join the group directly as a edge device with a guardian then the edge device will first need to become a member of the underlying group using CGKA function add before initiating the guardianship protocol to include the guardian. When exiting the guardianship protocol (i.e., returning to normal, non-guardianship membership), the guardian will be removed and the edge device will become a regular member of the group. If the edge device is removed, both guardian and all corresponding anchors will be removed from the underlying group. The next two algorithms, $\mathsf{enterLimitedMode}$ and $\mathsf{exitLimitedMode}$, toggle between the operational modes. These functions affect who is authorized to preform updates in the underlying group. The final three algorithms are used to preform and process key updates. Updates may either be issued to the underlying group or between guardian and edge device. Similarly, processing is split in two; a received message may either be in the format of the underlying group or a GP proprietary format. Depending on the type, the input is processed

accordingly. Formally, we define a guardian protocol as follows:

Definition 14. *Let* Π *be a CGKA protocol according to Definition 9. We extend* Π *to a guardian CGKA protocol, GCGKA, by introducing the following algorithms:*

enterGship ***In:*** $\gamma[\mathsf{ID}_E], \gamma[\mathsf{ID}_G]$ ***Out:*** $id_E, \mathrm{S}[id_E], id_G, \mathrm{S}[id_G]$

This algorithm takes the anchor states for an edge $\mathrm{S}[id_E]$ *and guardian* $\mathrm{S}[id_G]$ *as input and returns ids for the added edge and guardian together with their initial states. It is required that no edge or guardian is currently appended to the anchor ids. If the operation could not be preformed, an error symbol* $\perp$ *is returned.*

exitGship ***In:*** id_E, $\mathrm{S}[id_E]$, id_G ***Out:*** $\gamma[\mathsf{id}_E]$, T

This algorithm takes as input an edge id id_E*, the state of the edge* $\mathrm{S}[id_E]$*, and a guardian id* id_G*, and returns an updated anchor CGKA state* $\gamma[\mathsf{id}_E]$ *and a control message* T*. If* $getGuardian[id_E] \neq id_G$ *or* $getEdge[id_G] \neq id_E$*, the algorithm outputs an error symbol* $\perp$*. The algorithm terminates the running of guardianship protocol. If the edge and guardian have separate anchors, the guardian anchor is removed from the group via a call to the CGKA* rem *algorithm.*

removePair ***In:*** $\gamma[\mathsf{id}]$, id_E, id_G, ***Out:*** $\gamma[\mathsf{id}]$, T_1, T_2

The algorithm takes as input the underlying CGKA group leader's state, $\gamma[\mathsf{id}]$*, and the edge's and guardian's anchor ids,* ID_E *and* ID_G *respectively. If the edge and guardian share an anchor,* $\mathsf{ID}_E = \mathsf{ID}_G$*. If* $getGuardian[id_E] \neq id_G$ *or* $getEdge[id_G] \neq id_E$*, the algorithm outputs an error symbol* $\perp$*. The algorithm calls the CGKA* rem *remove algorithm twice:* $\mathsf{rem}(\gamma[\mathsf{id}], \mathsf{id}_G) \rightarrow (\gamma[\mathsf{id}]', T_1)$ *and then* $\mathsf{rem}(\gamma[\mathsf{id}]', \mathsf{id}_E) \rightarrow (\gamma[\mathsf{id}]'', T_2)$*, removing both anchors from the underlying CGKA group. An updated CGKA state* $(\gamma[\mathsf{id}] \leftarrow \gamma[\mathsf{id}]'')$ *together with two CGKA control messages is returned as output. If the edge and guardian share the same anchor, then* rem *is only called once and* T_2 *is set to* ϵ*.*

enterLimitedMode ***In:*** id_E, id_G ***Out:*** ϵ

This algorithm takes as input the id of an edge device and its guardian, id_G*. If* $getGuardian[id_E] \neq id_G$ *or* $getEdge[id_G] \neq id_E$*, the algorithm outputs an error symbol* $\perp$*. The algorithm sets* $mode[id_E] \leftarrow$ *isOffline to put the edge device into limited mode while* $mode[id_G] \leftarrow$ *isOnline sets guardian as active.*

exitLimitedMode ***In:*** id_E, id_G ***Out:*** ϵ

This algorithm takes as input the id of an edge device, id_E*, and its guardian,* id_G*.*

If $\mathit{getGuardian}[id_E] \neq id_G$ *or* $\mathit{getEdge}[id_G] \neq id_E$*, the algorithm outputs an error symbol* $\perp$*. The algorithm sets* $\mathit{mode}[id_E] \leftarrow \mathit{isOnline}$*, putting it online, while* $\mathit{mode}[id_G] \leftarrow \mathit{isOffline}$ *the guardian is set to offline mode (when possible* [10]*).*

updateE ***In:*** id_E, $\mathrm{S}[id_E]$ ***Out:*** $\mathrm{S}[id_E]$, T

The algorithm takes as input an edge device id, id_E*, and its corresponding state* $\mathrm{S}[id_E]$*. It checks that* $\mathit{mode}[id_E] = \mathit{isOnline}$ *and outputs an updated edge device state together* $\mathrm{S}[id_E]$ *with a control message* T*. It outputs an error symbol* $\perp$ *if device is not online.*

updateG ***In:*** id_G, $\mathrm{S}[id_G]$ ***Out:*** $\mathrm{S}[id_G]$, T

The algorithm takes as input a guardian id, id_G*, and its corresponding state* $\mathrm{S}[id_G]$*. It checks that* $\mathit{mode}[id_G] = \mathit{isOnline}$*, outputting an error symbol* $\perp$ *if not, and else outputs an updated guardian state* $\mathrm{S}[id_G]$ *together with a control message* T*.*

processUpdGE ***In:*** $\mathrm{S}[id]$, T ***Out:*** $\mathrm{S}[id]$, I

The algorithm takes as inputs the state of an edge or guardian, $\mathrm{S}[id]$*, and a control message* T*. It returns an updated device state and a group update secret* I*.* [11]

4.5 GCGKA Security

The CGKA security model is used as a basis for our guardianship security model. We assume the underlying group key exchange protocol, on which guardian protocol builds upon, is defined according to Definition 9. Consequently, the guardian schemes we explore can be applied to a variety of concrete CGKA schemes. The *guardian protocol (GCGKA)* model is constructed in such a way that group members may individually choose to run the GCGKA protocol or the underlying CGKA based protocol. It is therefore especially important that any communication between GCGKA and the underlying protocol is done according to CGKA specification.

For GCGKA we retain the following four requirements from CGKA [44], and include an additional fifth requirement, *guardian post-compromise security (GPCS)*.

Correctness: All group members output the same update secret I in update epochs.

Privacy: The update secrets look random given the transcript of control messages.

[10] If the guardian is identical to the anchor for id_E, i.e. $\mathsf{id}_E = id_G$ as in some possible constructions, it may not be possible to put it into offline mode.

[11] The algorithm can process control messages to update the internal state whether those are CGKA-based or through a dedicate guardian-edge channel.

Forward secrecy (FS): If the state of any group member is leaked at some point, all previous update secrets remain hidden from the attacker.

Post-compromise security (PCS): After every group member whose state was leaked performs an update, the update secrets become secret again.

Guardian post-compromise security (GPCS): After the guardian for any edge device whose state was leaked performs an update (that is processed by the group), update secrets become secret again.

Remark 1. *The guardianship architecture allows for a distinct forward secrecy feature in edge device limited mode. Namely, even if all edge devices are in limited mode and cannot issue updates – a mode normally expanding the FS vulnerability window – the guardian can still control and limit that vulnerability window by issuing updates. This may be done on some automated periodicity.*

Security Experiment

The security experiment for GCGKA can be found in Figure 4.2. and Figure 4.3. The security experiment encompasses that of CGKA, where parts taken directly from CGKA can be found in blue. The new additions are written in black.

Initialization. The protocol is initiated by setting up the variables used in the experiment. With the exception of a few new variables, the process is identical to the CGKA init-query. The identity states $\gamma[\mathsf{ID}]$ are instantiated with a call to CGKA init for a set of IDs. Like CGKA, every epoch (epoch[]) has an update leader (lead[]), update secret (I[]), and group members (**Group**[]). Because users can issue multiple updates and other actions (e.g. add/remove) within a single epoch, a counter ($ctr[]$) is used to track the actions for the update reconciliation process via the control messages (T) which are stored in $\mathbf{M}[]$. Like in CGKA, **Group**[] tracks the underlying CGKA group members (including anchors) while $\mathbf{E}[]$ and $\mathbf{G}[]$ specifically keep track of edge and guardian members, respectively.

Creation and Maintenance. To create a edge-guardian pair, first a call to $\mathsf{createGroup}$ must be made to establish the underlying group. Then, adding, removing, and updating can be accomplished through specific queries. For correctness, checks are made to ensure only valid members of **Group**[] can be added as edges, duplicate adds and removes cannot occur, and only edges can add or remove valid guardians.

Committing Updates. Similar to CGKA, all group operations are published to $\mathbf{M}[]$ via a control message T which is accessible to the adversary in read-only mode and is used by the server to adjudicate the order of commits. $\mathbf{M}[]$ is indexed on the epoch of the intended operation, the ID of the caller, the ID of the affected member, and the counter value. Since control messages can, in some constructions, be between the edge and guardian as non-CGKA control messages, we denote $\mathrm{T}_{CGKA} \subseteq \mathbf{M}[]$ as the subset that is specific to CGKA. (See [44, Section 3.2] for how the $send\text{-}update()$ and $deliver()$ within CGKA are used to deconflict and commit updates atomically.)

Reveal and Corrupt. Reveal follows that of CGKA functionality, while corruption is allowed for guardians, edges, and regular group members. Anchors associated to edges or guardians are specifically disallowed – these are logical constructs held in memory by the edge/guardian, so we follow real-world use where a corruption to access such state would be via the edge or guardian. To examine the security properties of the GCGKA compared to CGKA, we need to fine tune adversarial reveal and corruption capabilities. This is done through modifying the safety predicate from CGKA, which captures FS and PCS, to also capture GPCS via guardian-edge pairs (Section 4.5).

Definition 15 (GCGKA Security)**.** *Let GCGKA be an extension of CGKA as defined in Definition 14, let* Π *be a GCGKA protocol and let the adversary* $\mathcal{A}$ *be a PPT algorithm against* Π *as defined in Figure 4.2. We define the adversarial advantage of a* $\mathcal{A}$ *as*

$$\mathsf{Adv}^{\mathbf{X}-gcgka}_{\Pi-\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathbf{X}-GCGKA}_{\Pi-\mathcal{A}}(()\lambda)\right] - \frac{1}{2}\right|$$

and say that protocol Π *is* $\mathbf{X}-$GCGKA-secure *if* $\mathsf{Adv}^{gcgka}_{\Pi-\mathcal{A}}(\lambda)$ *is negligible for all* $\mathcal{A}$.

Win. The adversary wins the GCGKA experiment by correctly answering if the presented update key, is real or random. The safety predicate enforces PCS, FS, and GPCS and denies trivial wins. A trivial win is defined as either an adversary that corrupts and challenges a session without an update having been issued between the queries. Or, key deletion was disabled prior to a corruption. As in CGKA, the function q2e($\mathbf{q}$) returns the epoch corresponding to the query $\mathbf{q}$. This epoch is relative to ι, which may be an anchor ID or guardian/edge id, used as input to query $\mathbf{q}$.

4.6 Protocol Constructions

We discuss GCGKA protocol variation in this section. The breadth of GCGKA variations can be seen through three binary design decision points: whether to architect the edge-guardian pair organically within a protocol or extended off of it, whether the edge and guardian devices share a signing key, and whether the edge and guardian devices share randomness. These decisions yield the possible permutations of guardian-edge pair designs shown in the table in Figure 4.5. Where, an anchor refers to the underlying CGKA protocol's group member's logical node. A signature, (sk, pk), refers to the signing keys belonging to that node. Randomness, \$, refers to the seed used in key generation (see Remark 2). The specific protocols will be described in the following sections.

We choose MLS as the CGKA basis for our protocol designs, and Signal [66] as a CKA example where required. For simplicity, we use CGKA and CKA functions to the greatest extent possible. This also allows for clearer extrapolation to other constructions. For the mapping of other and underlying CGKA algorithms (init, add, rem, upd, proc) to MLS, see [44]; we assume these mappings and present the additional GCGKA functions relying on them.

Of the eight possible combinations in the table in Figure 4.5 , three (labeled a, b, and c) are dismissed due to incompatibility with secure group protocol notions. In construction (a), having a guardian edge pair anchored on distinct nodes inside the MLS protocol with unique signing keys but shared randomness removes all possibility of achieving PCS: if an attacker compromises one node, it would be able to compute any new keys that the other uses to heal the group with. This would nullify the goal of maintaining the security of the underlying CGKA. Next, consider (b) and (c); having distinct anchors and shared signing keys: this combination implies clones being allowed as group members, which invalidates non-repudiation and the MLS invariant of unique identities. In contrast, we do consider cases where two different signature keys are registered at the same anchor. The rest of this section explores the implications of our selection choice. Figure 4.5 gives a conceptual overview, while Figure 4.6 together with Figure 4.7 specifies the protocol description.

Π_1 **– MLS siblings** Π_1 is nearly a direct copy of MLS with edge and guardian as distinct members. Recall, MLS stores keys in a logical binary tree where group members are represented as leaf nodes and the shared session key is the root. MLS has been shown to

achieve FS and PCS for its group members [58], but if a user is unable to update (i.e., is in limited mode) the security of the entire group is affected as it leaves a path from a leaf node to root unaltered [60]. This construction requires guardian and edge device to be siblings in the MLS tree in order to share a path to the root (see Figure 4.6). This way, when the edge enters limited mode, guardian-updates will directly affect their parent node and the shared path to the root: thus, keys along the path are not stagnant. GPCS is, however, not achieve since only edge removal or self-update can heal the group (i.e. CGKA PCS). Use of updateable public key encryption could improve forward secrecy [69], but this is out of scope for this work.

Π_2 **– Message Forwarding** In Π_2, the guardian is an MLS group member, with a separate forwarding channel maintained between the edge and guardian. We use a CKA channel for simplicity in the shown construction. A notable consideration for Π_2 is that not only can the guardian effectively man-in-the-middle the connection between the edge and the group, but it is impossible for the edge to remove the guardian to regain underlying group control. Observe, exitGship relies on edge action to terminate the CKA channel with the guardian. This is an inherent issue to Π_2– any direct link between the edge and main group would change the protocol to function like e.g., Π_6.

Π_3 **– Shared Signature keys, Shared Rand.** Both the guardian and the edge share control of one anchor in this variant. Furthermore, we allow the edge to be aware of the guardian's internal choices through having shared randomness. We split Π_3 into two cases with separate constructions based on how the randomness is shared: Π_{3a} for a copy into active memory and Π_{3b} for a copy placed in secure hardware.

Remark 2. *If true random number generators are used separately in the edge and guardian, a shared random tape is infeasible. Instead, deterministically updated seed is shared between the two. Without the knowledge of the seed, key generation will look random, while access to the seed on either device implies compromise of both.*

Π_{3a} – *Unprotected Shared Randomness* In this subvariant, the shared randomness is copied into memory that is accessible to the adversary under a Corrupt(ID) query. Once an adversary corrupts the edge or guardian, both are corrupted since the randomness is a direct

duplication. Furthermore, the randomness, now in adversarial possession, is the same randomness used to generate new keys, the group only achieve PCS through the removal of both edge and guardian. GPCS can never be obtained.

Π_{3b} – *Protected Shared Randomness* In this subvariant, the shared randomness is copied into a secure hardware component and is kept separate from the active state, analogue to the protections placed on signature keys. Such memory is not accessible to the adversary under a $\mathsf{Corrupt}(\mathsf{ID})$ query, thus an adversary able to corrupt the state of the edge or guardian, would not get access to the randomness. PCS can therefore be achieved through a normal update process. Furthermore, GPCS is achieved since the guardian can issue updates to the rest of the group which the corrupted edge can generate locally. Since the update secret keying material does not need to be sent to the edge, an adversary is unable to gain access to the new group key even though they have a copy of the local state $\mathrm{S}[id_E]$.

Π_4 **– Distinct Signature Keys, Shared Randomness** Instead of allowing end users in the underlying protocol to register only a single authentication key pair, we allow multiple signing keys for a single CGKA node. By having the guardian and the edge device possess two distinct but registered signing keys the edge-guardian impersonation seen in Π_3 is impossible. However, this does require that CGKA members (in this case, MLS members), can associate more than one signing key to a given member.

By expanding the edge and guardian id to include the new signing keys the remaining protocol is identical to that presented in Figure 4.7 Π_{3a} and Π_{3b}. The new edge id and guardian with anchor ID will now be $id_E = (A, \mathsf{ID}, \cdot, \mathsf{sk}_A)$ and $id_G = (B, \mathsf{ID}, \cdot, \mathsf{sk}_B)$. The discussion about security levels from the previous sections remain.

Π_5 **– Shared Anchor and Signature key, Distinct Rand.** In Π_5, a guardian and edge share an anchor and signature key pair, as in Π_3. Unlike Π_3, however, the guardian and edge retain distinct individual states and randomness. This necessitates a separate, non-CGKA channel between edge and guardian, similar to Π_2, to share update information.

This option limits the fallout risk from edge compromise as an adversary does not automatically obtain the guardian state. Separation in to unprotected and protected randomness is unnecessary since distinct random tapes mean that both devices can use random number generators. GPCS is impossible, however, for the same reason. PCS and FS are maintained.

If a signature key is compromised, then both the edge and guardian can be impersonated.

Π_6 – Shared Anchor, Distinct Signature Key and Rand. Π_6 mirrors Π_4 except that distinct random tapes are used. As in Π_5, this solves issues relating to secure storage of a shared random tape, but also necessitates a separate channel between G and E. In our construction, this is a CKA channel.

4.7 Architectural comparison

Security considerations fall into two categories: protocol guarantees which can be analyzed using provable security, and properties that are affected based on architectural choices. A summary of the architectural-related properties associated with each protocol is shown in Table 4.1. Their associated overhead costs are shown in Table 4.2. Depending on use case, different features may be desirable.

Feature	Π_1	Π_2	Π_{3a}	Π_{3b}	Π_{4a}	Π_{4b}	Π_5	Π_6
Extensible to multiple edges	○	●	●	●	●	●	●	●
Guardian can be offline	●	○	●	●	●	●	●	●
Guardian removal without edge	●	○	○	○	●	●	○	●
Traceability of guardianship	●	○	◐	◐	●	●	◐	●
Low impersonation risk	●	○	○	○	●	●	○	●
FS	●	◐	●	●	●	●	●	●
PCS	●	◐	○	●	○	●	●	●
GPCS	○	○	○	●	○	●	○	○

Table 4.1. Architectural feature comparison. Green circles mean “Supported”, Unfilled circles mean “Unsupported”, and half-colored circles mean “It Depends”.

Expanding on the example posed in Section 4.1, a team of journalists are in a MLS chat group hosted by their company, including a remote reporter. The remote reporter is dealing with a degraded communication environment and suspects that they will be targeted for surveillance: the reporter’s device may be targeted when entering the region or in the hotel. Various considerations must be weighed in selecting the appropriate GCGKA construction for the scenario. Π_1 and Π_2 downgrades the FS and PCS window to the frequency of the reporter’s updates: if the phone is compromised en-route or thereafter, the group’s secrets

(or 1:1 secrets in Π_2) would be accessible by the adversary until the reporter updates. Of all other constructions, the traceability of the reporter is minimized most with Π_2 due to the secondary CKA channel. Π_{3a} and Π_{4a} should be avoided due to the impossibility of even obtaining PCS. Π_5 and Π_6 are also sub-optimal in this case due to the inability to reliably maintain or update the CKA channel between the guardian and edge. Π_{3b} and Π_{4b}, on the other-hand provide the best security with FS, PCS, and GPCS assurances even after state corruption.

Suppose, in contrast to the reporter/journalist team scenario, we have instead a scenario where the "reporter" is instead a whistle-blower or source such that limiting traceability is an important feature. In this case, Π_{3b} should be used over Π_{4b}, due to its ability to conceal the relative activity differences between the guardian and edge. That concealment would be considered a bug instead of a feature in other cases, where non-reputability is of greater interest. Namely, if a person is known to be on travel and a malicious commit message is sent from the edge or guardian, it is not possible to demonstrate which signing key was used. Option Π_{4b} allows for such differentiation and other group members may be able to recognize unexpected activity on the guardian vs. edge in case of compromise.

4.8 Security Analysis

What follows is an analysis of overall trade-offs among the proposed options. Due to space constraints the proofs can be found in Section 4.9. Theorem 3 forms a baseline check that the guardianship introduction has not affected the underlying security of the CGKA for the edge. As noted, Π_{3a} and Π_{4a} do not pass this check, while in Π_2 achieves FS and PCS only to the extent of the CKA channel between the edge and guardian – the edge is completely outside the FS and PCS scope of the group protocol.

Theorem 3. *[FS and PCS Security] Consider* Π_i*,* $i \in \{1, 2, 3b, 4b, 5, 6\}$*. Then* Π_i *is [FS,PCS]-GCGKA-secure secure under the CGKA security of the CGKA and the CKA security of the CKA. That is, for any PPT algorithm* $\mathcal{A}$ *against the [FS,PCS]-GCGKA security experiment,* $\mathsf{Adv}^{[\text{fs,pcs}]-gcgka}_{\Pi-\mathcal{A}}(\lambda)$ *is negligible.*

Theorem 4 demonstrates the viability of having the guardian issue updates on behalf of the edge. To achieve GPCS, a core assumption is that the shared randomness can be protected,

e.g., installed in protected hardware in the same way that long-term signing keys are or provided by the user, and are therefore out of scope of a Corrupt query. Thus, while GPCS is fundamentally a cryptographic notion, achieving it relies also on architectural decisions (see Section 4.7).

Theorem 4. *[GPCS Security] The protocols* Π_{3b} *and* Π_{4b} *are [FS,PCS,GPCS]-GCGKA-secure under the CGKA security of the CGKA and the CKA security of the CKA. That is, for any PPT algorithm* $\mathcal{A}$ *against the [FS,PCS,GPCS]-GCGKA security experiment,* $\mathsf{Adv}^{[\mathrm{fs,pcs,gpcs}]-gcgka}_{\Pi-\mathcal{A}}(\lambda)$ *is negligible.*

4.9 GCGKA Analysis

Recall the following theorems.

Theorem 3. *[FS and PCS Security] Consider* Π_i, $i \in \{1, 2, 3b, 4b, 5, 6\}$. *Then* Π_i *is [FS,PCS]-GCGKA-secure secure under the CGKA security of the CGKA and the CKA security of the CKA. That is, for any PPT algorithm* $\mathcal{A}$ *against the [FS,PCS]-GCGKA security experiment,* $\mathsf{Adv}^{[\text{fs,pcs}]-gcgka}_{\Pi-\mathcal{A}}(\lambda)$ *is negligible.*

Theorem 4. *[GPCS Security] The protocols* Π_{3b} *and* Π_{4b} *are [FS,PCS,GPCS]-GCGKA-secure under the CGKA security of the CGKA and the CKA security of the CKA. That is, for any PPT algorithm* $\mathcal{A}$ *against the [FS,PCS,GPCS]-GCGKA security experiment,* $\mathsf{Adv}^{[\text{fs,pcs,gpcs}]-gcgka}_{\Pi-\mathcal{A}}(\lambda)$ *is negligible.*

We provide sketch proofs for each of the different protocols.

4.9.1 Security Analysis of Π_1

Proof Theorem 3. Since Π_1 builds directly on CGKA protocol CGKA it follows directly from Definition 14 and Figure 4.6 that Π_1 is a GCGKA protocol.

To show that Π_1 is $[FS, PCS]-$GCGKA secure we need to show that any adversary against Π_1 has negligible advantage.

Let $\mathcal{A}$ be a PPT algorithm against Π_1, Figure 4.6, with advantage $\mathsf{Adv}^{[\text{fs,pcs}]-\text{gcgka}}_{\Pi_1-\mathcal{A}}(\lambda)$. We use $\mathcal{A}$ to create an adversary $\mathcal{B}_1$, Figure 4.8, against CGKA protocol CGKA with advantage $\mathsf{Adv}^{\text{cgka}}_{\text{CGKA}-\mathcal{B}_1}(\lambda) \geq \mathsf{Adv}^{[\text{fs,pcs}]-\text{gcgka}}_{\Pi_1-\mathcal{A}}(\lambda)$. We use MLS as CGKA, and since MLS has been shown to be CGKA secure [44] meaning that the advantage of $\mathcal{A}$ must be negligible, and by extension that Π_1 is $[FS, PCS]-$GCGKA secure.

When $\mathcal{A}$ issues a challenge, $\mathcal{B}_1$ will query the same challenge to $\text{Exp}^{\text{GCGKA}}_{\Pi_1}$, then pass the session key back to $\mathcal{A}$. Since $\mathcal{B}_1$ only does book-keeping before passing all the queries made by $\mathcal{A}$ onto $\text{Exp}^{\text{CGKA}}_{\text{CGKA}}$ $\mathcal{A}$ will not be able to distinguish the difference. A successful guess made by $\mathcal{A}$ will, therefore, result in a successful guess in $\text{Exp}^{\text{CGKA}}_{\text{CGKA}}$.

Finally, to ensure that any valid win for $\mathcal{A}$ is a valid win for $\mathcal{B}_1$ we need to make sure that if the CGKA safety predicate returns 1 then GCGKA safety predicate returns 1. Since we have

that $X = |FS, PCS|$ this follows directly from definition; [44], Section 4.5. As a result we get;

$$\mathsf{Adv}^{[\mathrm{fs,pcs}]-\mathsf{gcgka}}_{\Pi_1-\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\mathsf{cgka}}_{\mathrm{CGKA}-\mathcal{B}_1}(\lambda) < \mathsf{negl}(\lambda).$$

□

4.9.2 Security Analysis of Π_2

Proof Theorem 3. Since Π_2 builds directly on CGKA protocol CGKA it follows directly from Definition 14 and Figure 4.6 that Π_2 is a GCGKA protocol, the fact that an additional CKA communication channel exist does not interfere here.

To show that Π_2 is $[FS, PCS]-\mathsf{GCGKA}$ secure we need to show that any adversary against Π_2 has negligible advantage. We perform one game hop, Figure 4.9, using an adversary $\mathcal{B}_1$ and then reduce the problem to an CGKA adversary $\mathcal{B}_2$.

G0 - G1: The game hop between G0 and G1 exchanges the correct generation of CKA-session keys with randomly selected values. Let $\mathcal{B}_1$ be and adversary that can distinguish between G0 and G1. That means $\mathcal{B}_1$ is an adversary that can determine if k is a real or random CKA session key, i.e. $\mathcal{B}_1$ is an adversary against $\mathrm{Exp}^{\mathsf{CKA\text{-}PCS},t^*,\Delta_{\mathsf{CKA}}}_{CKA-\mathcal{B}_1}$. Depending on whom $\mathcal{B}_1$ challenges, edge or guardian, the key might be real or random, meaning the two games are identical, thus we get a factor $\frac{1}{2}$:

$$\frac{1}{2}\,|\Pr[G0] - \Pr[G1]| \leq \mathsf{Adv}^{\mathsf{cka\text{-}pcs}}_{\mathrm{CKA}-\mathcal{B}_1}(\lambda).$$

G1 - G2: Not relevant; the two games are identical.

For the final step we create an CGKA adversary $\mathcal{B}_2$ against CGKA, modeled as MLS, using $\mathcal{A}$ against Π_2 by running an internal version of G1 and forwarding CGKA queries to $\mathrm{Exp}^{\mathsf{CGKA}}_{\mathrm{CGKA}}$. Similar arguments for the validity of $\mathcal{B}_2$ against G2 as used in the previous section gives us.

$$\Pr[G2] \leq \mathsf{Adv}^{\mathsf{cgka}}_{\mathrm{CGKA}-\mathcal{B}_2}(\lambda).$$

As a result we get:

$$\mathsf{Adv}^{\mathsf{gcgka}-|\mathrm{fs,pcs}|}_{\Pi_2-\mathcal{A}}(\lambda) \leq 2\mathsf{Adv}^{\mathsf{cka\text{-}pcs}}_{\mathrm{CKA}-\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{cgka}}_{\mathrm{CGKA}-\mathcal{B}_2}(\lambda)$$

□

4.9.3 Security Analysis of Π_3

We will only give the proof for Π_{3b} the proof for Π_{4b} follows exactly with only a different construction of the edge and guardian id.

Proof of Theorem 4. Since Π_{3b} builds directly on CGKA protocol MLS it follows directly from Definition 14 and Figure 4.6 that Π_{3b} is a GCGKA protocol, the fact that an additional randomness exist and is utilized during run time does not interfere here.

To show that Π_3 is $[FS, PCS, GPCS]-$GCGKA secure we need to show that any adversary against Π_{3b} has negligible advantage. We perform one game hop, Figure 4.9, using an adversary $\mathcal{B}_1$ against KDF and then reduce the problem to a CGKA adversary $\mathcal{B}_2$.

G0 - G1: Not relevant; the two games are identical.

G1 - G2: The game hop between G1 and G2 exchanges the KDF derivation of key k with true randomness. Let $\mathcal{B}_1$ be and adversary that can distinguish between G1 and G2, i.e. an adversary against real or random (ROR), and as in the previous section only the case of update is changed leaving us with a factor $\frac{1}{2}$.

$$\frac{1}{2}\,|\Pr[G1] - \Pr[G2]| \leq \mathsf{Adv}^{\mathrm{ror}}_{KDF-\mathcal{B}_1}(\lambda).$$

For the final step we create an CGKA adversary $\mathcal{B}_2$ against CGKA using $\mathcal{A}$ against Π_{3b} by running an internal version of G2 and forwarding CGKA queries to $\mathrm{Exp}^{\mathsf{CGKA}}_{\mathrm{CGKA}}$.

To determine the validity of the win we are not able to rely on the argument made in the previous sections. If forward secrecy holds in $\mathcal{A}$ then it must naturally hold for $\mathcal{B}_2$ as old randomness and keys are deleted after session. The problem occurs with PCS and GPCS; if

$\mathcal{A}$ reveals the state of an edge or a guardian and the new state and by extension group key is created deterministically from the previous state PCS will not be achievable. This is the case for Π_{3a} (Π_{4a}). However if the randomness used for deriving the next state/key is kept separate and secure, then after one update preformed by the revealed entity will result in fresh randomness, in other words PCS is achieved. The latter scenario is that presented in Π_{3b} (Π_{4b}).

Similarly if an edge is corrupted and then the guardian updates, new randomness will be introduced as long as the randomness is kept separate. Meaning Π_{3b} (Π_{4b}) achieves GPCS. This gives us the following advantage against G2:

$$\Pr[G2] \leq \mathsf{Adv}^{\mathsf{cgka}}_{\mathrm{CGKA}-\mathcal{B}_2}(\lambda).$$

As a result we get:

$$\mathsf{Adv}^{\mathsf{gcgka}-|\mathrm{fs,pcs}|}_{\Pi_2-\mathcal{A}}(\lambda) \leq 2\mathsf{Adv}^{\mathrm{ror}}_{KDF-\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathsf{cgka}}_{\mathrm{CGKA}-\mathcal{B}_2}(\lambda).$$

□

4.9.4 Security Analysis of Π_5 and Π_6

Similarily as in Section 4.9.3 we will only give the proof for Π_5 the proof for Π_6 follows exactly with only a different construction of the edge and guardian id.

Proof Theorem 3. Note that the one of the main differences between Π_5 and Π_2 is that in the latter the edge has access to the CGKA signing key. When the edge device is in limited mode it will not be able to follow group actions directly but will be reliant on the guardian forwarding messages. Only when the edge is online is there a notable difference. In that case the edge will update the group itself instead of relying on the guardian to preform the action. The proof for Theorem 3 for Π_5 is, therefore, as a result almost identical to the given in Section 4.9.2.

Since Π_5 builds directly on the CGKA protocol MLS it follows directly from Definition 14

and Figure 4.6 that Π_5 is a GCGKA protocol, the fact that an additional randomness exist and is utilized during run time does not interfere here.

To show that Π_5 is $[FS, PCS]-$GCGKA secure we need to show that any adversary against Π_5 has negligible advantage. We perform one game hop, Figure 4.9, using an adversary $\mathcal{B}_1$ against CKA and then reduce the problem to an MLS adversary $\mathcal{B}_2$.

G0 - G1: As stated earlier, the game hop between G0 and G1 exchanges the correct generation of CKA-session keys with randomly selected values. Let $\mathcal{B}_1$ be and adversary that can distinguish between G0 and G1. That means $\mathcal{B}_1$ is an adversary that can determine if k is a real or random CKA session key

$$\frac{1}{2}|\Pr[G1] - \Pr[G2]| \leq \mathsf{Adv}^{\text{cka-pcs}}_{\text{CKA}-\mathcal{B}_1}(\lambda).$$

G1 - G2: Not relevant; the games are identical.

For the final step we create an CGKA adversary $\mathcal{B}_2$ against MLS using $\mathcal{A}$ against Π_5 by running an internal version of G2 and forwarding CGKA queries to Exp^{CGKA}_{MLS}.

The validity argument presented in Section 4.9.1 holds here as well.

This gives us the following advantage against G2:

$$\Pr[G2] \leq \mathsf{Adv}^{\text{cgka}}_{\text{CGKA}-\mathcal{B}_1}(\lambda).$$

As a result we get:

$$\mathsf{Adv}^{\text{gcgka}-|\text{fs,pcs}|}_{\Pi_2-\mathcal{A}}(\lambda) \leq 2\mathsf{Adv}^{\text{cka-pcs}}_{\text{CKA}-\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\text{cgka}}_{\text{CGKA-}\mathcal{B}_2}(\lambda).$$

□

4.10 Features Discussion

Connectivity: In the edge-guardian scenario, it is assumed that the edge may be in limited/receive-only mode for extended periods of time. During that time, it is expected that the guardian can provide updates – this, however, raises the question of whether the guardian *must* be online as a functionality requirement.

With the exception of Π_2 none of the proposed protocols require both an online guardian and edge device to function. In the case of Π_2, the edge is entirely dependent on the reliable connectivity of the guardian. If, e.g., the user is traveling with the edge device and left the guardian in a secure location at home, this could come with network delays as the communication must route through the guardian. Also in Π_2 it is impossible for the edge device to participate in the group independently of the guardian even if online.

Guardian Removal: Device removal has differing effects across the protocol options, meaning that exitGship results in differing behaviors. In Π_1, a guardian can be removed from the group without affecting its edge. The same holds for Π_4 and Π_6. In contrast, removal of the guardian in Π_2, Π_3, or Π_5 will necessarily incur the removal of the edge. For Π_2 the guardian is the gateway to the group for the edge while in Π_3 and Π_5 the guardian and edge share a signing key – thus removal of the signing key identity automatically incurs removal of the edge.

If an attempt is made in Π_{3a}, Π_{3b}, or Π_5 to remove a guardian while maintaining the edge's access to the group, the guardian will still be able to decrypt messages in the case of either Π_{3a} or Π_{3b}, and can impersonate an update and regain access while locking out the edge in the case of Π_5.

Traceability: In some protocols, other group members may be aware of the guardianship and can trace the online/offline status of the guardian or limited mode for the edge device. In other protocols, the DS must have knowledge.

In the case of Π_1, Π_4 and Π_6 the edge and guardian have separate signature keys used when preforming an update into the MLS group. As such, any group member will be able to differentiate between edge and guardian, and thus determine the operational mode in play.

In the remaining cases signature keys are identical for both guardian and edge. Thus an

arbitrary group member will not be able to determine which of the two entities is online.

The *distribution service (DS)*, however, being used to deliver data between members in the correct order, could trace the edge/guardian relationship and online/offline status – if a single DS is used. Under Π_2 it is possible to implement the channel between the edge and guardian using a separate DS from the CGKA, as no coordination with the CGKA updates is necessary. In the case of Π_3 and Π_5, the DS must have awareness of the edge/guardian pairing in order to deliver messages. It may also be aware of online/offline statuses. This is dependent on a push or pull design of the DS. The DS must be able to distinguish between the edge and guardian in order to send them both messages in a push context, whereas the edge-guardian relationship can be more obscured to the DS in a pull context.

As a core group member, it is not a requirement that the DS is aware of the guardian existence or status; however, a lack of awareness implies that the guardian is privy to all application messages. In other words, either the DS delivers messages to the guardian based on activity status, or the DS always delivers messages to the guardian.

Notable Changes	Π_1	Π_2	Π_{3a}	Π_{3b}	Π_{4a}	Π_{4b}	Π_5	Π_6
Increased Tree Size	✗							
Preloaded, Secure Storage of Shared Random Seed			✗	✗	✗	✗		
Additional Channel Maintenance		✗					✗	✗
Additional Per-Device Signature Key in MLS					✗	✗		✗

Table 4.2. Summary of changes from baseline MLS across various constructions.

Extensibility: A natural question on guardianship is the extensibility to place multiple edge devices with a single guardian. While we do not model this under the GCGKA experiment, it is not unreasonable to consider the option. Π_1 loses the sibling connection if further edges are added. Π_2 can easily be extended. Likewise, Π_3, Π_4, Π_5, and Π_6 can have further edges added to the anchor with a single guardian.

There are security costs to some such arrangements – in Π_3 and Π_4, for example, the same random tape would necessarily be shared among all such edge devices as well as the

guardian, creating more opportunity for compromise. In the case of Π_{3a}, Π_{4a}, where the shared randomness is kept in active memory, this is especially unsafe. Even in Π_{3b} and Π_{4b}, where the shared randomness is kept in protective memory, any one edge is still at risk from compromise of another as the same assumption do not apply to edges as guardians (i.e., that the guardian is kept in a secure location). In contrast, Π_1 does not present such hazards among edge devices.

Impersonation Risk: There are a number of straight-forward observations that can be made on the comparative risk of a guardian impersonating an edge device. These are in part dependent on whether application messages are signed or only group control messages – a choice dependent on the underlying CGKE, e.g., MLS or other.

In Π_1, Π_{4a}, Π_{4b}, and Π_6, the edge device and guardian device will have distinct signing keys and will not have any knowledge of the others keying material. This assures non-repudiation. Impersonation detection by other group members, as well as by the edge and guardian device may therefore be possible. This is true for both tree updates as well as regular message transmission.

For Π_2, Π_{3a}, Π_{3b}, and Π_5, the guardian possesses a copy of the edge's signing key or acts as the anchor relay, and can therefore impersonate the edge device – in fact, impersonation is a requirement for normal functionality. This means that guardianship is opaque to other group members but comes at the cost of non-repudiation (for Π_{3a}, Π_{3b}, and Π_5) as well as an increased reliance on the honesty/security of the guardian.

Costs: Each protocol presented has some additional cost pertaining to storage, computation, or additional hardware. Table 4.2 gives an overview of notable changes based on each protocol. Further calculation can show costs for specific cases. These should be taken into account when compared to possible improvements, as seen from Table 4.1.

With the exception of Π_1, all the other proposed protocols use the same anchor for both guardian and edge device. This means that in order for Π_1 to function, an additional MLS leaf node needs to be added per edge device; resulting in a greater tree of depth. Since the depth of the MLS tree directly correlates to the number of encryptions and decryptions needed to be preformed for each update, Π_1 would result in one encryption/decryption per update.

For protocols Π_4 and Π_6 the tree size does not need to be altered but an extension change to MLS is required. Specifically, the allowance of multiple authentication keys to a singular MLS member is required. Π_4 additionally needs extra key management to load the shared seed, a requirement also found in Π_3. In Π_{3b} and Π_{4b}, secure hardware requirements also needs to be met. While need for shared randomness is not found in Π_6, it still requires additional key management in order to keep a separate communication channel between edge and guardian. This similarly applies to Π_2 and Π_5.

4.11 Conclusion

While prior work has considered PCS to be uniquely tied to the ability for a compromised node to issue key updates, we have investigated the possibility for a designated third party to perform this function on the behalf of vulnerable members through GPCS. Achieving GPCS is non-trivial. Both it and the overall security of the protocol are highly dependent on a number of system architectural choices, and such decisions can introduce trade-offs. As shown in Section 4.8, among the design space options, only Π_{3b} and Π_{4b} achieve GPCS security. Given the options listed, Π_{4b} nominally demonstrates the best selection of properties, environment and architecture contingent.

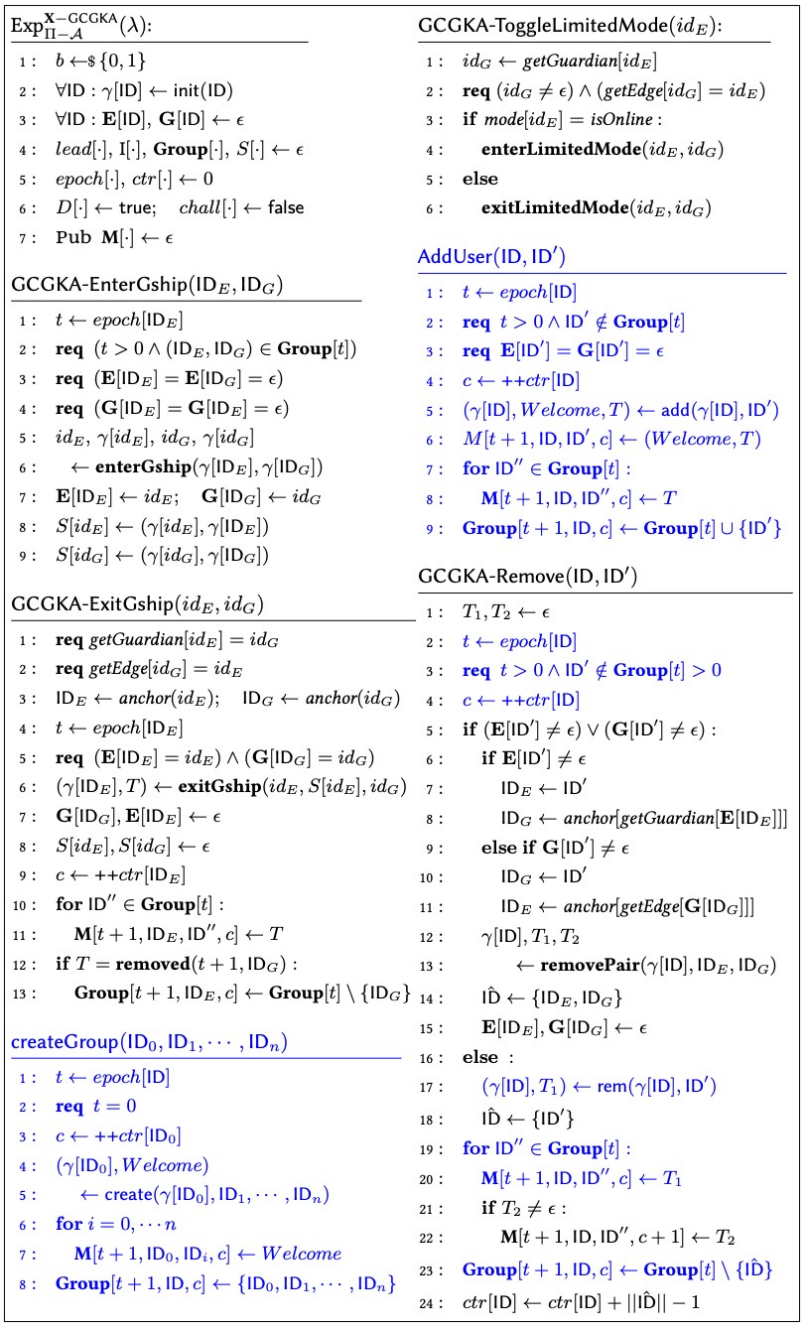


Figure 4.2. Oracles for the GCGKA security game for a scheme GCGKA = (enterGuardianship, exitGuardianship, removePair, enterSilentMode, exitSilentMode, updateED, updateGuard,processGP). blue text indicates CGKA experiment syntax and black indicates new or altered components. Algorithms and sets are indicated in bold font and variables in italics.

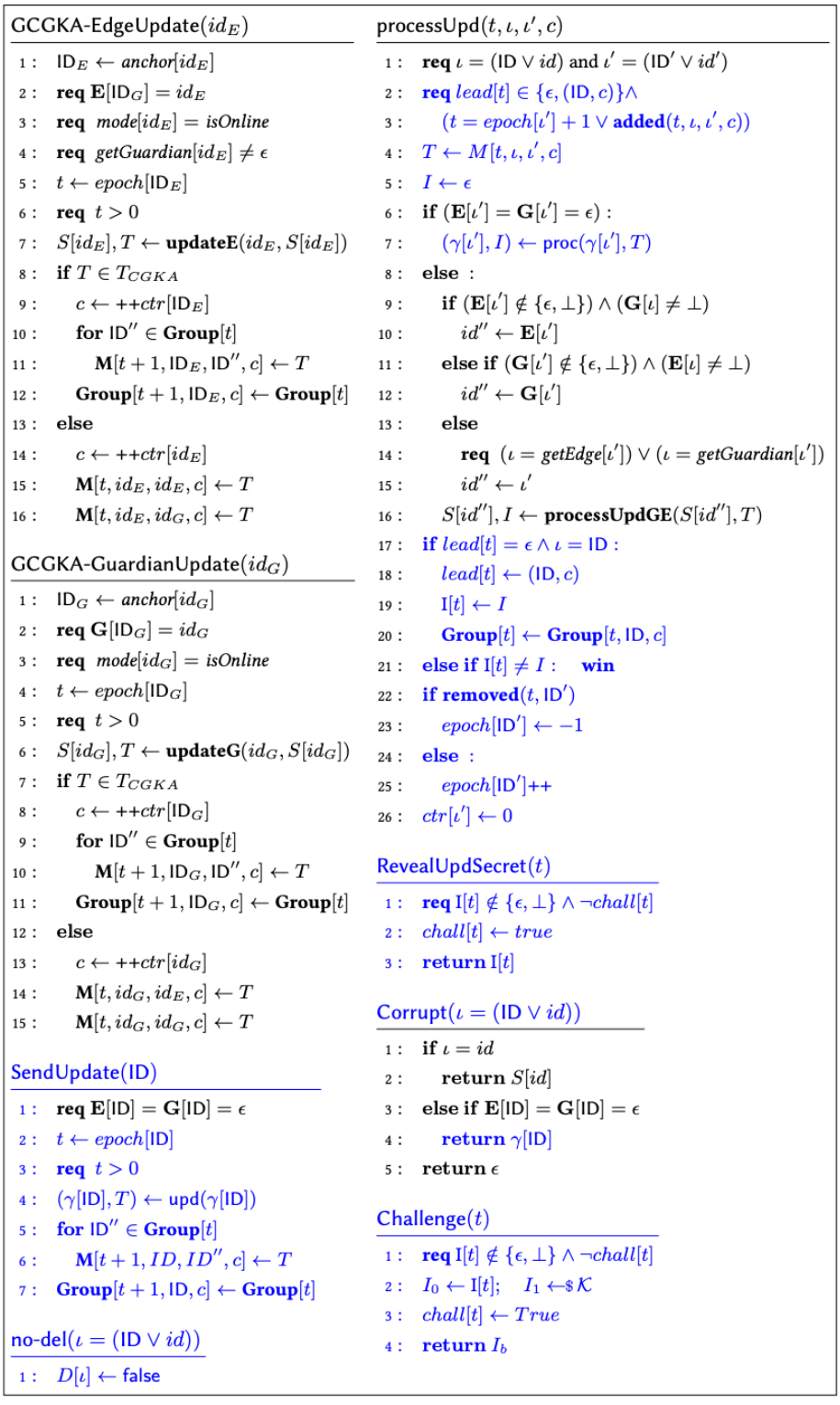


Figure 4.3. GCGKA experiment continued.

for $\iota = \mathsf{ID} \vee id$, let

1 : $Q_{GPCS}(\iota) = \{\mathsf{GCGKA\text{-}GuardianUpdate}($
$getGuardian[\iota])\}$

2 : $Q_{FS,PCS}(\iota) = \{\mathsf{GCGKA\text{-}EdgeUpdate}(\iota),$
$\mathsf{GCGKA\text{-}Remove}(*, anchor[\iota]),$
$\mathsf{GCGKA\text{-}Remove}(*, \iota),$
$\mathsf{SendUpdate}(\iota),$
$\mathsf{GCGKA\text{-}ExitGship}(\iota, *)\}$

3 : $Q_{FS,PCS,GPCS}(\iota) = Q_{GPCS} \cup Q_{FS,PCS}.$

(a) The list of experiment queries the different security notions allow.

X-safe$(q_1, \cdots, q_q)$

1 : **for** $\big((i,j)\ s.t. \mathbf{q}_i = \mathsf{Corrupt}(\iota) : \iota = (\mathsf{ID} \vee id)$
$\wedge\ \mathbf{q}_j = \mathbf{chall}(t^*)$ *for some* $t^*\big)$:

2 : **if** $\mathsf{q2e}(\mathbf{q}_i) \leq t^* \wedge \nexists k$ s.t.
$(0 < \mathsf{q2e}(\mathbf{q}_i) < \mathsf{q2e}(\mathbf{q}_k) \leq t^*)$
$\wedge \big((\mathbf{q}_k \in Q_{\mathbf{X}}) \vee (\iota = id_G \wedge (\mathbf{q}_k \in Q_{GPCS}))\big)$

3 : **return** 0

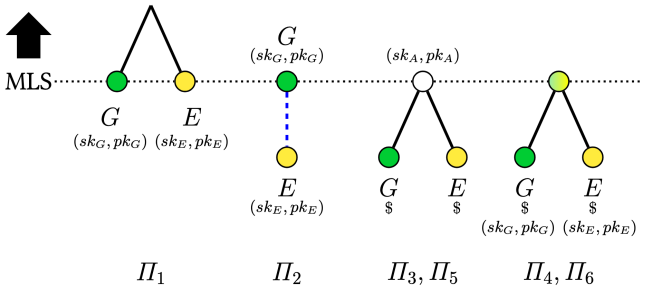
4 : **if** $\mathsf{q2e}(\mathbf{q}_i) > t^*$
$\wedge\ \exists\ k$ s.t. $\mathsf{q2e}(\mathbf{q}_k) \leq t^* \wedge \mathbf{q}_k = \mathsf{no\text{-}del}(\iota)$

5 : **return** 0

6 : **return** 1

(b) Safety predicate, where $\mathbf{X} \in \{[FS, PCS], [FS, PCS, GPCS]\}$.

Figure 4.4. **X-safe** adapted from [44], prevents trivial attacks by an attacker.



(a) Guardian GCGKA Π constructions in the context of MLS.

Shared Property			
Anchor	(sk, pk)	**$**	Π
0	0	0	1
0	0	1	(a)
0	1	0	(b)
0	1	1	(c)
1	0	0	2 / 6
1	0	1	4a/b
1	1	0	5
1	1	1	3a/b

(b) Truth table of design choices.

Figure 4.5. GCGKA protocol options and truth table of design choices with omitted options (a),(b), and (c). Π_3 and Π_4 are separated into a/b cases depending on if the shared randomness is kept in active memory (e.g., part of $\gamma[id_E]$) or pre-installed in a secure module as may be done with signature keys. Options 5 and 6 mirror 3 and 4 except with distinct random tapes (this necessitates an external channel between G and E for both 5 and 6).

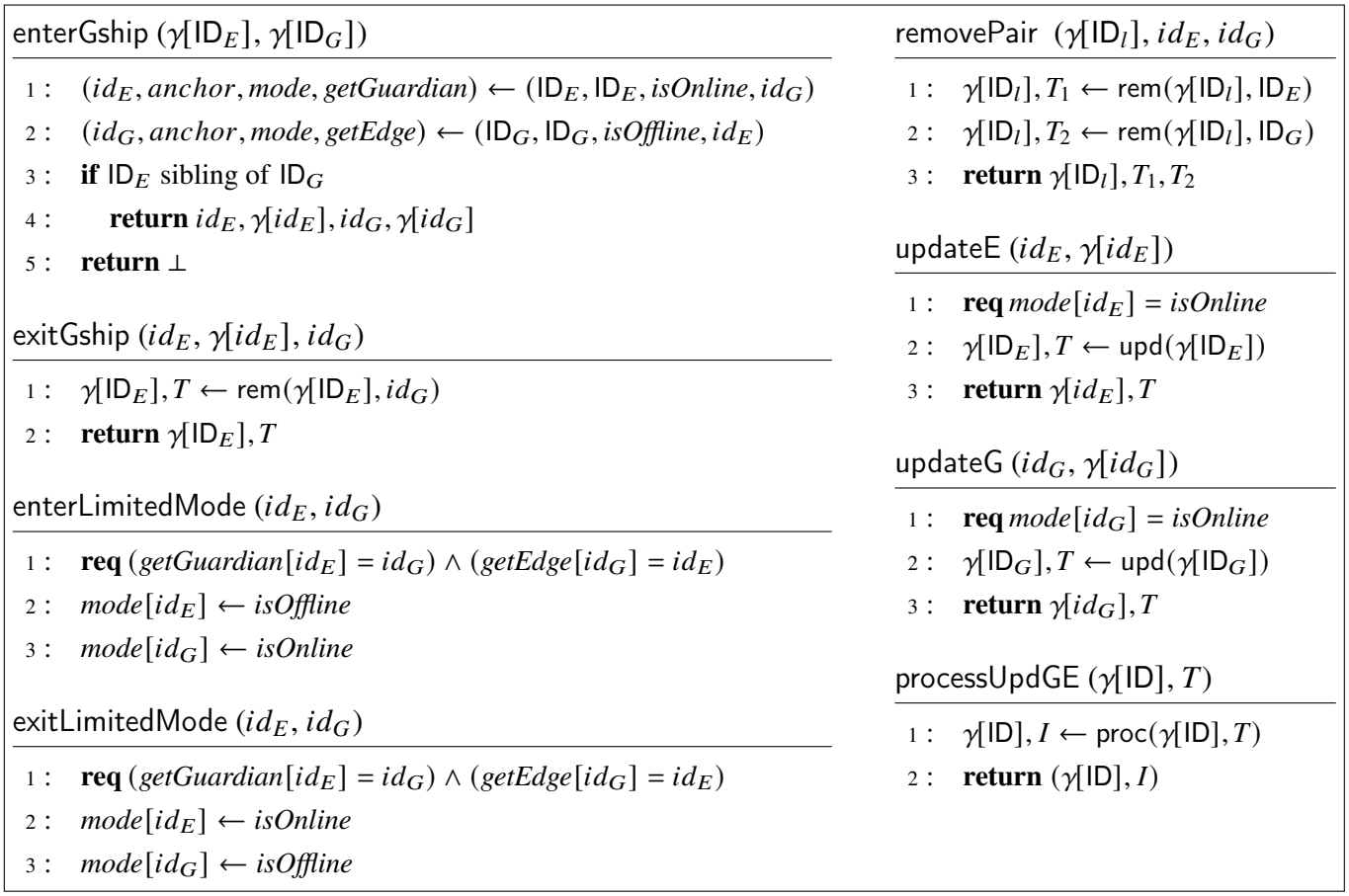


Figure 4.6. Construction of GCGKA Π_1.

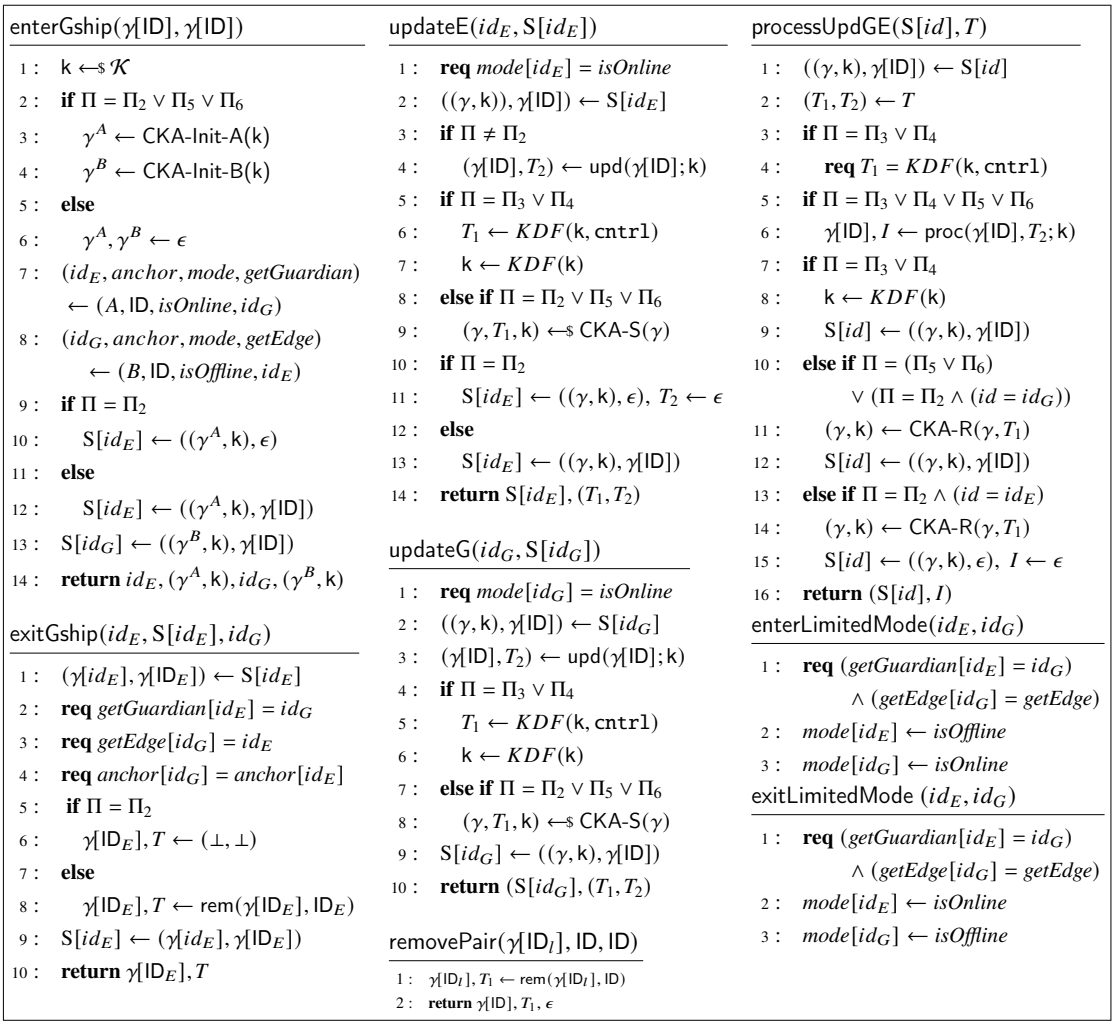


Figure 4.7. Constructions GCGKA protocols Π_2, Π_3, Π_4, Π_5 and Π_6.

$\mathcal{B}_1$ against $\text{Exp}^{\mathsf{CGKA}}_{\text{CGKA}}$.

Run an internal version of $\mathcal{A}$.
When $\mathcal{A}$ outputs b output b.
if $\mathcal{A}$ sends a GCGKA-EnterGship($\mathsf{ID}_E, \mathsf{ID}_G$) query:
  if $\mathsf{ID}_E, \mathsf{ID}_G$ are group members.
    Record pair ($\mathsf{ID}_E, \mathsf{ID}_G$, true).
if $\mathcal{A}$ sends a GCGKA-ExitGship(id_E, id_G) query:
  if $(id_E, id_G, \cdot)$ is recorded.
    Send a CGKA $remove(id_E, id_G)$ to $\text{Exp}^{\mathsf{CGKA}}_{\text{CGKA}}$.
if $\mathcal{A}$ sends a GCGKA-ToggleLimitedMode(id_E) query:
  if (id_E, ID, x) is recorded for some id ID and boolean value x.
    Record $(id_E, \mathsf{ID}, \neg x)$.
if $\mathcal{A}$ sends a GCGKA-Remove($\mathsf{ID}, \mathsf{ID}'$) query:
  Send $remove(\mathsf{ID}, \mathsf{ID}')$ to $\text{Exp}^{\mathsf{CGKA}}_{\text{CGKA}}$.
  if $(\mathsf{ID}', \mathsf{ID}'')$ or $(\mathsf{ID}'', \mathsf{ID}')$ is recorded for some id ID''.
    Send $remove(\mathsf{ID}, \mathsf{ID}'')$ to $\text{Exp}^{\mathsf{CGKA}}_{\text{CGKA}}$.
    Delete recording.
if $\mathcal{A}$ sends a GCGKA-EdgeUpdate(id_E) query:
  if $(id_E, \mathsf{ID}, \text{true})$ is recorded for some id ID.
    Send update(id_E) to $\text{Exp}^{\mathsf{CGKA}}_{\text{CGKA}}$.
if $\mathcal{A}$ sends a GCGKA-GuardianUpdate(id_G) query:
  if $(\mathsf{ID}, id_G, \text{false})$ is recorded for some id ID.
    Send update(id_G) to $\text{Exp}^{\mathsf{CGKA}}_{\text{CGKA}}$.
for any other query $\mathcal{A}$ sends.
  Forward the query to $\text{Exp}^{\mathsf{CGKA}}_{\text{CGKA}}$.
Forward any received/updated information returned from $\text{Exp}^{\mathsf{CGKA}}_{\text{CGKA}}$ to $\mathcal{A}$.

Figure 4.8. Reduction of a $[FS, PCS]-\mathsf{GCGKA}$ secure adversary against Π_1

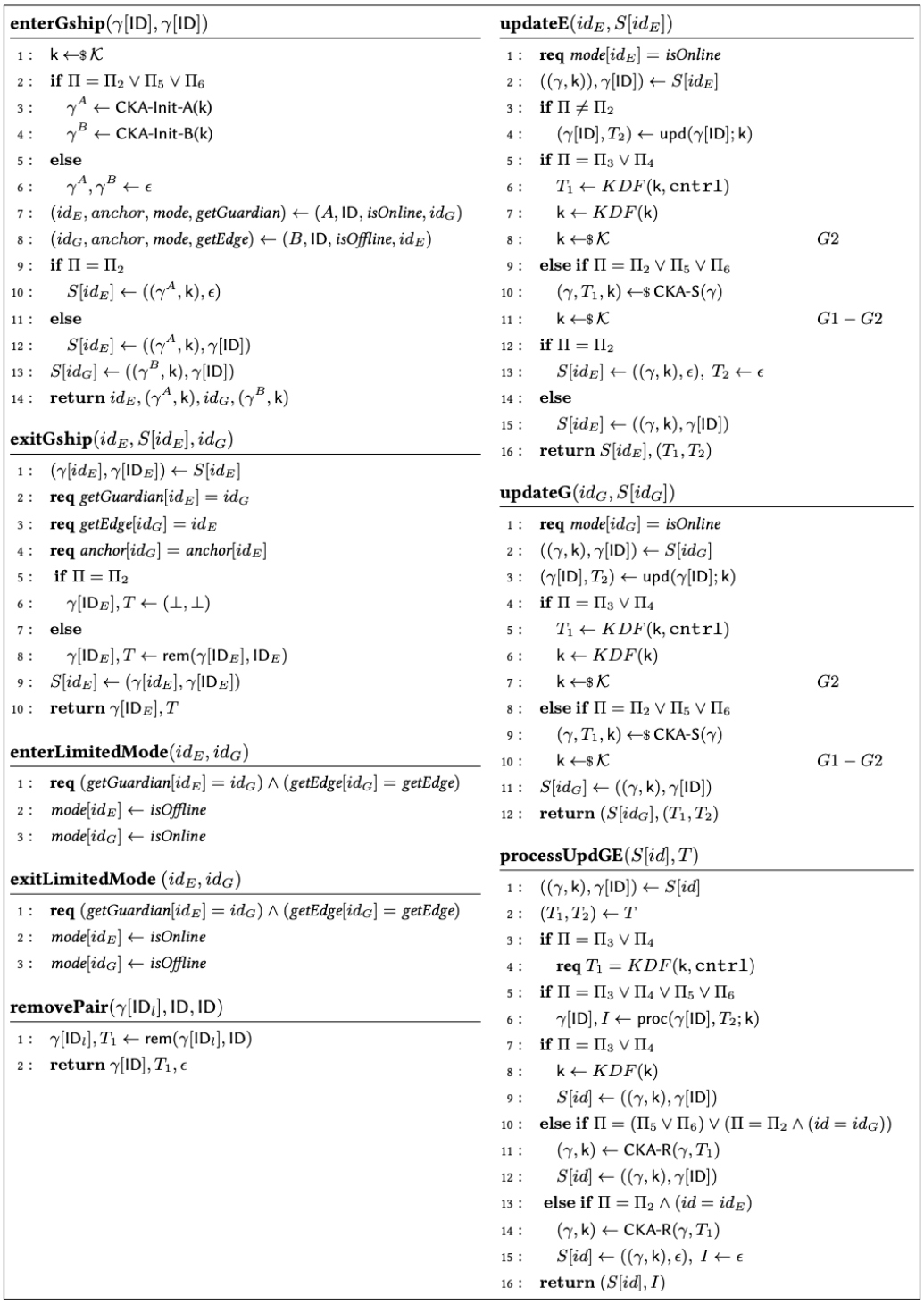


Figure 4.9. Game hops G0 and G1 used for proofs of Theorem 3 and Theorem 4 for protocols Π_2- Π_6.

Part III

Design, Analysis, and Implementation of Key Exchange for Space Settings

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Key Establishment in the Space Environment

Addressing the rigidity of traditional space security

As established in Chapter 1, traditional space security architectures suffer from fundamental rigidity: reliance on preshared keys and inflexible networking stacks prevents interoperability, inhibits recovery from compromises, and cannot adapt to evolving mission requirements. Space systems are typically launched with hardware crypto devices providing preshared keys for communication through bent-pipe architectures. This rigidity makes it difficult or impossible to heal from compromises, hinders efficient scaling of networks, and requires mission planners to pre-negotiate all possible secure communication paths—a process taking months of manual coordination that leaves systems brittle to any changes. Future space architectures like NASA's Lunanet will employ both Internet Protocol (IP) and Delay Tolerant Network (DTN) protocols across heterogeneous networks. Without a unified, dynamic key agreement approach tailored to both network stacks, space security will remain inefficient, ad-hoc, and fragmented, thereby perpetuating the very rigidity problems that constrain current operations.

Pillar II: Design, Analysis, and Implementation of Key Exchange for Space Settings. Recall from Chapter 1 that Pillar II addresses the rigidity problem by integrating modern dynamic and asynchronous key agreement mechanisms into the two dominant network architectures for space communications: Internet Protocol (IP) networks and Delay Tolerant Networks (DTN). While terrestrial security protocols like Transport Layer Security (TLS) and QUIC excel in low-latency, continuously-connected environments, they degrade or fail entirely when confronted with the long propagation delays, intermittent connectivity, and limited bandwidth characteristic of space links. Conversely, while DTN architectures are explicitly designed to tolerate delay and disruption, they have historically lacked integrated, standards-based key agreement mechanisms, relying instead on out-of-band key distribution or hardware-based preshared keys that inhibit operational flexibility and security resilience. Under Pillar II, QUIC-MLS transforms QUIC from a one-to-one synchronous handshake protocol into an asynchronous, many-to-many continuous group key agreement and secure

channel protocol, and BPSec-MLS introduces the first standardized, in-band key agreement mechanism integrated directly into DTN security. Together, these integrations represent the first comprehensive suite of standards-based, provably secure, and operationally validated key agreement solutions purpose-built for space environments.

Chapter Contribution and Impact. This chapter provides the foundational analysis that motivates and guides the subsequent works implementing and testing MLS integration into both BPSec and QUIC (detailed in Chapter 6 and Chapter 7, respectively). For the first time, this work identifies and addresses the fundamental challenge of dynamic key agreement in both DTN and IP settings for space communications, proposing a unified solution that works across heterogeneous network architectures.

At the highest level, this work marks a revolutionary shift in network security by modularizing key agreement to be tailorable to operational needs rather than hard-coded into specific protocol stacks. By proposing Messaging Layer Security (MLS)—the only standardized continuous key agreement protocol—as a common foundation across both QUIC (IP) and BPSec (DTN), this work enables unprecedented operational flexibility. Future missions can dynamically establish secure communications with previously unknown peers after launch, recover from key compromises without physical intervention, and adapt group membership as satellites, ground stations, and relay nodes come online or go offline. The unified use of MLS across heterogeneous network stacks provides interoperability: a satellite can seamlessly switch between IP and DTN modes while maintaining continuous security, and multi-stakeholder missions (NASA, ESA, commercial partners) can conduct ad hoc multilateral operations without pre-coordinating cryptographic material.

At the protocol level, this chapter outlines the critical gaps and issues with the key agreement components (or lack thereof) in QUIC and BPSec and proposes the use of a stateful continuous key agreement protocol to address them. For QUIC, the work identifies that standard QUIC's synchronous, stateless, two-party TLS handshake is fundamentally misaligned with space operations involving diverse constellations, ground station networks, and intermittent connectivity windows. For BPSec, the work identifies the complete absence of standardized in-band key agreement mechanisms, forcing reliance on out-of-band key provisioning that perpetuates the rigidity problems outlined above. By championing the standardized MLS protocol as the most viable ready-to-deploy solution for both protocols, this work provides

a concrete pathway from the current fragmented landscape to a unified, flexible security architecture.

At the technical level, this work provides the first detailed analysis of how to integrate MLS into leading space communications secure channel protocols. Specific merits and considerations for MLS integration as a key agreement component to each channel protocol are discussed, including: alignment of MLS's asynchronous key updates with DTN's store-and-forward architecture; exploitation of MLS's post-compromise security to enable self-healing after compromises; adaptation of MLS's group key agreement to support multiparty space communications; and reconciliation of MLS's delivery requirements with the guarantees provided by QUIC and BPSec respectively. This foundational analysis establishes the theoretical background and integration strategies that the subsequent implementation and evaluation chapters build upon.

Real-World Impact. This work has been presented at the IEEE Space Mission Challenges for Information Technology and IEEE Space Computing Conference (IEEE SMC-IT/SCC) Space Cyber workshop, receiving feedback from space mission designers, satellite operators, and security researchers. The analysis provided here directly informed the design decisions in the BPSec-MLS and QUIC-MLS implementations, ensuring that the integration strategies address real operational constraints faced by space missions. By establishing MLS as a common key agreement foundation across both IP and DTN protocols, this work provides the architectural blueprint for the next generation of space communications security—one that replaces rigid, preshared-key architectures with flexible, self-healing, dynamically-adaptable security mechanisms suitable for the evolving demands of space operations.

Joint Work and Contributions. This chapter presents joint work with Benjamin Dowling (Kings College London), Britta Hale (Naval Postgraduate School), and Bhagya Wimalasiri (Kings College London) on a paper titled "Key Establishment in the Space Environmen."[12] The use of MLS for space communications and its integration into IP protocols was prior

[12]Reprinted, with permission, from all authors. 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. IEEE will claim and protect its copyright in international jurisdictions where permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

work by Britta Hale. The candidate's contributions include original proposals for specific integration of MLS into BPSec and QUIC. The candidate wrote the informal analysis portions of the paper and a first full draft of the paper. The candidate performed further refinement motivations and editing as joint work. The candidate presented the paper at the Space Cyber workshop at the IEEE Space Mission Challenges for Information Technology and IEEE Space Computing Conference (IEEE SMC-IT/SCC).

Chapter Abstract

As reliance on space systems continues to increase, so does the need to ensure security for them. However, public work in space standards have struggled with defining security protocols that are well tailored to the domain and its risks. In this work, we investigate various space networking paradigms and security approaches, and identify trade-offs and gaps. Furthermore, we describe potential existing security protocol approaches that fit well into the space network paradigm in terms of both functionality and security. Finally, we establish future directions for enabling strong security for space communication.

5.1 Introduction

For the goal of scientific exploration, space systems have often been considered low-value from a cyber-threat standpoint, and virtually inaccessible. This led to early communication security for space systems being overlooked as largely irrelevant – after all, what value would there be in hacking a Mars rover? The security that was considered was relatively primitive, based on symmetric key approaches such as encryption only, without key establishment [70], [71]. As such, it is unsurprising that there has been limited development in space communications security over the last several decades in comparison to the immense body of literature designing and analyzing protocols for terrestrial networks. However, space systems are increasingly prevalent in industrial use and even relied upon for daily mundane tasks. SpaceX's revolutionary reusable rockets came to market in the 2010s [72] and the proliferation of near-space satellites as an Internet technology has revolutionized use of non-terrestrial platforms and possibilities. Space based Internet providers [73], tele-health [74], space tourism [75], asteroid mining [76] and a host of other ventures have developed that continue to expand reliance on space – and its security – into people's daily lives [77]. Now, everything from banking information to critical infrastructure management flows through

space connections. Public safety, health, financial transactions are all high-value targets and motivate attacks against space communications [78]. Space systems now require that which has never before been an intrinsic goal: secure channel establishment.

Secure channels are frequently defined by encryption and authentication, ensuring that the data sent and received remains private and unaltered. Such cryptographic functions require secret keys (either symmetric or asymmetric). Some initial security approaches applied manually-installed, pre-shared keys which are among approaches recommended by the Consultative Committee for Space Data Systems (CCSDS) [11], [79]. This approach has naturally limited scaling, not to mention exceedingly long vulnerability windows due to the typical service lifetime of a system and infeasibility of updating systems after launch. Consequently, other approaches have attempted to project terrestrial network protocols onto space systems [80], [81], which would allow for ad-hoc session establishment. Yet, space is not a terrestrial environment: delay times in transmission imply an even higher delay cost from key establishment protocols that require 1.5-2 round trips. If such protocols operate point-to-point (e.g., link layer), they introduce a further risk of data being in the clear at every network 'hop', i.e., space or ground terminal. Furthermore, packet dropping and link unavailability could incur retransmission attempts. For protocols that require an entire new session establishment after failure, further precious power and bandwidth resources are wasted. In addition, the threat of quantum computing towards current cryptographic key establishment approaches necessitates use of new cryptographic primitives for key establishment and authentication, which frequently come with larger ciphertexts and public keys. This raises a fundamental question:

What key establishment method is appropriate
for current space architecture approaches?

Adaptation of key exchange protocols to their use case is not a new endeavor. Key exchange for the Internet of Things (IoT) has been customized for similar reasons [82]–[85], due to use case and environment requirements. Like the IoT setting, simply "gluing" key establishment protocols designed for Internet use cases into the space setting results in suboptimal operation at best. Internet protocols can rely on consistent, low-latency bidirectional connections which simply does not apply to the space setting. Cellular communication, such as those

standardized by 3GPP [86] for 5G, similarly does not rely on common Internet communication key establishment protocols. Thus, key establishment protocols for space systems should consider the unique characteristics of the setting, including delays, relay reliance, intermittent link availability and high latency, properties that are further exacerbated by distance, space weather, and system power limitations.

In this work, we look at two current approaches to space networking and the role of key establishment in each. We summarize Delay Tolerant Networking (DTN), designed for deep space interplanetary communications centered on the store-and-forward paradigm, using the Bundle Protocol, and open questions on key establishment therein. We also discuss current trends using IP-based protocols such as QUIC within the space setting, which is widely adopted for internet applications like Facebook [87], YouTube [88], and Instagram [87], and observe ill-fitting characteristics of its key establishment approach for space systems. Finally, we discuss Continuous Key Agreement (CKA) protocols – a class of key establishment protocols suited to space environments. These have been previously proposed, being apparently first publicly proposed in 2021 [89] and later also described in [90]–[92]. However, although prior work describes the benefits of CKAs in the space setting, how it can be incorporated into current efforts for space protocol stacks has not been described. This work takes a closer look at that gap, showing how CKAs can be incorporated into *either* the DTN or IP-based transport stacks for ensuring secure channels as a unified and functionally appropriate security approach. There are various uses or protocols related to satellites and other space systems that drive work on DTN and QUIC. We observe for the first time how integration of RFC9420 [3], a standardized CKA by the Internet Engineering Task Force (IETF), can solve a key establishment challenges and open issues in approaches in both of these cases. We also discuss how protocol changes to QUIC can be performed to incorporate such a CKA.

5.2 Space Networking Stacks

The IETF and the Consultative Committee on Space Data Systems (CCSDS) have investigated and made efforts towards the use of DTN protocols in space. There is also growing enthusiasm to use the IP stack architectures as an alternative to or in conjunction with the DTN stack [81], [93]. Supporting arguments of either approach depend on the particular space mission and their locations in space: *near-earth*, *lunar*, or *deep-space* [94], [95]. We

briefly summarize this context to lay the ground work for discussion of security approaches. However, recommendations for selection of or use-case preferences among the different fundamental networking stack approaches is out of scope of this work.

Near-earth. Communications in Low Earth Orbit (LEO) and Geostationary Orbit (GEO) have the advantage of relatively low round-trip times, on the order of 20 milliseconds (ms) to 250 ms respectively. The relative lower delay makes it potentially feasible to use traditional IP protocols which have the benefit of widespread implementation and performance characteristics in low latency settings. However, even at these distances link availability is still susceptible to the effects of atmospheric conditions (e.g. rain, clouds, scintillation) and orbit dwell times (e.g. context switching in LEOs and downtime from perigee of Highly Elliptical Orbits). To ameliorate these constraints, discussions on IP stack use in near-earth missions generally include careful pre-planning (i.e. network topology designs) and additional infrastructure to close the gaps in connectivity (e.g. more ground station transceivers or proliferated mega-constellation networks) [96], [97]. Prior research suggests that DTN protocols begin to eclipse IP in performance (i.e., better goodput [98]) when subjected to round-trip times longer than 200ms [99].

Deep Space. In deep space (i.e. beyond lunar), the near-earth concerns are amplified to the point where TCP/IP protocols alone have received skepticism on insufficiency [95]. The transmission round-trip time from Earth to Mars, for example, is on the order of just under one minute to 23 minutes depending on the orbital positions of the two systems [100]. Further, systems that operate in deep space must be even more judicious with power and resource conservation: their replenishment is cost prohibitive after launch.

Lunar. Lunar communications can be viewed as a middle ground for integrating near earth and deep-space solutions. Lunar communications typically have round-trip times of 5 to 14 seconds [101]. In addition to distance, network designs must also account for lunar orbiting nodes being periodically unreachable due to being blocked by their orbit around the moon. Lunanet, a NASA and ESA project aimed at providing cis-lunar communications, approaches lunar communications issues with a hybrid architecture involving use of IP, 3PP, and DTN protocols [102].

In both DTN and IP stack networking approaches, appropriate key establishment methods need to be selected. The following two subsections summarize the current security

approaches for each. We then discuss the security considerations of these and current gaps.

5.2.1 DTN Stacks with BPSec Security

The Bundle Protocol version 7 (BPv7 or simply BP) [4] is a store-and-forward application layer protocol with integrated transport[13] currently deployed as a DTN component by NASA in the DTN stack [94]. A bundle is comprised of a *primary block* containing necessary header information, a *payload block*, and *extension blocks*. Bundles are passed hop-by-hop among source, intermediate, and destination nodes, operating on a best-effort store-and-forward basis. This may include *convergence-layer protocols* (i.e. DTN adapted transport protocols) for end-to-end bundle delivery assurance. As such, BP is compatible with both DTN-specific transport protocols and and TCP/IP based protocols (through their respective convergence-layer adapters such as QUICL). Security of the BP blocks is provided through the *BPSec* protocol [9].

BPSec is a secure channel protocol that provides confidentiality and integrity. Unlike a traditional secure channel, BPSec relies on shared keys across source, intermediate, and destination nodes. This is to enable flexibility; intermediate nodes are expected to process and potentially decrypt and act on bundles as part of the store-and-forward paradigm. Such functionality can enable intermediate node to act as network gateways or proxies, and even discard blocks. Supporting shared keys across all nodes raises questions about key establishment and maintenance, however. BPSec, unlike traditional secure channels (i.e. IPSec), does not perform key establishment itself but instead relies on so-called out-of-band key establishment, i.e., pre-shared symmetric keys (PSKs) under the default *security context*, i.e., default mode [11]. PSKs are used for integrity protection in Message Authentication Codes (MAC) and confidentiality protection in authenticated encryption with associated data (AEAD). Alternative ciphersuites and parameter sets may also be used for BPSec depending on the *security context* supported by participating nodes. This allows for the potential introduction of asymmetric cryptographic methods and assignment as new security contexts [103].

[13] Torgerson et. al. in [27] refer to BP as existing in the "bundle layer", but we use classification by [4] for clarity and familiarity.

5.2.2 TCP/IP Stacks with QUIC Security

In the past, transport protocols such as TCP were defined separately from key establishment protocols, e.g., Transport Layer Security (TLS) [104], leading to inefficiencies in connection setup and specifically, round-trips. QUIC was developed as an "all-in-one" alternative approach, combining a UDP-based transport protocol with key establishment and secure channel protocols. By combining the two into a single protocol, various optimizations were possible. Key establishment in QUIC is based on the TLS Handshake protocol – which it separates from the TLS Record Layer and Alert protocols (used in TLS for the secure channel phase) [105]. It instead incorporates the TLS Handshake with a customized stack to reduce the inefficiencies in connection establishment. Since the TLS Handshake provides entity authentication (either mutual or unilateral authentication) by leveraging public key infrastructure certificates, so too does QUIC. The symmetric keys derived from the handshake are used with AEAD for the QUIC secure channel. The QUIC secure channel does not use the TLS Record Layer protocol, however, instead opting for a separate customization in framing format. Since reference terminology will be useful in what follows and QUIC does not define terminology for its components, we will abuse TLS terminology and refer to the components of the TLS-based QUIC Handshake as QUIC-HS and the subsequent QUIC secure channel as QUIC-RL (i.e., "QUIC Record Layer").

QUIC's use of UDP instead of TCP has made it more appealing than other transport layer options for space applications, as latency is a concern. Under reliable network settings, QUIC performs better than other interactive IP protocols due to fewer round-trips [106] since it removes the need to for a separate TCP SYN/ACK protocol initiation before the security protocol starts. While round trip reductions have improved the latency costs, QUIC is still a session-based protocol and the costs are not optimally minimized, as will be discussed in Section 5.2.3. In space settings, any potential for latency reduction is critically important for consideration.

5.2.3 Gaps and Issues

BPSec. Generally, reliance on PSKs in BPSec limits both security and functionality. First, the a static PSK incurs an increased vulnerability window, i.e., the compromise of the PSK reveals all communications in the past, present, and future that have been protected by that key. In cryptographic terms, it lacks *forward secrecy* [107] (protection of past data in the

event of compromise) and *post-compromise security* [108] ('self-healing' of the connection and protection of future data in the event of compromise, under certain conditions). Second, it has limited flexibility as it is difficult to add or remove access (the PSK must be distributed to all nodes through some pre-coordination). In cases of manual PSK installation this is particularly problematic. The lack of interoperability impacts federation potential, while no formal protocol definition for distribution of PSKs increases management complexity for devices and the system over time.

QUIC. As with TLS, QUIC was designed for the interactive client-server model. It assumes synchronous, session-based connections between two parties where data is in the clear on both ends. QUIC-HS is by default stateless, designed for internet-style use cases where tens of thousands of clients could easily be connecting to a single server per second [109]. Thus, QUIC executes full handshakes to establish each new connection. This requirement of interactive connection establishment presents an issue in environments where low-latency is a concern. While this model is ideal for one-to-one or many-to-one internet applications such as web-browsing or content streaming,it is incongruent with the space setting.

Specifically, high latency and intermittent link availability directly makes session-based protocols sub-optimal; if a link has to be reestablished, an entire new handshake is required. Even with the round trip improvements in QUIC over TLS, this still costs a full round trip. With intermittent link availability, the cryptographic overhead of a session-based key establishment protocol used to create a secure channel becomes a self-imposed denial of service. Notably, a stateless protocol such as this is not even inherently required for most space settings; unlike with web traffic where it can be problematic to hold state for tens of thousands clients that may or may not reconnect to a server, the space nodes that connect to each other are relatively few in number.[14]

QUIC does support an option for stateful connections using *session resumption* and *0-RTT*, but sacrifices security for efficiency in such instances by reusing the previous connection state. The QUIC designers have advised disabling these features due to privacy concerns and replay vulnerabilities [110], making them ill-advised as options for solving the latency and handshake overhead challenges in space. Despite these warnings by the QUIC designers,

[14] Relatively few space devices and the potential for stateful connections is the very reason that PSKs could even be considered in the BPSec context.

early testing on use of QUIC in space systems have specifically looked at the 0-RTT option given normal latency concerns under secure QUIC session establishment [111]. QUIC also does not offer post-compromise security once keys are lost, including in its session-resumption option.

Furthermore, the one-to-one QUIC connections scale poorly ($O(()n^2)$) for groups of devices – a space networking scenario that is anticipated by the space development community as evidenced by the design decisions of BPSec.

5.3 The Space Security Environment

The above issues highlight necessary directions for improving for key establishment designs relative to space settings. Such a protocol should 1) reduce round-trips to the maximum extent possible, to better support cases where latency is an issue. Such a protocol is also, notably, 2) not required to be stateless, as unlike Internet use cases there is not a high risk of exhaustion from tens of thousands of different connection establishments. It is therefore possible to utilize stateful protocols that may offer better security or functionality benefits.[15] Such a protocol should 3) have the capacity to refresh keys at regular or predetermined intervals for forward secrecy and post-compromise security. It should 4) ideally be scalable to groups of devices (for BP), where there is the potential to insert further spacecraft into the communication, and 5) should support asynchronicity (for BP). What is most notable here is that the design space expands when stateful protocols are considered. Because connections in space are orders of magnitude longer-lived in terms of paired communicating identities (potentially decades) [112] and orders of magnitude fewer than internet connections, ad-hoc state updates are advantageous over handshakes: it takes fewer messages to maintain and/or update state than it does to complete handshakes to obtain fresh keys.

5.3.1 Solutions

Among stateful protocols with reduced latency, the category of continuous key agreement (CKA) protocols, a.k.a. ratcheted protocols, is salient. Here participants establish and maintain a shared cryptographic state which can be updated over time with fresh keying material.

[15] Note that a stateful protocols implies a typical ability to maintain state and not an inability to recover from state loss.

Instead of repetitious session establishment as in QUIC, CKAs establish a session once, and update keys as needed – this enables no additional round-trip overhead or indeed even bandwidth costs when new transmission connections are started. The stateful protocol can instead send encrypted data immediately, without transmitting certificates or other keying material, with instead key updates taking place when appropriate as determined by the application. This is both an efficiency and security upgrade from stateless protocols like QUIC-HS/TLS. Furthermore, it does not inhibit support for cases of state loss, as a new session can be established, but rather lowers latency in all other cases.

Additionally, many CKAs are asynchronous protocols, making them well-suited for the relay context such as BP requires, where communication partners may be dynamically unavailable or transmissions pass through intermediate nodes. CKAs are also usually designed for forward secrecy and post-compromise security protections, with currently deployed examples including the Signal protocol [113], WhatsApp Sender Keys protocol [114], and the Messaging Layer Security (MLS) protocol [115]. While CKA deployments have historically been in messaging applications, there have been increasingly other applications in disrupted, relay, or high-latency settings [116], [117]– making them unsurprisingly suited for space.

Among CKAs, MLS features as an IETF standardized, open protocol [3]. IETF standardization has been a feature for other space system deployments (both BPSec and QUIC being standardized in this way), and could further enable adoption by organizations like the CCSDS and its members (e.g. NASA and ESA). MLS can also be deployed with post-quantum security [118], [119], including efficient and flexible post-quantum options for amortizing the computational cost of post-quantum algorithms [119].

5.3.2 Functionality and Security Comparisons

At a high level, MLS is not only a CKA, but a multi-party CKA, which supports goal (4) for scalability. It was designed to support scalable connections for multiple parties without the need for pairwise handshakes, or indeed all parties being connected simultaneously. It scales logarithmically with the size of a group ($O(()n\log(n))$ vs $O(()n^2)$ for pairwise QUIC handshakes), and supports operations to *update* keying material, *add* new group members, and *remove* existing members. MLS supports client-server models and pairwise

connections where desired (e.g., analogous to QUIC), as well as decentralized models (similar to BPSec).

5.4 CKA Integration with QUIC and BPSec

Since a CKA is a key establishment protocol it offers the possibility for smooth integration in the existing space network architectural landscape. Here we describe at a high level how MLS, as an example CKA, can be incorporated into the both the DTN stack and the IP stack approaches (Figure 5.1).

5.4.1 MLS as Key Establishment for BPSec

As noted previously, BPSec assumes out-of-band key agreement, typically assuming a PSK. MLS outputs keys that is indistinguishable from random (a standard cryptographic assumption for key establishment) [115]. Such keys can be used directly for the BPSec channel wherever a shared key is required.

In further practical implementation considerations, MLS implementations [120] use *pre-key bundles* which contain public keys. Distribution of these public keys can be fulfilled through DTN-tailored Public Key Infrastructure protocols such as the Delay Tolerant Key Administration (DTKA) protocol or KeySpace [121], [122].[16] In a pure DTN stack, the Licklider [123] protocol, which provides reliable data transport using scheduled transmissions and quality of service mechanics could provide the underlying transport delivery service assurance.

5.4.2 MLS as Key Establishment for QUIC

We present the novel recommendation for incorporating the MLS key establishment and large QUIC protocol. Doing so within QUIC is notionally even more straightforward than in BPSec. Namely, the current QUIC-HS is a TLS-based handshake, which itself has been cryptographically analyzed for producing indistinguishable-from-random keys [124], which are then used in the secure channel phase (QUIC-RL). Since MLS also produces indistinguishable-from-random keys [125], not only is the functional replacement imply

[16] PSKs *can* be used to bootstrap MLS key agreement instead of public keys, but it is non-standard practice and comes with security considerations [3]. Normal MLS initial key establishment is advisable.

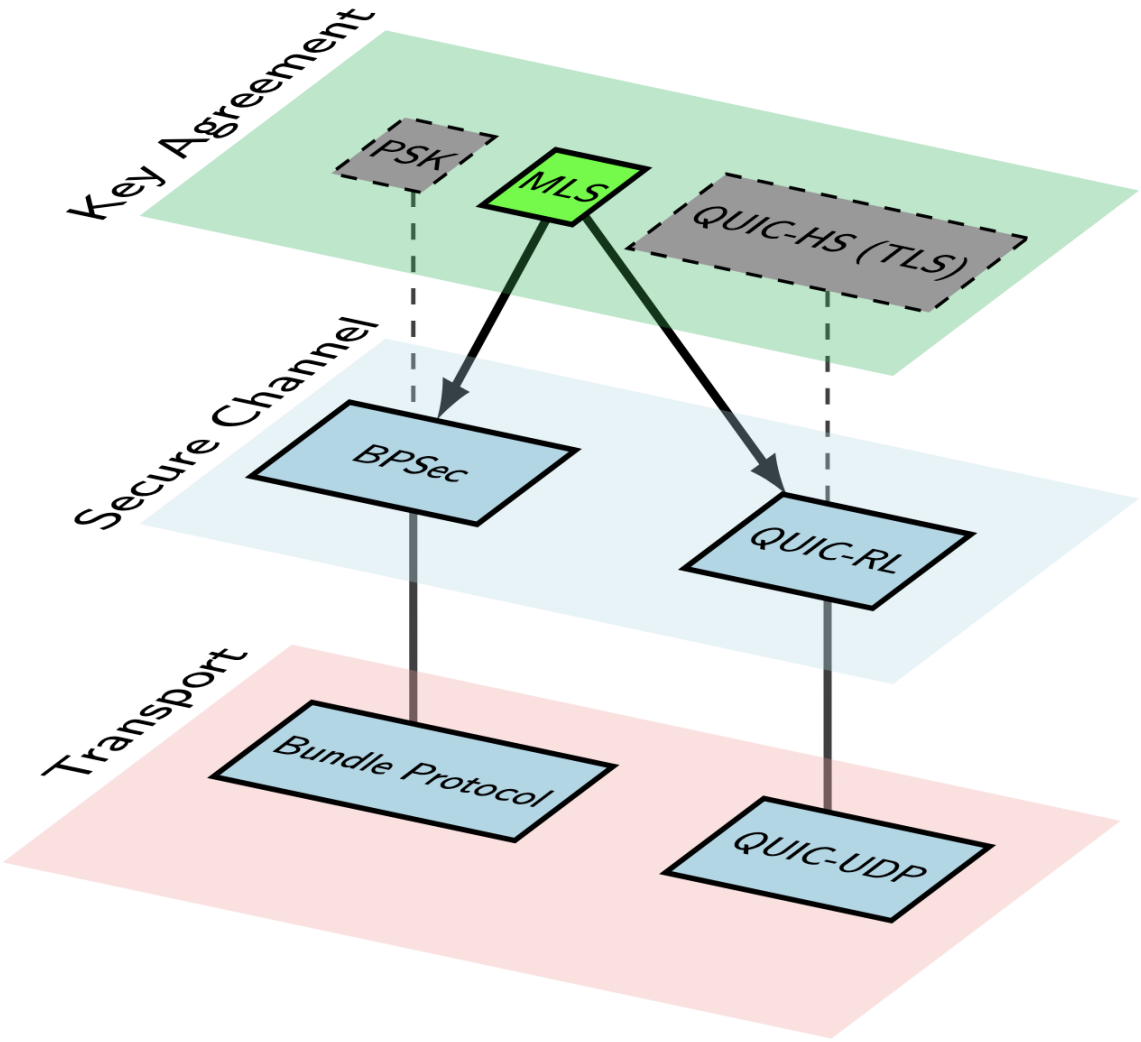


Figure 5.1. MLS as a Key Establishment for BPSec or QUIC. As shown, MLS provides the key (replacing the PSK) in BPSec. It can also replace the TLS handshake in QUIC, with resultant keys usable directly in the QUIC "Record Layer". Solid edges represent the envisioned stack while dashed edges represent current pathways. Convergence Layer adapters can be used to integrate QUIC with BP, if desired, and are not shown.

that these keys can be used directly in the QUIC secure channel phase (QUIC-RL), but that security analyses are composable.

Furthermore, MLS provides an authenticated key exchange which can be evolved, either

ad-hoc or on a pre-determined schedule. This removes the security need for a round trip when previously connected devices reconnect, and removes the session resumption risk of QUIC (see Section 5.2.3). MLS also supports authenticated negotiation of application protocols using the content advertisement extension [126]. In turn, in contexts where QUIC is already considered for space applications, MLS does not introduce any changes certificate management but could, optionally, use lighter weight certificate management approaches due to the flexibility in "identity" definition [3]. Since MLS is a change and replacement for the key establishment phase only, any desired connection reliability benefits from QUIC's transport are preserved.

5.5 Conclusion

The disconnect between the space engineering and cryptography fields risks contributing to a design of space networking that is not suited for its purpose, as nuances in either field are not always well communicated to the other. Functionality requirements in space are different than in Internet connections and change the types of security protocols that should be considered. Failure to choose security protocols that meet the use case can risk long-term sub-optimal functionality or security vulnerabilities. This work outlines existing conflicts among methods for key establishment in space communications and we propose a path towards both functional and security improvements that meets various networking stack approaches. Ultimately, progression in space has always required close collaboration across many disciplines, and that continues to apply with the expansion of security in space.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
BPSec-MLS: Asynchronous Key Agreement for Space Communications

Overcoming DTN rigidity through dynamic key agreement

As established in Chapter 1, the rigidity of traditional space security architectures creates operational vulnerabilities that are particularly acute for Delay Tolerant Networks (DTN). DTN security has historically relied entirely on out-of-band key provisioning: mission planners must preload all necessary cryptographic keys before launch or coordinate key distribution through separate, non-DTN channels. This approach prevents missions from adapting to unforeseen scenarios such as new partnership opportunities (e.g., ad hoc collaboration between NASA, ESA, and commercial operators), constellation expansions (e.g., adding relay satellites to extend coverage), or security incidents (e.g., recovering from a compromised node). The months-long manual coordination required to pre-negotiate secure communication paths leaves DTN systems brittle to any changes in mission requirements. Without in-band, standards-based key agreement mechanisms, DTN security remains locked in the rigid, preshared-key paradigm that Chapter 1 identifies as a fundamental inadequacy.

Pillar II: Dynamic Key Agreement for DTN Environments. Recall from Chapter 1 that Pillar II addresses the rigidity problem by integrating modern dynamic and asynchronous key agreement mechanisms into the two dominant network architectures for space communications: Internet Protocol (IP) networks and Delay Tolerant Networks (DTN). While DTN architectures are explicitly designed to tolerate delay and disruption through store-and-forward custody-based routing, they have historically lacked integrated key agreement mechanisms, forcing reliance on preshared keys that inhibit operational flexibility and security resilience. BPSec-MLS (this chapter) introduces the first standardized, in-band key agreement mechanism integrated directly into BPSec, enabling dynamic, asynchronous establishment and rotation of cryptographic keys entirely within the DTN infrastructure. Together with QUIC-MLS (Chapter 7), which addresses the IP component of Pillar II, these integrations provide comprehensive key agreement coverage across heterogeneous space network architectures. The security properties established in Pillar I transfer to

these integrations: the formal analysis of BPSec in Chapter 3 provides confidence that BPSec-MLS preserves secure channel guarantees, and the guardianship mechanisms from Chapter 4 ensure that even receive-only or emission-controlled DTN nodes can maintain post-compromise security.

Chapter Contribution and Impact. This chapter presents the first demonstration of dynamic key agreement for Delay Tolerant Networks that is compliant with existing IETF standards: RFC9172 [9] (BPSec) and RFC9420 [3] (MLS). By implementing MLS as the key agreement component for the BPSec secure channel protocol used by major space agencies worldwide—including NASA and the European Space Agency (ESA)—this work transforms space operations from rigid, pre-planned configurations to flexible, adaptive security postures.

At the highest level, this work enables operational capabilities previously impossible with traditional DTN security. Space systems can now establish secure communications with previously unknown peers after launch, enabling spontaneous collaborations between space agencies, commercial operators, and international partners without months of advance coordination. Missions can recover from key compromises through post-compromise secure updates initiated by other group members, containing the security breach without requiring physical access to spacecraft or mission replanning. Constellations can adapt group membership dynamically as satellites, ground stations, and relay nodes come online or go offline due to orbital mechanics, equipment failures, or mission phase transitions. For future architectures like NASA's Lunanet—which will integrate DTN and IP protocols across lunar infrastructure—BPSec-MLS provides the DTN security foundation that complements the QUIC-MLS IP foundation, enabling seamless secure communications across heterogeneous network boundaries.

At the protocol level, this work demonstrates that MLS's asynchronous key agreement mechanisms are well-suited to DTN's store-and-forward architecture. The implementation exploits DTN's custody-based forwarding to reliably deliver MLS control messages (Commits, Proposals, Welcome messages) across intermittent links, and adapts MLS group operations to tolerate the long propagation delays and limited bandwidth characteristic of deep space communications. The work presents multiple configuration options for MLS group operations that yield different performance and security tradeoffs, enabling mission de-

signers to tune BPSec-MLS to their specific operational constraints: high-security missions can perform frequent key rotations to minimize compromise windows, while bandwidth-constrained missions can amortize key agreement overhead across longer epochs.

At the implementation level, results are measured using NASA's Interplanetary Overlay Network (ION) software suite, which implements BPSec and has been validated for compliance with both CCSDS and IETF specifications. The performance evaluation demonstrates that BPSec-MLS operates efficiently ranging from linear to logarithmic scaling depending on group configurations, with empirical results showing acceptable overhead for realistic space communication scenarios based on proposed Earth-Mars and lunar networks. Specific metrics measured include: key agreement message overhead, latency introduced by MLS group operations, computational overhead for key derivation and cryptographic operations, and bandwidth and storage requirements for MLS group state at intermediate custodial nodes. As an artifact of this work, the implementation design and integration procedures may be used for immediate deployment on real systems that are similarly compliant with existing BPSec and MLS specifications, providing a concrete pathway for space agencies and commercial operators to upgrade from preshared-key architectures to dynamic key agreement.

Real-World Impact and Future Standardization. This work was published at the IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE), peer-reviewed by experts in space communications and extreme environment networking. It was also presented as an invited talk to the Space-Terrestrial Internetworking workshop, receiving feedback from satellite designers and mission managers on operational feasibility. Most significantly, this work has been presented at the Consultative Committee for Space Data Systems (CCSDS) Fall 2025 meeting, where CCSDS—which promulgates communications standards for the majority of space agencies including NASA, ESA, JAXA, and others—is drafting standards based on this work. The intention is to submit a draft IETF RFC that stipulates the procedures for formal incorporation of MLS into BPSec, ensuring that BPSec-MLS becomes part of the standardized internet protocol suite available for broader adoption beyond space-specific applications. Furthermore, the European Space Agency has extended invitations to test BPSec-MLS on a system to be launched in 2026, providing real-world flight validation of the dynamic key agreement mechanisms developed in this work.

Joint Work and Contributions. This chapter presents the integration of MLS into BPSec

and was joint work with Paul Westland from the Naval Postgraduate School.[17] This paper is joint work with Paul Westland in support of his Master's thesis. The candidate supervised Paul Westland and based this work on the paper in Chapter 5. The methodology was planned jointly by the candidate and Paul Westland with guidance provided by Britta Hale, the analysis was done jointly by the candidate and Paul Westland, and the list of future works was developed jointly by the candidate and Paul Westland. The candidate drafted the paper to include all figures and graphs and edited the paper with input from Paul Westland.

6.1 Introduction

The growing domain of space scenarios [127]–[129] requiring interoperability, efficient scaling, and strong security guarantees is outpacing the limits imposed through use of pre-shared keys and synchronous handshake protocols [101], [122], [130]. Pre-shared symmetric keys limit networks from being able to dynamically add or remove users. Moreover, compromise of a pre-shared key reveals all past and future communications to an adversary. Alternatively, synchronous handshake protocols allow for dynamic session setups which protects previous communications from key compromise, however, their interactive nature is unsuited for delay and disruption prone networks [27].

In alignment to the requirements of delay tolerant networks (DTN), the Messaging Layer Security (MLS) protocol [3] was designed for scalable asynchronous group key agreement. MLS achieves forward secrecy (FS) and post-compromise security (PCS) to protect previous and future communications against key compromise through the evolution of a shared group state across epochs in a continuous session. FS guarantees that past conversations remain secure after a compromise of the current session key. PCS guarantees that future communications are secure after a compromise.

The ability to efficiently and continuously update keys and dynamically add or remove group members *post-launch* is a particular advantage of using MLS that is currently absent from the Bundle Protocol Security (BPSec) which used to provide application data confidentiality and

[17]Reprinted, with permission, from all authors. 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. IEEE will claim and protect its copyright in international jurisdictions where permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

integrity in DTNs. In terms of efficiency, MLS can update group keys with communication costs as low as logarithmic in the best case, though in practice the cost is usually closer to linear on average [131]–[133]. By contrast, synchronous protocols that rely on fresh Diffie-Hellman values (i.e. starting a new session to achieve comparable FS/PCS), incur polynomial overhead in relation to the group size. If keys are updated by bootstrapping off the existing session keys then FS and PCS are lost.

Dowling et al. in [134] first outlined the merits of MLS as a key agreement protocol for space communications architectures, highlighting many of the aforementioned issues. This work implements their proposal to incorporate MLS in BPSec and maps it to relevant space scenarios and measures costs of such an incorporation. Our non-intrusive approach ensures that the reference libraries of BPSec and MLS do not deviate from their standards, thus reducing systems engineering barriers to deploying BPSec-MLS.

We make the following contributions in this work:

- To the best of our knowledge, we provide the first implementation of a continuous group key agreement protocol for delay tolerant networks. Furthermore, our implementation is in compliance with both BPSec and MLS standards and is tailored for space scenarios through incorporation of MLS in BPSec.
- We detail practical considerations for incorporating MLS architectural components to the DTN environment.
- We show MLS overhead in a full DTN stack covering two group administration strategies and present their trade-offs in terms of efficiency and security guarantees.

6.2 Background

We provide preliminaries and relevant context to key agreement protocols, delay tolerant networks, and efforts to combine the two in the following section.

6.2.1 Continuous Group Key Agreement Protocols

The Signal protocol's double ratchet algorithm inspired formalization of the concept of a Continuous Key Agreement (CKA) protocol by Alwen et al. in [135] as a cryptographic primitive that enables two parties to agree on a shared secret key that evolves over time. As

an extension of CKA, Continuous Group Key Agreement (CGKA) is used in dynamic multi-party settings where members may join, leave, or update their keys frequently [125]. CGKA protocols provide strong security properties such as forward secrecy, post-compromise security, and efficient group operations. TreeKEM is a specific CGKA protocol that uses a tree-based structure to manage keys and efficiently handle group changes. It serves as the cryptographic backbone of MLS and has been formally analyzed and improved upon in lockstep [136]–[138].

Messaging Layer Security Protocol

The MLS protocol was standardized as RFC9420 in 2023 by the Internet Engineering Task Force (IETF) to provide end-to-end security [3]. As an open and interoperable standard, it has been deployed commercially by Cisco Webex, RingCentral, and Wire [139]. MLS uses the TreeKEM protocol to maintain a shared cryptographic state upon which keys for various purposes (e.g., key encapsulation, encryption, signing, etc.) may be derived. The MLS architecture document describes various components necessary to run MLS [140]. To add a new group member, the sender first acquires a KeyPackage associated to the invitee signed by an authentication service. Then, using the KeyPackage, the sender adds the member to their local group state generates a Welcome message for the invitee to allow them to calculate the group state and join the group. Because this changes the group state, the sender must also send an Add proposal which must be committed (either through another member or via self-commit) to synchronize the new group state. To achieve PCS, members should also issue an updates that are committed periodically.

In general, MLS group members advance the shared cryptographic state (via adding, removing, updating) through a propose-and-commit scheme. This characteristic, which maximizes autonomy of members, can lead to synchronization issues if the underlying delivery service is unreliable [140]. For this reason, our DTN implementation of MLS which may be prone to disruptions and delays delegates the ability to commit update proposals to a group leader. Naively applied, this paradigm can create a ratchet tree where a majority of the intermediate nodes are blanked which nullifies any logarithmic scaling provided by a well populated ratchet tree. However, this may be avoided if each even indexed member is allowed to issue an update and commit the first time they join as part of the group instantiation process. We elaborate further on this in Section 6.3.3.

Performance Analysis

Theoretical asymptotic analysis of CGKAs has been previously analyzed by Beinstock et al. in [141] who show CGKAs generally achieve $\Omega(n)$ worst-case communication complexity. Anastos, et al. in [142] refine this by showing the cost (number of uploaded ciphertexts) of replacing a set d users in a group of size n is $\Omega(d \ln(n/d))$. Theoretical and experimental performance analysis of Server Aided Insider-Secure TreeKEM, was performed by Alwen, et al. in [132]. They show, at the scale of tens of thousands of members in the client-server setting, communications bandwidth in best-cases can scale logarithmically for individual senders and receivers; however, in the worst-case scenarios bandwidth scaling was still linear for senders (servers) the total combined bandwidth of the group was $\Omega(n \log(n))$. Chavelier, et. al in [143] provide further insight on the variability of communications performance depending on the configuration of the MLS ratchet tree. They highlight inefficiencies associated with the MLS left-balanced ratchet tree expansion algorithm and note that the lower bound logarithmic scaling would only be attainable under the ideal ratchet tree structure (i.e., a full balanced binary tree) which can only occur if the group size is a power of two. Soler, et al. in [133] provide another experimental analysis of MLS using different delivery services. Soler, et al. show under practical configurations and various group sizes ranging to one thousand, that the computational and storage costs scale generally linear rather than logarithmic as has been previously claimed.

6.2.2 Delay Tolerant Networks (DTN)

DTNs describe a collection of protocols developed by NASA originally for deep space communication that follow a store-and-forward paradigm. DTN protocols exist for each level of the network stack. As a DTN counterpart to IP, the Bundle Protocol (BP) [4] when used with BPSec [9], is a secure DTN application messaging protocol with integrated transport functionalities. BPSec uses Block Integrity Blocks (BIB) and Block Confidentiality Blocks (BCB) to provide data authentication and confidentiality to blocks within a bundle according to local security policies. BIB and BCB are cryptographically parameterized by a Security Context, configured and loaded out-of-band. Formal analysis of BPSec under the Default Security Context has been analyzed in [144].

Unlike other secure transport protocols such as QUIC, BPSec explicitly does not provide the function of authenticated key exchange [11]. Instead, users must perform entity authen-

tication and key agreement out-of-band (e.g. via pre-installed keys). In this regard, BPSec falls short of providing users with modern end-to-end security guarantees. One attempt to bridge this gap has been explored in [145] on a QUIC convergence layer protocol (QUICL) for BP which uses Transport Layer Security (TLS) for key agreement. However, as [134] argues, several issues remain with using standard QUIC with TLS in high-latency and intermittently connected networks. In a recent draft [146] submitted to the IETF, the authors propose to use QUIC with MLS instead of TLS for use in IP based space communications settings. However, until a convergence layer adapter is created for QUIC-MLS, it won't be compatible with BP networks.

Non-standardized Approaches to Key Agreement

Various efforts have tried to incorporate key agreement in delay tolerant networks and space scenarios. Recently, the TripleKEM algorithm was proposed for the Space Data Link Security (SDLS) protocol [147]. However, the interactive nature of TripleKEM limits its scalability to go beyond hop-to-hop pairwise scenarios endemic to the link layer. Another drawback of TripleKEM and other key agreement protocols proposed for DTNs [148]–[150] is that none have been standardized, made open-source, nor are widely adopted. On the other hand, MLS is an open-source, standardized protocol, commercially deployed, and has been thoroughly tested and vetted among academic and commercial researchers for its security, performance, and interoperability [115], [137], [138], [151].

6.3 Design

In this section we explore the design space associated with implementing MLS as a key agreement mechanism in BPSec. End-to-end security (E2E) and efficiency guarantees aforementioned differ in degree depending on how MLS is integrated and on assumptions of MLS architectural components.

6.3.1 Integration Rationale

MLS provides E2E authenticated key agreement between group members. In BP networks, trusted (aka *privileged*) intermediate nodes between a source and destination with a shared key-set must sometimes add or process BPSec blocks (e.g. performance enhancement proxies). This inherent flexibility of BPSec which facilitates the store-and-forward paradigm

is at odds with the E2E paradigm in the strictest sense of the meaning [152]: that only endpoints should handle the security functionality.

If a strict E2E secure channel is required, then a simple approach might be to use MLS *on top* of BP. This would include sending MLS protected application messages in addition to MLS handshake messages in the payload (see right side of Figure 6.1). This design relegates intermediate nodes to only function as forwarders, with no insight on the encrypted payload, which may be incompatible with networks that use proxies or gateways for security services. This approach would also subvert the flexibility of BPSec in providing block-level granularity and interactions among security operations as analyzed in [144]. In fact, MLS would obviate BPSec entirely as this tunneling technique includes both authenticated key agreement and confidential authenticated messaging.

In order to preserve the flexibility of BPSec while providing the option to enforce E2E design (i.e., by excluding intermediate nodes), our implementation integrates MLS into BPSec while maintaining compliance to the current standards [9], [11]. Our design, shown in Figure 6.1, sends MLS handshake messages through the payload block. Alternatively, a more efficient design would have MLS messages be carried in a designated extension block type or administrative record [153] but we chose to not do so to maintain compliance with current standards. Once received, bundles containing MLS messages are processed by recipients to establish or update a shared cryptographic state from which keys are derived in accordance with the MLS Key Schedule [3]. Cryptographic parameters and capabilities are also conveyed through the group state which can be checked against security contexts supported by group members. Rather than implementing this feature, we hard-coded the parameters to match.

6.3.2 Architectural Components and Assumptions

Delivery Service

MLS group operations are enacted through a propose-and-commit sequence which necessitates a Delivery Service (DS) that ensures the ordered delivery of these messages and resolves any conflicts. When a MLS group member sends a group operation proposal (e.g. add, remove, update), any member of the MLS group may send a commit to the list of proposals and move the group state forward. The DS is an MLS architectural assumption

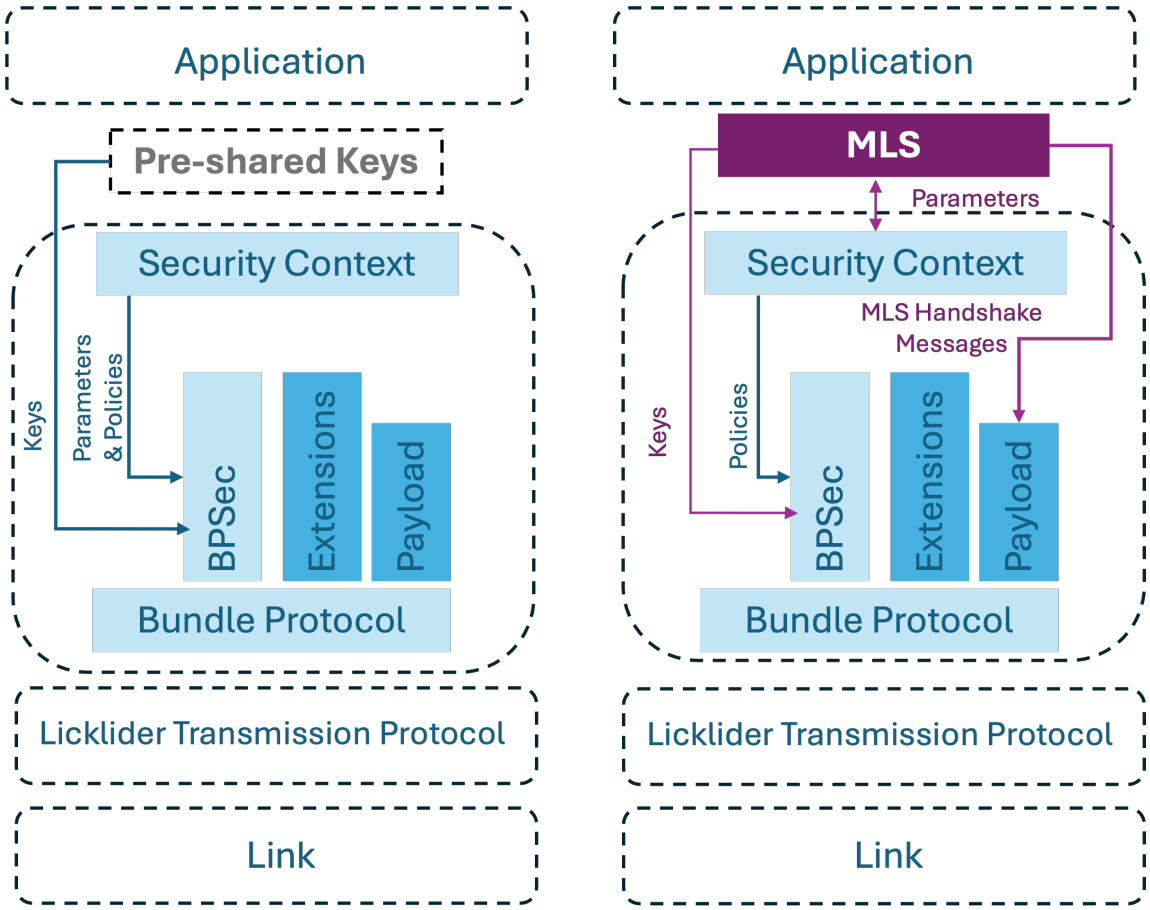


Figure 6.1. BPSec using out-of-band (pre-shared) keys (left) versus our BPSec in-band key agreement using MLS (right)

that decides the ordering and precedence of proposals and commits which helps to keep the common group state synchronized. In existing implementations on the internet, the DS role is relegated to either the underlying transport protocol such as TCP or a centralized server.

After consideration of alternatives (see Section 6.4) , our DS is fulfilled by using the Licklider Transport Protocol (LTP) for reliable delivery of messages and having a single group leader to reconcile proposals and sending commits [123].

Authentication Service

The role of the Authentication Service (AS) for MLS is to associate an identity to a signature key pair [140]. In the DTN setting, several systems have been proposed that can fulfill the role

of the AS. For example, as a public key distribution service, DTKA can fulfill the role of the AS. More recently, authors in [122] demonstrated the viability of standard terrestrial style public key infrastructure through customization of the Online Status Certificate Protocol and Certificate Revocation List delivery methods based on the same set of stipulations set by DTKA in [121]. Since a litany of AS candidates exist, we leave AS design analysis and comparison for future work and simply preset identity bindings to signature keys for our implementation.

6.3.3 Network Scenario Alignment

Our implementation uses a hub-spoke topology as shown in Figure 6.2 for key agreement that is aligned with standard space architectures and scenarios described by the Consultative Committee for Space Data Systems in [95]. Specifically, we model communications between terrestrial control center(s) to lunar terminals and relay orbiter(s) to lunar terminals.

The well-connected hub is the MLS *leader* group member that adjudicates commitments and proposals sent by variably connected *subordinates* group members. Subordinates, which may or may not be well connected among each other, can then use the group key to communicate directly within any topology using simplex or duplex channels.

The choice to consolidate commit power to the leader reduces the complexity in managing group state through unilateral arbitration of proposals for group consensus. It also reduces the computation and bandwidth burden on subordinates. However, this leader-subordinate paradigm can come at a cost to MLS performance and security guarantees depending on how the group is administered. We present three configuration options for consideration:

1. *Leader Only:* If subordinates never update themselves and only apply commits from the leader, then only the leader can achieve PCS and subordinates gain FS only. Furthermore, the MLS ratchet tree ostensibly flattens to a linked list which negates the logarithmic scaling benefits of a well structured tree [143].
2. *Subordinate Proposal:* If subordinates are allowed to propose updates but not ever commit, then they would regain PCS but still retain the poorly formed ratchet tree with blank intermediate nodes along the direct paths of subordinate nodes. Subordinates have more agency in updating but commit sequencing and conflict resolution still rest with the leader.

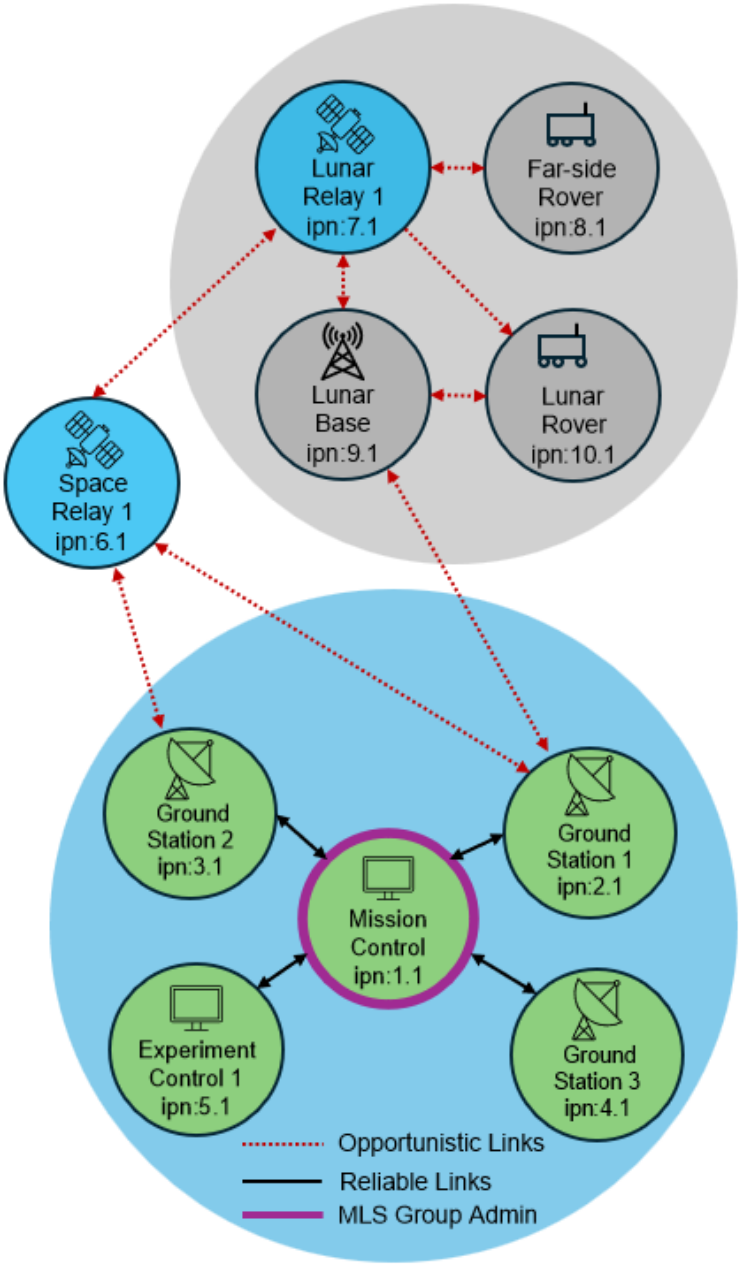


Figure 6.2. Hub-spoke MLS key agreement overlayed on DTN scenario adapted from [95]. Mission Control (purple) acts as the leader responsible for fanning out MLS commits to the rest of the members with varying connections.

3. *Subordinate Commit:* If subordinates are allowed to propose updates *and* commit their own update (e.g., just after joining the group), then both FS and PCS can be attained AND the ratchet tree could be filled and balanced so logarithmic efficiencies could be realized. The leader acts as a commit forwarder here and additional out-of-band coordination would be required to ensure no conflicting commits arrive.

Of the three leader-subordinate configurations, Option 1 is the most accommodating to DTN scenarios where subordinates can be severely resource constrained (compute, bandwidth, power, etc.) and delegation of MLS operations to the leader should be maximized. Options 2 and 3 achieve better security outcomes but requires more subordinate resources and potentially more coordination (i.e., timers, event logging, conflict resolution, etc.) to ensure group state synchronization. For testing and evaluation purposes, since Option 2 maintains a poor ratchet like in Option 1, our implementation examines only Options 1 and 3 (i.e., worst-case and best-case settings). Summarily, we hypothesize Option 3 to have lower overhead than Option 1. In practice, we recommend a combination of Option 2 and 3 to achieve the best security and efficiency outcomes. That is, subordinates should commit their own update after being added as part of group instantiation and subsequently only send proposals to the leader for commitment to ameliorate the concerns of out-of-order or conflicting commits due to delay or packet loss.

6.3.4 Parameter Selection

Relevant to using MLS for space based CGKA are computation cost, memory use, and size of the data transmitted. These metrics directly correspond to the practical constraints faced by onboard spacecraft computers and resource-constrained embedded systems. Compute time was measured to assess the processing burden imposed by cryptographic operations, particularly for handling welcome and update messages, since onboard CPUs often operate at lower clock speeds and under strict power budgets. Monitoring CPU usage allows us to attribute operations that could delay critical tasks or exceed real-time processing windows. Memory usage was tracked to ensure that the MLS session state and buffer handling remained within the limited memory typically available on flight-qualified hardware, where excessive heap allocation could cause failures to mission critical programs. Packet size was also recorded, as spectrum availability and bandwidth are both severely limited in deep space environments. Larger bundles result in longer transmission times and higher

energy costs per bit transmitted. By collecting these metrics, we provide tradeoffs between cryptographic security guarantees and the limitations of a device's physical hardware and communication channels in which these systems must operate.

6.4 Delivery Service (DS) Considerations

The MLS architecture in [140] discusses two types of delivery services for MLS: *strongly consistent* and *eventually consistent*. A strongly consistent DS is found in MLS implementations in traditional internet settings that utilize reliable delivery and ordering guarantees provided by TCP or a centralized server. An eventually consistent DS, which is more appropriate for a distributed peer-to-peer setting utilizes clients for reconciliation of conflicts. [18].

Several approaches for a DS are possible for the DTN setting. The most straight-forward is the strongly consistent DS which would use convergence layer adapters like TCP Convergence Layer (TCPCL) or Licklider Transmission Protocol (LTP) for reliable in-order delivery of MLS messages. An eventually consistent DS is more topologically flexible but is more complex, requiring considerably more storage, local processing, and bandwidth to store previous group states and either attach or solicit missing/unconfirmed commit messages. Soler et al. in [133] explore using the MQTT based GossipSub protocol and distributed hash tables to implement such an eventually consistent delivery service and found efficiency losses when compared to a hub-spoke style strongly consistent DS. As a proof of concept, our DS simply relies on LTP and we appoint a fixed group leader through which group control messages are passed to and committed. //TODO: Find a better place for this

6.5 Implementation Setup

Our setup ran inside a virtual machine (VM) on a personal computer with 64GB of RAM and an AMD Ryzen 5 3600 six-core processor. The Linux-based VM was configured with two cores and 54 GB of RAM. We used the Interplanetary Overlay Network (ION) software suite [154], which implements the Bundle Protocol [4], BPSec [155], and the Default Security Context [11]. For MLS, our MLS Daemon utilizes Cisco's MLSpp library [156].

[18] More discussion on how the CAP (Consistency, Availability, Partition Tolerance) Theorem is applied in designing a strongly consistent vs. eventually consistent DS can be found in [140]

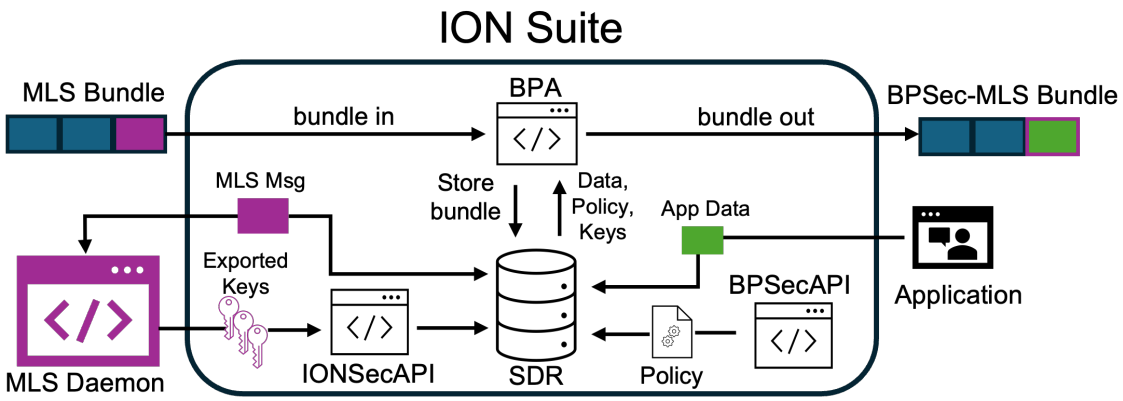


Figure 6.3. Simplified overview of components of an ION node running BPSec with MLS. Implementation details available at https://github.com/xisentian/ion-dtn-mls.

Each ION node is configured in accordance with Figure 6.3 and has its own MLS Daemon. The Bundle Protocol Agent (BPA) handles bundle transmission and reception. The Simple Data Recorder (SDR) stores data, policies, and keys used in BP operations. The ION Security API is used to read and write MLS keys to the SDR. The BPSec API is used to load a security policy into the SDR, which is referenced by the BPA when bundles are received or transmitted. If the policy specifies that security operations such as encryption or decryption are required, the policy also specifies the key to be used for those operations. Our MLS Daemon handles MLS messages parsed by the BPA for group creation, adding and removing members, as well as updating the group state. Depending on the role of the ION node (leader or subordinate), which is assigned by policy (out-of-band to MLS), the MLS Daemon allows the particular node to issue commits to proposals sent by other group members.

MLS group instantiation occurs with the leader adding all subordinates en masse which results in the creation of a single commit message (referencing a list of add proposals) and welcome message, both created from subordinate key packages generated using their unique endpoint IDs. After a new member processes the Welcome message contained in a bundle payload, the MLS daemon constructs the group state and is able to immediately process (e.g., encrypt, decrypt, sign, verify) BCB or BIB protected data. This concludes group instantiation for the configuration described in Section 6.3.3 for Option 1. For Option

3, an additional step is required to populate the contents of parent nodes in the MLS ratchet tree: every even-indexed member sends an empty commit to the leader to be broadcast after being added to the group. Once the MLS group has been established, keys can be updated via commits as frequently as needed. Ideally, all members should update on a regular basis to attain PCS.

6.6 Results

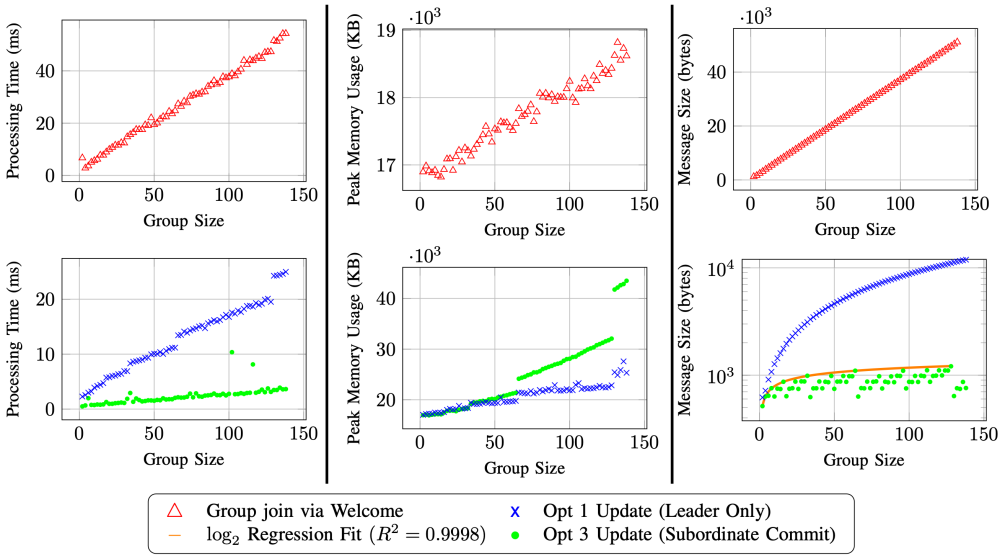


Figure 6.4. Group size effect on memory, processing time, and message size for group updates and joins (via Welcome messages). Note the use of logarithmic axis (bottom right) to compare the order of magnitude difference between message sizes for Option 1 vs. 3 updates. Logarithmic regression fit based on data from group sizes of powers of two.

Following the setup and group evolution sequences (Options 1 and 3) described in Section 6.5, we measured the computational and bandwidth costs of joining a group and processing an update shown in Figure 6.4. Since both options share the step of processing a Welcome message when subordinates join a group, only one set of data is shown. Measurements were collected at the leader for Option 1 and at the last member to join the group for Option 3.

Joining a group by processing the Welcome message exhibits a linear scaling pattern

with group size across all measured metrics. This follows logically because our Welcome messages (in both Options 1 and 3) carry group state information, including the public ratchet tree and individualized encrypted group secrets, which grow directly in proportion to the group size. This finding is consistent with [133].

The overhead for group key updates via applying a commit message shows mixed trends but generally favors Option 3 in terms of bandwidth (message sizes) and processing time. Memory usage favored Option 1 due to the sparseness of intermediate nodes. We observed step-wise increases in both processing time and memory usage when group sizes grew to the next higher power of two, consistent with [133]. Their analysis, corroborated by [143], suggests this jump is due to processing the MLS ratchet tree as it grows an additional layer deeper, and we concur. The largest deltas in the saw-tooth pattern of Option 3's message sizes also occur when the group size increments by one from a perfect power of two. The saw-tooth pattern occurs due to taking measurements at the last member to join the group whose location and depth in the tree changes as group size increases. At group sizes equal to powers of two (i.e., "a perfect tree" described in [143]), the message sizes fit a logarithmic regression model with a high fidelity fit of $R^2 = 0.9998$. Otherwise, a harmonically repeating sawtooth pattern that trends logarithmically is observed for Option 3 message sizes. On the other hand, Option 1 update (commit) message sizes increased linearly, as expected, due to having the worst-case ratchet tree configuration.

Overall, our results confirm that Option 3 outperforms Option 1 and that our implementation generally scales linearly. We observed that high-fidelity logarithmic scaling occurs only under ideal group sizes within Option 3's configuration for bandwidth (message size) overhead, as supported by analysis from [143]. Otherwise, a saw-tooth scaling pattern with a logarithmic tendency occurs. In practical use cases, dynamic group membership can result in suboptimal ratchet tree configurations, so we do not expect this pattern to hold.

6.7 Future Work

This work focused on the viability of integrating MLS with BPSec using modest design choices and assumptions; future work should push these boundaries. Scaled testing on the magnitude of thousands of group members using devices more representative of fielded systems may be warranted, especially for settings where a leader-subordinate paradigm is

undesirable. Alternative DS designs (see Appendix 6.4), such as an eventually consistent design under a distributed network, should also be explored to make BPSec-MLS more useful in wider settings. Furthermore, due to the longevity of space missions, strategies to employ quantum-resistant BPSec-MLS (e.g., using NIST-recommended ML-KEM and ML-DSA) in efficient manners should be explored now in preparation for the emergence of cryptographically relevant quantum computers during the lifetime of space systems being fielded today.

6.8 Conclusion

The integration of MLS with BPSec marks a significant advancement in securing space communications that utilize the DTN stack. By leveraging MLS, BPSec gains efficient group key management and security features such as forward secrecy and post-compromise security, which were previously unattainable through traditional methods. Our implementation of BPSec-MLS, tested in two contrasting configurations under a DTN scenario, reveals both practical limitations and actionable solutions, paving ways to achieve more secure and efficient space network architectures. While linear scaling remains the norm except under ideal conditions that show logarithmic tendencies, our findings highlight the potential for future optimizations and broader adoption.

CHAPTER 7:
QUIC-MLS: Making QUIC Resilient for Disconnected Environments

Transforming QUIC from rigid handshakes to dynamic key agreement

As established in Chapter 1, the rigidity of traditional space security extends beyond preshared keys to encompass the synchronous, stateless handshake protocols used in modern Internet Protocol (IP) stacks that is being considered for space use. QUIC is being proposed for near-earth use for its efficient data transfer and integrated transport and built-in security [157]. Yet QUIC's two-party, synchronous handshake—inherited from Transport Layer Security (TLS)—fundamentally misaligns with space operations involving diverse constellations, ground station networks, and intermittent connectivity windows. Standard QUIC requires both parties to be simultaneously online to complete the handshake, an assumption that breaks down when satellites orbit in and out of line-of-sight, when ground stations have limited contact windows, or when relay nodes buffer and forward traffic across network disruptions. Furthermore, QUIC's stateless key establishment provides no post-compromise security: once a session key is compromised, all future communications in that session remain compromised. This rigidity—requiring synchronous handshakes and offering no self-healing capability—represents the same fundamental inadequacy identified in Chapter 1 for preshared-key systems, just manifested at the session-establishment layer.

Pillar II: Dynamic Key Agreement for IP Environments. Recall from Chapter 1 that Pillar II addresses the rigidity problem by integrating modern dynamic and asynchronous key agreement mechanisms into the two dominant network architectures for space communications: Internet Protocol (IP) networks and Delay Tolerant Networks (DTN). While terrestrial security protocols like TLS and QUIC excel in low-latency, continuously-connected environments, they degrade or fail entirely when confronted with the long propagation delays, intermittent connectivity, and limited bandwidth characteristic of space links. QUIC-MLS (this chapter) transforms QUIC from a one-to-one synchronous handshake protocol into an asynchronous, many-to-many continuous group key agreement and secure channel protocol. Together with BPSec-MLS (Chapter 6), which addresses the DTN component of

Pillar II, these integrations provide comprehensive key agreement coverage across heterogeneous space network architectures. The security properties established in Pillar I transfer to QUIC-MLS: the guardianship mechanisms from Chapter 4 ensure that even receive-only or emission-controlled QUIC endpoints can maintain post-compromise security, and the formal analysis methodology from Chapter 3 provides the tools to validate that QUIC-MLS preserves the secure channel guarantees of both QUIC and MLS.

Chapter Contribution and Impact. This chapter presents the integration of MLS in QUIC as an alternative handshake protocol to TLS, fundamentally retooling QUIC's key establishment component to be stateful, asynchronous, and multiparty-compatible. For the first time, this work tailors QUIC, a major terrestrial protocol ubiquitous in web applications [87], [111], to operate effectively in space settings while simultaneously advancing the state-of-the-art for terrestrial use cases requiring resilient group security.

At the highest level, this work marks a paradigm shift for both space and terrestrial communications. By replacing QUIC's synchronous, two-party TLS handshake with MLS's asynchronous, multiparty continuous key agreement, QUIC-MLS enables operational capabilities previously impossible with standard QUIC. Space constellations can establish group communications where satellites multicast telemetry to multiple ground stations without requiring pairwise session establishment with each recipient. Ground station networks can maintain continuous secure channels with satellites across intermittent contact windows, with the group state healing automatically through MLS updates rather than requiring expensive handshake renegotiations after each disruption. Most significantly, QUIC-MLS provides post-compromise security (PCS): if a session key is compromised, subsequent MLS updates heal the security without requiring the compromised party to actively participate—a critical capability for satellites that may be compromised while out of ground contact. Beyond space applications, QUIC-MLS expands terrestrial communications beyond traditional client-server architectures to asynchronous end-to-end multiparty communications, benefiting tactical military networks (where EMCON prevents synchronous handshakes), IoT mesh networks (where devices have intermittent connectivity), and collaborative edge computing (where participants join and leave groups dynamically).

At the protocol level, QUIC-MLS gains three major security and functional improvements over standard QUIC. First, post-compromise security (PCS) through MLS's continuous key

ratcheting ensures that compromises have bounded impact—future keys remain secure even after compromise. Second, truly fresh keys for 0-RTT (zero round-trip time) encryption eliminate the replay vulnerabilities that plague standard QUIC's 0-RTT mode, which reuses keys from previous sessions. Third, many-to-many communication support enables QUIC to be used in group, multicast, and broadcast network settings rather than being limited to pairwise client-server connections. These improvements transform QUIC from a rigid, pairwise, synchronous protocol into a flexible, group-capable, self-healing protocol aligned with the operational requirements of space communications.

At the technical level, this work achieves these results through computational cryptanalysis and implementation validation. The chapter expands the Flexible Authenticated and Confidential Channel Establishment (FACCE) security model used in analyzing QUIC to create a new group ACCE (gACCE) model that captures stateful, asynchronous, multiparty secure channels. The model is modified to use commit transcripts to synchronize group state asynchronously (rather than requiring synchronous handshake completion), and the notion of freshness is expanded to account for the post-compromise security inherited from MLS's continuous group key agreement. Honest partnering definitions are generalized from pairwise sessions to group memberships, capturing the semantics of multiparty secure channels. The work provides formal security proofs demonstrating that QUIC-MLS satisfies the gACCE security definitions, guaranteeing that the integration preserves the security properties of both QUIC and MLS. Additionally, a proof-of-concept implementation measures the overhead imposed by MLS integration, showing the costs of the additional features from the MLS overhead.

Advancing Secure Channel Analysis. This work significantly pushes the boundaries of the field of secure channel analysis by demonstrating how to combine a continuous key agreement protocol with an authenticated confidential channel to provide additional security properties and improved channel functionality. The gACCE model developed here represents a foundational contribution applicable beyond QUIC-MLS: any secure channel protocol seeking to integrate continuous key agreement can leverage this model to validate that the integration preserves security properties. The formal analysis methodology transfers the security guarantees proven for MLS (forward secrecy, post-compromise security, authentication) to the integrated QUIC-MLS protocol, providing mathematical certainty rather than informal assurance. This rigorous approach exemplifies the formal security

foundations established in Pillar I and demonstrates how those foundations enable protocol innovations in Pillar II with proven security guarantees.

Real-World Impact and Standardization. This paper has been accepted to the Security Standardization Conference (SSR) 2025, where it will be presented to the security standardization community including representatives from IETF working groups, standards bodies, and protocol implementers. The work is accompanied by an IETF Internet-Draft (Appendix A.3) specifying the integration procedures for QUIC-MLS, providing a concrete pathway for standardization and deployment. The European Space Agency has extended invitations to test QUIC-MLS on a system to be launched in 2026 alongside BPSec-MLS, providing real-world flight validation for both the IP and DTN components of Pillar II's dynamic key agreement framework. The proof-of-concept implementation provides a foundation for future production deployments, demonstrating that QUIC-MLS is not merely a theoretical construction but a practical protocol ready for operational use.

Joint Work and Contributions. This chapter presents the integration of MLS in QUIC as an alternative handshake protocol to TLS. This work is based on the paper detailed in Chapter 5 with the same set of co-authors: Benjamin Dowling (Kings College London), Britta Hale (Naval Postgraduate School), and Bhagya Wimalasiri (Kings College London). It has been reprinted, with permission, from all authors.[19] The candidate's contributions include outlining and reconciling the requirements from QUIC toward its handshake protocol, reconciling MLS delivery service requirements, and general integration mechanics (further detailed in the accompanying IETF draft). The candidate collaborated with co-authors on the protocol construction in Section 4, adapting the Flexible Authenticated and Confidential Channel Establishment as a security model. The candidate made initial drafts of gACCE definitions (definitions 2-4) and syntax and refined them with collaboration from coauthors. General writing and editing was joint work with the co-authors.

Chapter Abstract

Among standardization efforts for space and interplanetary network security, the Internet Engineering Task Force (IETF) is driving work on space network security, accounting for

[19]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.

the unique properties of space environments that make space communication challenging. This includes long, variable-length delays, packet loss, and intermittent end-to-end connectivity. Within these efforts, there is a focus on using IP-based protocols for security, and in particular the use of the QUIC protocol. This is unsurprising given QUIC's growing popularity and offer of optimization intended for reducing latency. However, QUIC uses the Transport Layer Security (TLS) key exchange handshake protocol, which was originally designed for 'connect and forget' style Internet connections at scale. It is also session-based, where protocol participants require reestablishment of the session for each reconnection – a costly maneuver in the space setting. Furthermore, TLS by default does not achieve strong post-compromise security properties within sessions, exhibiting a risk under long-lived connections, and need for synchronous handshakes to counteract this are in functional contrast to the space environment, which has intermittent end-to-end connectivity.

We address both drawbacks of QUIC by introducing QUIC-MLS: a variant of QUIC which replaces the session-based, synchronous TLS handshake with the standardized continuous key agreement protocol, Messaging Layer Security (MLS), which achieves asynchronous forward secrecy and post-compromise security. In addition to the design itself, we implement our design and provide benchmarks, and analyze our new construction in a formal cryptographic model.

7.1 Introduction

Space may be a vacuum, but humans are helping fill it up. According to the United Nations Office for Outer Space Affairs [158], humans have sent more than 2,500 objects into space in 2024, a trend that continues to increase. Since many of these objects are unmanned, they require communication protocols to enable control systems on Earth to make mission-critical adjustments.

However, secure communication protocols used in the terrestrial setting are not generally appropriate as-is for use in space. One can see this in the Internet Engineering Task Force (IETF) where the Taking IP to Other Planets (TIPTOP) working group has been outlining characteristics and use case requirements for space, that deployed protocols must address, including challenges such as "variable delays", "frequent and long interruptions of communications" and "one-way / unidirectional links". Within the same working group, the

first and primary focus for deployable secure IP-based protocols for space networking is the QUIC protocol [105], [159].

The internet-wide adoption of the QUIC protocol is contributing to its use IP-based space communications [134], [157], [160], delay tolerant networks [161], and vehicular adhoc mobile networks [162]. In internet usage, generic QUIC traffic held the #3 spot for total volume and accounted for 6-8% of global internet traffic in 2023 [163]. An arguable driver in the use of QUIC over more traditional protocols such as TLS 1.3 is the efficiency gains and latency reductions from its integrated approach across network layers. While Transport Layer Security (TLS) version 1.3 must be combined with some transport protocol, and therefore is not optimized for combination with any particular one, QUIC has the benefit of optimized integration of transport and security protocol aspects. Multiplexing capabilities are another benefit of the QUIC protocol. However, QUIC has some characteristics that make it difficult to support a deep space use-case.

QUIC inherited the legacy key establishment characteristics of its predecessor, TLS 1.3, specifically for key exchange (KEX). QUIC uses the TLS 1.3 KEX, which incurs at least one full round trip (RTT) even with the QUIC optimizations [105]. The 0-RTT TLS mode could intuitively be considered as a solution to this inefficiency, but has known weaknesses [164], [165]. This leads to a tension between security and the QUIC goals of reducing latency. Within terrestrial uses these issues are not prominent, but within space settings, where transmission delays can take minutes or even hours, the both latency and security risks become more prominent. We highlight that the core functionality of the TLS 1.3 KEX is negatively impacted by the challenges of the deep space setting. A 1 round trip time (RTT) synchronous handshake requires bidirectional, synchronous communication in order to achieve forward secrecy.

However, a fundamental review of the core challenges reveals a path forward. The TLS 1.3 KEX is a session-based key agreement mechanism, meaning that parties are not assumed to maintain state between sessions. This is an artifact of TLS's original design for use on the Internet, where tens of thousands of new and changing connections make it undesirable to maintain state between instances of client connections. In contrast, in uses such as space settings, where the set of communication parties is usually small and can even be static, state can be maintained between communications. This obviates the need for the original

session-based design and opens the aperture for lower latency together with improved security through use of continuous session protocols. Moreover, given that the use of TLS KEX within QUIC is itself a selected handshake option, a such more appropriate handshake alternative could likewise replace it.

Continuous session protocols, a.k.a. continuous key agreement or ratcheted protocols, are stateful protocols that enable key rotation on demand, perhaps according to a schedule, which removes the need for session handshakes at the start of every connection and therefore reduces RTT. This does not necessarily come at the cost of security; in fact, continuous session protocols have been shown to achieve properties that were unattainable under traditional TLS [113]. Normal TLS sessions provide forward secrecy (FS), where *past* communications remain secure if the current session is compromised. In contrast, a continuous session protocol can achieve both FS and post-compromise security (PCS), thus "self-healing" and locking out adversarial access *after* a key compromise [166].

TLS and QUIC have slowly been moving towards a continuous setting, namely with additions for session resumption. Yet these session-based protocols have never been analyzed as continuous session protocols nor fundamentally designed for those goals, and even recent work on session resumption is still synchronous in nature, contrary to the functional needs of space communications. Thus, in this work we show how to leverage another standardized, yet continuous key agreement protocol, Messaging Layer Security (MLS) [3], as an alternative KEX within QUIC in order to create a continuous session protocol QUIC variant. This reduces the latency and security issues of QUIC within space that were previously noted [164], [165], supports PCS, and still maintains the networking functionality optimizations of QUIC. Furthermore, it opens up potential for moving uses of QUIC beyond the client-server setting [157], [162]. As an added benefit, since MLS was designed for scalable asynchronous group key agreement [3], our solution introduces options beyond pairwise QUIC, namely reducing the overhead of key updates in QUIC for larger multi-device configurations, bringing it down from $O(n^2)$ to $O(n \cdot log(n))$ for groups of size n [131]. This benefit is optional, as even 1-to-1 (group size of 2) settings obtain the other benefits of the construction.

In this work we provide a cryptographic design for using the MLS KEX with QUIC, a security model and computational analysis of MLS-QUIC, and benchmark testing of the

solution. For generality we model and analyze for the more complex multi-party case, where group key agreement resulting in a shared secure QUIC data channel. This is fully extensible to the 1-to-1 communication scenario where MLS-QUIC is used in a typical pair-wise case, e.g., client-to-server or client-to-client. Our MLS-QUIC variant turns QUIC from a stateless session based protocol to a stateful continuous protocol opening doors for wider adoptions beyond the traditional web applications, and is suitable for the challenges highlighted by the IETF.

7.2 Preliminaries

Here we describe the abstract models used to capture key establishment methods such as TLS, QUIC, and MLS.

7.2.1 Secure Channel Establishment

Session-based key establishment and continuous key agreement (CKA) represent two distinct paradigms for establishing a secure channel. Session-based key exchange protocols like TLS or Internet Key Exchange (IKE) execute so-called handshakes to negotiate cryptographic parameters and perform cryptographic operations to derive an ephemeral session key used between two parties that wish to securely communicate. In particular, handshakes are executed independently, with certificates transmitted at each establishment to authenticate the new connection. This approach is used for one-to-one sessions that are short-lived and benefit from statelessness between sessions.

In contrast, CKA protocols, such as the Signal protocol, have often been used as the cryptographic core of secure messaging: they establish a shared cryptographic state between communicating parties and allow for new keys to be generated and updated incrementally and asynchronously (i.e., *without* additional handshakes upon re-connection). This allows for asynchronous key updates, granular control key lifetime between refresh and update, and recovery from adversarial state compromise. For systems that can afford to manage and store cryptographic states, CKA protocols are often more secure and efficient than session based ones. CKAs are extended to the multi-user setting as Continuous Group Key Agreement (CGKA), enabling groups of users to establish a series of shared keys consistently. The TreeKEM sub-protocol of the MLS protocol [3] is a prime example of a deployed CGKA.

Here we provide the formal definition of a CGKA.

Definition 16 (CGKA from [167])**.** *A continuous group key-agreement (CGKA) scheme* $\mathsf{CGKA} = (\mathsf{KGen}, \mathsf{Create}, \mathsf{Add}, \mathsf{Join}, \mathsf{Upd}, \mathsf{Rem}, \mathsf{Commit}, \mathsf{Process}, \mathsf{Key})$ *consists of the following algorithms:*

- ***Key Generation****:* $(\mathrm{pk}, \mathrm{sk}) \leftarrow \mathsf{KGen}()$ *samples a fresh public/secret key pair.*
- ***Group creation****:* $\gamma \leftarrow \mathsf{Create}()$ *takes no input and returns a fresh group state* γ *containing only the party running the algorithm. This represents the first* epoch *of a new session.*
- ***Add****:* $(\gamma', \mathrm{p}) \leftarrow \mathsf{Add}(\gamma, \mathsf{id}_t, \mathrm{pk}_t)$ *proposes adding a new member to the group. On input a protocol state* γ*, identity of the new member* id_t *and their public key (generated by* KGen*), it outputs an updated state* γ' *and an* add proposal message p.
- ***Remove****:* $(\gamma', \mathrm{p}) \leftarrow \mathsf{Rem}(\gamma, \mathsf{id}_t)$ *proposes removing a member from the group. On input a protocol state* γ *and identity* id_t*, it outputs an updated state* γ' *and* remove proposal message p.
- ***Update****:* $(\gamma, \mathrm{p}) \leftarrow \mathsf{Upd}(\gamma)$ *proposes updating the member's key material. It outputs an updated state* γ' *and an* update proposal message p.
- ***Commit****:* $(\gamma', \mathrm{commit}, \mathrm{welc}) \leftarrow \mathsf{Commit}(\gamma, \overrightarrow{\mathrm{p}})$ *applies (aka* commits*) a vector of proposals to a group. The output consists of an updated protocol state* γ'*,* commit message *and a (potentially empty)* welcome message *(depending on if any add proposal messages were included in* $\overrightarrow{\mathrm{p}}$*).*
- ***Join****:* $(\gamma', \overrightarrow{\mathsf{G}}, \mathsf{id}_i) \leftarrow \mathsf{Join}(\mathrm{sk}, \mathrm{welc})$ *allows a party with secret key* sk *(generated by* KGen*) to join a group with a welcome message* welc*. The outputs are: an updated protocol state* γ'*, a group* roster $\overrightarrow{\mathsf{G}}$ *(i.e., a set of IDs listing the group members), an epoch ID, and the ID of the inviter (i.e., the party that created the welcome message).*
- ***Process****:* $(\gamma', \mathrm{info}) \leftarrow \mathsf{Process}(\gamma, \mathrm{commit}, \overrightarrow{\mathrm{p}})$ *processes an incoming commit message and the corresponding proposals to output a* commit info message info *and an updated group state* γ' *which represents a new epoch in the ongoing CGKA session. The commit info message captures the semantics of the processed commit and has the form:* $\mathrm{info} = (\mathsf{id}, (\mathrm{propSem}_1, \ldots, \mathrm{propSem}_z))$ *where* id *is the ID of the sender of the commit message the* propSem*vector conveys the semantics of the committed add and remove via triples of the form* $\mathrm{propSem} = (\mathsf{id}_s, \mathsf{op}, \mathsf{id}_t)$*. Here,* id_s *denotes the identity of the proposal's sender* $\mathsf{op} \in \{\text{“addP”}, \text{“remP”}\}$ *is the proposal's type and* id_t *is the identity of the proposal's target (i.e. the party being added or removed).*

- ***Get Group Key***: $(\gamma', K) \leftarrow \mathsf{Key}(\gamma)$ *outputs the current group key for use by a higher-level application and deletes it from the state.*

We next cover the TLS, QUIC, and MLS standards of the of the IETF, and how their components fit together.

7.2.2 Transport Layer Security

As a session based protocol, main goal of TLS [104] has always been to establish a secure communications channel across an untrusted network (the Internet). In what began with Secure Socket Layer (SSL) protocol created by Netscape, the TLS protocol has been upgraded multiple times by the IETF, resulting in the current iteration of TLS 1.3 (hereto simply referred as TLS) [104]. TLS itself is comprised of sub-protocols: the handshake protocol which establishes keys (TLS-HS), the record layer protocol (TLS-RL) which uses those keys to provide a secure channel for data transmission using ,authenticated encryption with associated data (AEAD) and an alert protocol. TLS supports session resumption using an established secret for faster establishment of future sessions with the same client (e.g. 0RTT).

7.2.3 QUIC

QUIC was first developed by Google in 2012 and later adopted and refined as an internet standard by the IETF [105]. Because QUIC uses TLS-HS for security key establishment [110], its evolution has been tied to TLS's evolution. For analysis purposes, we will also use a similar break down in reference to QUIC, namely as having a handshake component (QUIC-HS $\approx$ TLS-HS) for key agreement and a record layer (QUIC-RL) component. Unlike TLS, once a session key is established in QUIC (i.e., after QUIC-HS), it can be updated using a key-phase bit which is flipped between parties to indicate the use of a new key from a key derivation function (KDF) applied to the old key. This is a symmetric key ratchet, and should not be interpreted as introducing new randomness or achieving PCS. Furthermore, QUIC offers a 0-RTT connection mode for previously connected clients through relying on a server-signed configuration file stored and presented by the client; this is based on the TLS 0-RTT mode.

7.2.4 Messaging Layer Security (MLS) Protocol

The MLS (RFC 9420) [3] standard was designed for end-to-end encryption, providing users with secure and scalable communications. Similarly to QUIC, MLS can be broken down into key exchange and security channel or "record layer" phases; however, as it was originally intended for application messages, the "record layer" phase is at the application layer. Notably, as with TLS and QUIC, the networking layer use of the overall protocol is determined by the "record layer" protocol definition—the key exchange portion of MLS is not tied to a particular networking layer use. This allows us to modualarly change the QUIC-HS from TLS-HS to MLS, as shown in Figure 7.1.

Instead of session-based handshake negotiation for key exchange, MLS and similar CKAs establish the cryptographic session state once and then update keys as needed or as scheduled (potentially asynchronously) without additional round-trip negotiations. Such key updates are also what enables PCS. MLS allows for a notable differentiation from prior pairwise CKA protocols in that it supports *groups*, meaning that not only is it a CKA, but a *continuous group key agreement* (CGKA). Group members can efficiently perform the CGKA with entity authentication and key indistinguishability security guarantees [168]. Group members/clients can also be added or removed, enabling adaption.

7.3 Related Works

MLS has undergone extensive computational analysis as a CGKA [137], [167] resulting in major changes to the protocol by the IETF. Similarly, symbolic analysis (in the Dolev-Yao model) has been used to analyze MLS [136]; such symbolic analysis abstracts cryptographic primitives as "black boxes". In this work, we use the computational analysis approach to analyze and prove security QUIC-MLS, leveraging existing models from work on CGKAs.

Work on CKAs include analyses on the Signal protocol. While there is extensive work in this domain [53], [169]–[172], we opt to use MLS as it is already standardized. This fits with the overall theme of shifting out one standardized key establishment (TLS-HS) with another (MLS) within the QUIC protocol.

Other related works include analyses of QUIC, especially the work done on record layer analysis [173], [174] and TLS vs. QUIC [175]. As noted before, prior work has raised

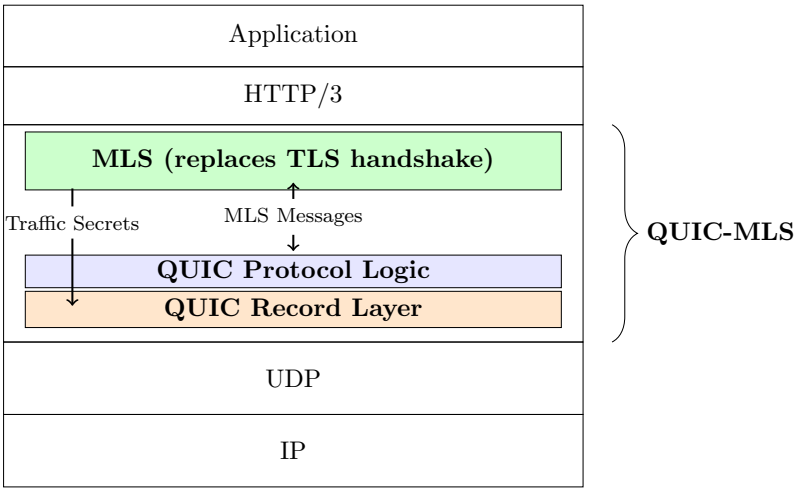


Figure 7.1. High level overview of QUIC-MLS, showing slot-in replacement of TLS-HS with MLS.

concerns on use of the TLS 0-RTT option [164], [176].

7.4 Protocol Design and Mechanics

In this section we provide a high level overview of how MLS can be incorporated into QUIC in terms of requirements analysis and protocol construction. QUIC combines key agreement and secure channel management into one protocol: for analysis purposes we break these out as Handshake (QUIC-HS) and Record Layer (QUIC-RL) responsibilities, respectively, to reflect the TLS 1.3 components that they are derived from (i.e. TLS Handshake and Record Layers). Low level details such as wire formatting are out of scope for this work.

7.4.1 General Requirements

QUIC has three general requirements of its handshake component: 1) authenticated key exchange, 2) transport parameter authentication, and 3) authenticated application protocol negotiation [105]. MLS can be used as a drop-in replacement for TLS from the perspective of QUIC-RL, as MLS meets the minimal requirements to be an authenticated key agreement protocol. With minimal additions to the MLS `GroupInfo` to specify transport parameters and application protocols the other requirements are met with MLS as well (see Section 7.4.4).

In satisfying the QUIC requirements, MLS has remaining requirements of its own to function properly. Specifically, to facilitate client communication, MLS relies on two abstract services: an Authentication Service (AS) and a Delivery Service (DS). The AS, as previously mentioned, attests to the bindings of identifier to public key material which can be adapted from existing PKI network services. For our analysis purposes, we assume that the AS functions as it currently does through existing PKI as used by TLS. The DS receives and distributes messages between group members and ensures the ordered delivery of MLS handshake messages [177]. A *strongly consistent* DS ensures messages arrive in order and that all members agree on that order. In web applications, for example, a strongly consistent DS can be enforced through reliable and order enforcing transport protocols or via a centralized server. An *eventually consistent* DS only guarantees that messages arrive (eventually) to all members but makes no guarantees on their ordering, leaving clients responsible for reconciliation. An eventually consistent DS is endemic to decentralized, peer-to-peer, or hop-by-hop networks.

While a protocol designed for an eventually consistent DS can be streamlined for an strongly consistent DS, the reverse is not true. Thus, we design QUIC-MLS for an eventually consistent DS with tunable parameters to fit the reality of the underlying network up to a strongly consistent DS. To provide a clear mechanism to ensure deterministic handling of group state updates and avoid race conditions arising from concurrent update proposals, we propose introducing a designated group administrator responsible for all membership changes which may change in a fixed order (e.g. based on the sequence in which members joined the group) from epoch to epoch. This is consistent with the notion of epoch leaders in previous work [125] except that only the administrator may initiate add or remove proposals; any such proposals from other group members will be ignored. Additionally, to ensure all members can synchronize group state, a "commit transcript" of a tunable length is appended to all messages sent. In the worst case of an eventually consistent DS, this may contain all previous commits since the start of the session. The length can be parameterized to the reliability of the network; with a strongly consistent DS, the length is zero.

7.4.2 Construction Overview

Our QUIC-MLS construction, detailed in Figure 7.2, is composed of KGen, Init, Enc, and Dec. KGen is used to generate public keys, simply by calling CGKA.KGen. In Init, a group

admin instantiates the group session (using the intended partner's public keys) using the CGKA algorithms Create, Add and Commit. All welcome messages generated by Init are sent to recipients via QUIC-RL who call Init as a participant (ρ = p) upon receipt to join the group (via CGKA.Join).

Enc enables payload encryption via QUIC 0RTT, and generation of group control proposals (e.g., Add, Rem, Upd). Similarly, Dec decrypts payload messages and processes proposals and commits. To ensure reliability, ciphertexts contain "a commit transcript" ast with a tunable length tl to reconcile group state synchronization. In an asynchronous scenario with a two-party group and strongly consistent DS the tl can be set to 1 when assuming parties are never more than 1 epoch apart. If Dec updates the session's group state, they derive a new key (via CGKA.Key) for payload encryption via QUIC 0RTT.

7.4.3 Limitations

While our approach is intended as a clean substitution of TLS in QUIC, it introduces a limitation on non-repudiation due to the nature of group communications versus two-party communications. QUIC does not sign messages because non-repudiation is implicit in the channel among two parties who have completed authenticated key exchange. In a group channel, however, insiders can impersonate one-another if messages are not signed with the sender's private key. To avoid this, group channel messages for QUIC-MLS can also be signed using public and private keys of clients used for MLS. However, this composition requires further analysis as we must then also consider allowing adversary access to a signing and verification oracle within the context of the CGKA security model. We leave this for future work and assign handling non-repudiation of insiders to the application layer.

7.4.4 Changes to Respective Protocol Standards

We make no changes to the underlying cryptographic components of MLS or QUIC-RL as standardized. However, we change the semantics of some QUIC terminology in order to adapt MLS. The association of QUIC security levels to certain message types throughout the QUIC handshake is obviated by the single stage MLS key agreement mechanism. As observed in Figure 7.3, clients continue to use 0-RTT messages for sending data after the initial volley of messages. This choice conveys the efficiency of QUIC-MLS and that MLS security is obtained from the session start.

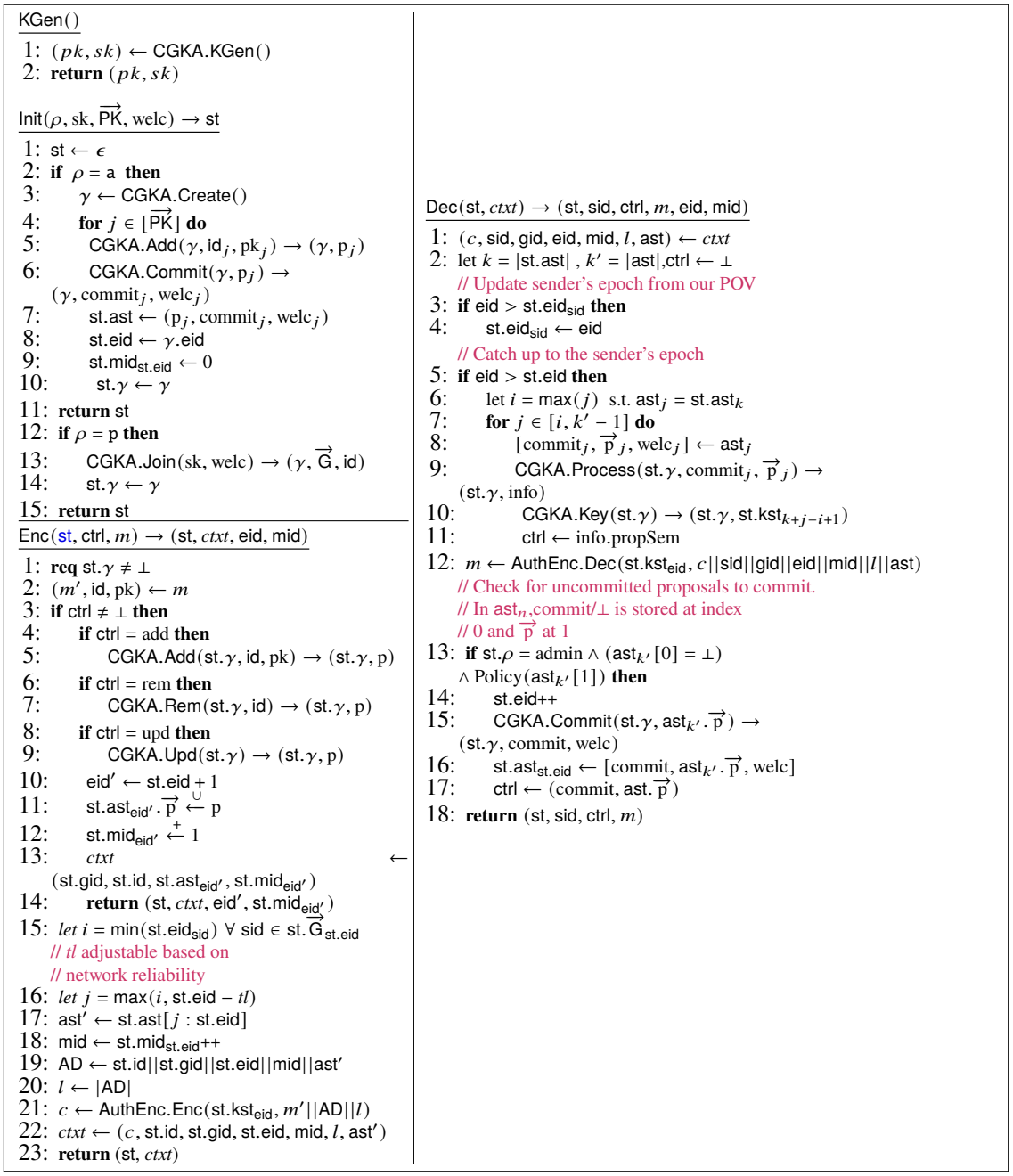


Figure 7.2. QUIC-MLS algorithms. All algorithms abort if any CGKA sub-algorithms return $\perp$.

To address the requirements for QUIC-HS to offer authenticated transport parameter and application protocol negotiation, we utilize the MLS Safe Extension [178] to add fur-

ther context to the group. Specifically, for application-layer protocol agreement, a list of protocols would be included in the `data` vector of the `ComponentData` struct within the `app_data_dictionary` in the `GroupContext`. For transport layer parameters advertisement, this can simply be a list added to the `aad_item_data` with an appropriate `component_id` in the `SafeAAD` struct to be more transparent. These changes have been proposed to the MLS working group (see pull requests for [178]).

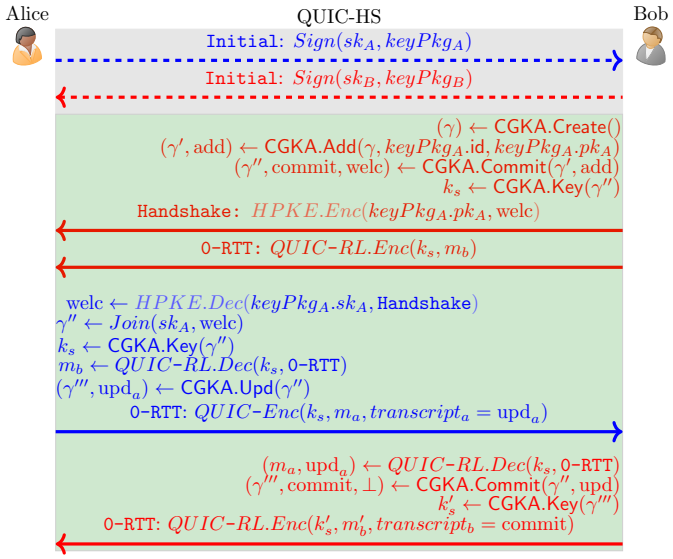


Figure 7.3. QUIC-MLS execution with two parties. The use of Initial QUIC packets to send MLS Key Packages is optional if key packages are installed out-of-band/prior.

7.5 Benchmarking QUIC-MLS

In this section, we present benchmarking results for our proposed QUIC-MLS construction. As part of our benchmarking efforts, we developed a reference implementation [20] prototyping QUIC-MLS functionality in Go, leveraging the existing libraries `(quic-go)`, `(go-mls)`, and `(crypto/aes)` `(crypto/tls)`, [179] [180], [181]. We note, at the time of writing, our reference implementation is not a complete integration of MLS into QUIC. Instead, it serves as a prototype for assessing the computational and bandwidth costs associated with our proposed QUIC-MLS construction. This is due in part to the tight coupling of the `(quic-go)` [179] library with Go's native TLS stack, making it non-trivial to fully

[20] https://osf.io/qun62/?view_only=e814346aa4334fe1be43a81a89f01c8d

replace TLS with MLS without extensive refactoring. However, we simulate the core functionality of the QUIC record layer by encrypting application data using AEAD schemes with session secrets derived from MLS, enabling a functionally comparable evaluation of performance and overhead.

We instantiated QUIC-MLS with the cipher suite `P256_AES128GCM_SHA256_P256` and benchmarked its average performance across group sizes of 2, 5, 10, 20, 50, and 100 members, using 100 iterations per group size. For each epoch, the derived `ExporterSecret` was utilized for `0-RTT` encryption of application-layer data.

In Table 7.1, we compare the average performance of the cryptographic procedures involved in the key package and group creation stages of QUIC-MLS. During key package creation, QUIC-MLS issues each member a set of credentials that can be leveraged in subsequent group operations. We observe that average creation times increase steadily with group size, scaling gradually as more members are added. Within the group creation stage, QUIC-MLS facilitates the formation and management of dynamic groups, where a designated group *admin* handles Add and Remove proposals by processing Join requests and issuing corresponding Welcome messages. Reflecting the increased overhead of managing group state and the complexity of underlying cryptographic operations, average group creation time exhibits a more dramatic rise, particularly as the group becomes large.

Group Size	KeyPackage Creation	Group Creation
2	$\approx 0.373 \pm 0.235$	$\approx 0.946 \pm 0.723$
5	$\approx 1.42 \pm 0.564$	$\approx 6.72 \pm 1.781$
10	$\approx 2.12 \pm 0.508$	$\approx 15.79 \pm 2.261$
20	$\approx 3.11 \pm 0.9$	$\approx 45.97 \pm 7.129$
50	$\approx 10.09 \pm 2.613$	$\approx 191.49 \pm 27.395$
100	$\approx 17.566 \pm 6.253$	$\approx 712.57 \pm 267.381$

Table 7.1. Average runtime (in milliseconds over 100 iterations) of cryptographic operations during key package and group creation in QUIC-MLS, evaluated on an **Intel(R) Core(TM) i7-11370H CPU @ 3.30GHz**.

In Table 7.2, we compare the cost of QUIC-MLS Update operations in terms of both performance and bandwidth overhead. In particular, we model a use case involving a two-member group, where one member processes committed Update messages from the other

```
QUICMLSUpdate { group_id, epoch, sender, authenticated_data,
content: Commit { proposals: [empty], path: DirectPath {
nodes: [...] } }, signature, confirmation_tag }
```

Figure 7.4. QUIC-MLS packet structure for maintaining admin state (See Fig. 7.2 ast)

after 5, 10, 20, 50, and 100 Commit operations. This setup is intended to simulate a scenario in which a QUIC-MLS group member remains part of the group but is inactive for an extended period. When the inactive member resumes communication, they must update their local group state to the latest epoch before re-engaging with the group. Figure 7.4 illustrates the expected format of a QUIC-MLS Update packet within our construction. Crucially, to resume sending encrypted messages over the QUIC record layer, the inactive member must first derive the most recent `ExporterSecret` from the updated group state. In a catch-up scenario, the inactive member processes previously committed `Commit` messages to update its local state. The `path` field provides the encrypted key material necessary to advance to the latest epoch.

Our evaluation shows that epoch synchronization time grows predictably with the number of missed updates, reflecting the accumulation of cryptographic state. Sync message size follows a similar trend, increasing with the volume of group information that must be communicated.

Missed Commit Count	State Sync Time	Message Size
5	$\approx 5.41 \pm 0.501$	$\approx 6,119$
10	$\approx 10.02 \pm 2.873$	$\approx 11,204$
20	$\approx 20.01 \pm 3.325$	$\approx 21,369$
50	$\approx 51.55 \pm 8.062$	$\approx 51,903$
100	$\approx 101.43 \pm 22.756$	$\approx 102,740$

Table 7.2. Average runtime and bandwidth evaluation (in milliseconds for sync time and bytes for message size over 100 iterations) during epoch Update for inactive member in QUIC-MLS evaluated on **Intel(R) Core(TM) i7-11370H CPU @ 3.30GHz**.

7.6 Security Analysis of QUIC-MLS

Both TLS and QUIC have been extensively analyzed under various provable security approaches that either separate the key exchange and secure channel phases or consider them

together as an intertwined protocol. Security goals commonly analyzed across the various approaches include key indistinguishability, authenticated and confidential channel establishment, protection against replay attacks, defense against state reveals, and forward secrecy. Jager et al. [182] formalized the monolithic approach as *authenticated and confidential channel establishment* (ACCE) protocols, which are separated into pre-accept and post-accept phases (similar to the TLS Handshake and Record Layer phases [183]). Follow-on work from Lychev et al. in [184] take a more holistic approach in analyzing QUIC and define the Q-ACCE model based on how QUIC uses TLS with multiple keys for key agreement and data messages in a two stage approach. Fischlin and Günther in [185] analyze QUIC in their multi-stage key exchange (MSKE) model in the Bellare and Rogaway style of [186], highlighting their model's composability with any symmetric-key application protocol. Tangential to this composability emphasized approach, Dowling et. al. in [187] propose the Flexible ACCE (fACCE) model which eliminates the boundary between the handshake and channel yet is able to measure fine-grained security guarantees at each phase.

7.6.1 gACCE for QUIC-MLS

We use a modified fACCE model we call gACCE in our analysis of QUIC. To this end, we make some changes to the fACCE primitive to accommodate group channel security in what we dub gACCE (Definition 17).

Definition 17 (Flexible ACCE for Groups ([187])**.** *A flexible group ACCE protocol* gACCE *is a tuple of algorithms* $\mathsf{gACCE} = (\mathsf{KGen}, \mathsf{Init}, \mathsf{Enc}, \mathsf{Dec})$ *defined over a secret key space* $\mathcal{SK}$*, a public key space* $\mathcal{PK}$ *and a state space* $\mathcal{ST}$*. The syntax of a* gACCE *protocol is as follows:*

- $\mathsf{KGen} \rightarrow_{\$} (\mathrm{sk}, \mathrm{pk})$ *generates a long-term key pair where* $\mathrm{sk} \in \mathcal{SK}, \mathrm{pk} \in \mathcal{PK}$.
- $\mathsf{Init}(\rho, sk_i, \overrightarrow{\mathsf{PK}}_{\overrightarrow{\mathsf{G}}}, \mathrm{welc}, ad) \rightarrow_{\$}$ *st initializes a session to begin communication, where* sk_i *(optionally) is the caller i's long-term secret key,* $\overrightarrow{\mathsf{PK}}_{\overrightarrow{\mathsf{G}}}$ *(optionally) are the long-term public keys of the intended group session partners* $\overrightarrow{\mathsf{G}}$ *such that* $i \notin \overrightarrow{\mathsf{G}}$*,* $\rho \in \{\mathsf{a}, \mathsf{p}\}$ *the session's role (i.e. admin or partner),* ad *is the data associated with this session, and* $sk \in \mathcal{SK} \cup \{\perp\}, \mathrm{pk} \in \mathcal{PK} \; \forall \; \mathrm{pk} \in \overrightarrow{\mathsf{PK}}_{\overrightarrow{\mathsf{G}}}, ad \in \{0,1\}^*, st \in \mathcal{ST}$.
- $\mathsf{Enc}(\mathsf{st}, \mathsf{ctrl}, m) \rightarrow_{\$} (\mathsf{st}', ctxt, \mathsf{eid}, \mathsf{mid})$ *enables a group member to encrypt a message, or to propose to* evolve *the group by adding, removing or updating members of the group. It*

takes as input a state st*, an optional control message* ctrl *and a message m, and outputs new state st′, and a ciphertext ctxt (where* $\mathsf{st}, \mathsf{st}' \in \mathcal{ST}$, $\mathsf{ctrl} \in \{\perp, \mathsf{Add}, \mathsf{Upd}, \mathsf{Rem}\}$, $m \in \{0,1\}^*$ *when* $\mathsf{ctrl} = \perp$ *otherwise* $m \leftarrow (\mathsf{st.id}, \mathrm{pk}_{\mathsf{st.id}})$, $ctxt \in \{0,1\}^* \cup \{\mathrm{propSem}\}$, $\mathsf{eid} \in \mathbb{N}$ and $\mathsf{mid} \in \mathbb{N}$).

- $\mathsf{Dec}(\mathsf{st}, ctxt) \rightarrow_{\$} (\mathsf{st}', \mathsf{id}, \mathsf{ctrl}, m, \mathsf{eid}, \mathsf{mid})$ *processes any updates to the group or decrypts messages by other group members. It takes as input a state* st *and ctxt and outputs new state st′, sending partner's identifier* id*, message m, control message* $\mathsf{ctrl} \in \{\perp, \mathsf{Upd}, \mathsf{Add}, \mathsf{Rem}, \mathsf{Commit}\}$*, and epoch and message counters* eid *and* mid*, where* $\mathsf{st} \in \mathcal{ST}$, $st' \in \mathcal{ST} \cup \{\perp\}$, $m \in \{\{0,1\}^* \cup \perp\}$, $ctxt \in \{0,1\}^*$, $\mathsf{mid} \in \mathbb{N}$ and $\mathsf{eid} \in \mathbb{N}$. *If* $m = \perp$*, then this denotes a failure to process the message.*

To capture security in a group setting, we need to modify the fACCE's matching conversations and correctness definitions. Two group members share a *matching conversation* when they agree on the *control messages* used to evolve the group, and that any received payload messages are the same as the original sent message. The control messages are held in CT and the payload application messages are held in AT. Due to the propose-and-commit nature of MLS, we introduce the concepts of a *confirmed transcript* and *opportunistic transcript* for CT as an expansion of the message-stage-ciphertext transcripts in [187]: group members can propose changes that are (validly) discarded by the admin.

The confirmed transcript maintained by a group member party is a pair of structures ordered by the currently member $\mathsf{id} \in \overrightarrow{\mathsf{G}}$ and indexed by epoch eid. $(\mathrm{CT}^{\mathrm{c,e}}_{\mathsf{id,eid}}, \mathrm{CT}^{\mathrm{c,d}}_{\mathsf{id,eid}})$: where $\mathrm{CT}^{\mathrm{c,e}}_{\mathsf{id,eid}}$ records all messages that have been encrypted (sent) and $\mathrm{CT}^{\mathrm{c,d}}_{\mathsf{id,eid}}$ that holds decrypted messages (received) from a party $n \in \overrightarrow{\mathsf{G}}$ in epoch eid. The opportunistic transcript is similarly composed of a $\mathrm{CT}^{\mathrm{o,e}}_{n.\mathsf{eid}}$ and $\mathrm{CT}^{\mathrm{o,d}}_{n.\mathsf{eid}}$ except these only hold *proposed* group *control messages*, respectively. The opportunistic transcript control messages have not been committed: when they do, they are moved to the *confirmed* transcript. Together, these allow gACCE to both match application messages and match epoch evolutions through staging pending proposals until a successful decryption of a commit message triggers a new epoch. We now define correct execution of an gACCE protocol. Finally, the application transcript $\mathrm{AT}^{x}_{\mathsf{id,eid}}$ (where $x \in \{e, d\}$) contain all application messages (i.e. payload) encrypted and decrypted by id in epoch eid. Note that the decrypted $\mathrm{CT}^{\mathrm{d}}_{\mathsf{id,eid}}[\mathsf{id}']$ is indexed by the claimed sender partner identifier id′.

Definition 18 (Correctness of gACCE (adopted from [187])**.** *A* gACCE *protocol is* correct *if, for any pair of keys* $(\mathrm{sk}_i, \mathrm{pk}_i)$ *and* $(\mathrm{sk}_j, \mathrm{pk}_j)$ *output by* KGen *such that* $(i, j) \in \overrightarrow{\mathsf{G}}$*, with* $i \neq j$*, the following holds. Let* $ad \in \{0,1\}^*$*. The session states are initialized as* $\mathsf{st}_i^{\mathsf{a}} \xleftarrow{\$} \mathsf{Init}(\mathsf{a}, \mathrm{sk}_i, \{\mathrm{pk}_{\overrightarrow{\mathsf{G}}}\}, ad)$ *and* $\mathsf{st}_j^{\mathsf{p}} \xleftarrow{\$} \mathsf{Init}(\mathsf{p}, \mathrm{sk}_j, \{\mathrm{pk}_{\overrightarrow{\mathsf{G}}}\}, ad)$*. Application transcripts are initialized as* $\mathrm{AT}^*_* \leftarrow \epsilon$*, and control transcripts as* $(\mathrm{CT}^{*,*}_{i,*}, \mathrm{CT}^{*,*}_{j,*}) \leftarrow \epsilon$ *(indexed by* $\mathsf{id} \in \overrightarrow{\mathsf{G}}$*,* eid*). Then, for all sequences of operations* $\Big((op^0, \mathsf{id}^0, \mathsf{ctrl}^0, \rho^0, m^0), \ldots, (op^n, \mathsf{id}^n, \mathsf{ctrl}^n, \rho^n, m^n)\Big)$*, with* $0 \leq l \leq n$*, the following holds:* $op^l \in \{e, d\}$*,* $\mathsf{id}^l \in \{i, j\}$*, control actions* $\mathsf{ctrl}^l \in \{\perp, \mathsf{Add}, \mathsf{Rem}, \mathsf{Upd}, \mathsf{Commit}\}$*, role* $\rho^l \in \{\mathsf{a}, \mathsf{p}\}$*, and message* $m^l \in \{0,1\}^*$*. This* gACCE *correctness definition holds provided the protocol is executed precisely as follows.*

- *if* $op^l = e$*, invoke:* $\mathsf{Enc}(\mathsf{st}^{\rho^l}_{\mathsf{id}^l}, \mathsf{ctrl}^l, m^l) \xrightarrow{\$} (\mathsf{st}^{\rho^l}_{\mathsf{id}^l}, ctxt^l, \mathsf{eid}, \mathsf{mid})$ *and,*
  - *if* $\mathsf{ctrl}^l = \perp$*, update:* $\mathrm{AT}^{\mathrm{e}}_{\mathsf{id}^l,\mathsf{eid}} \leftarrow \mathrm{AT}^{\mathrm{e}}_{\mathsf{id}^l,\mathsf{eid}^l} \,\|\, (\mathsf{id}^l, \mathsf{mid}, m^l, ctxt^l)$
  - *else if* $\mathsf{ctrl}^l \in \{\mathsf{Add}, \mathsf{Rem}, \mathsf{Upd}\}$*:*
    - *if* $\rho_{\mathsf{id}^l} = \mathsf{a}$*, update:* $\mathrm{CT}^{\mathrm{c,e}}_{\mathsf{id}^l,\mathsf{eid}^l} \leftarrow \mathrm{CT}^{\mathrm{c,e}}_{\mathsf{id}^l,\mathsf{eid}^l} \,\|\, (\mathsf{eid}^l, \mathsf{ctrl}^l, m^l, ctxt^l)$
    - *if* $\rho_{\mathsf{id}^l} = \mathsf{p}$*, update:* $\mathrm{CT}^{\mathrm{o,e}}_{\mathsf{id}^l,\mathsf{eid}^l} \leftarrow \mathrm{CT}^{\mathrm{o,e}}_{\mathsf{id}^l,\mathsf{eid}^l} \,\|\, (\mathsf{eid}^l, \mathsf{ctrl}^l, m^l, ctxt^l)$
- ***OR****, if* $op^l = d$*, invoke:* $\mathsf{Dec}(\mathsf{st}^{\rho^l}_{\mathsf{id}^l}, ctxt^l) \xrightarrow{\$} (\mathsf{st}^{\rho^l}_{\mathsf{id}^l}, \mathsf{id}_*, \mathsf{ctrl}^l_*, m^l_*, \mathsf{eid}^l_*, \mathsf{mid}^l_*)$
  - *if* $\mathsf{ctrl}^l_* = \perp$*, update:*

$$\mathrm{AT}^{\mathrm{d}}_{\mathsf{id}_*,\mathsf{eid}^l_*}[\mathsf{id}_*] \leftarrow \mathrm{AT}^{\mathrm{d}}_{\mathsf{id}_*,\mathsf{eid}^l_*}[\mathsf{id}_*] \,\|\, (\mathsf{id}_*, \mathsf{mid}^l_*, m^l_*, ctxt^l)$$

  - *else if* $\mathsf{ctrl}^l_* \neq \perp$*, update:*

$$\mathrm{CT}^{\mathrm{c,d}}_{\mathsf{id}^l,\mathsf{eid}^l_*}[\mathsf{id}_*] \leftarrow \mathrm{CT}^{\mathrm{c,d}}_{\mathsf{id}^l,\mathsf{eid}^l_*}[\mathsf{id}_*] \,\|\, (\mathsf{eid}^l_*, \mathsf{ctrl}^l_*, m^l_*, ctxt^l)$$

If $m^l_* \neq \perp$*, then encrypted and decrypted messages, control messages, and epoch and message counters outputs equal* $m^l_* = m^l_o, \mathsf{ctrl}^l_* = \mathsf{ctrl}^l_o, \mathsf{eid}^l_* = \mathsf{eid}^l_o, \mathsf{mid}^l_* = \mathsf{mid}^l_o$*, that epoch counter outputs increase monotonically* $(\forall l^* < l$ *with* $op^l = op^{l^*} = e$ *and* $\rho^{l^*} = \rho^l$ *it holds that* $\mathsf{eid}^{l^*} \leq \mathsf{eid}^l_o)$*, and the following transcript invariants hold:*

1. $\mathrm{CT}^{\mathrm{c,d}}_{\mathsf{id},\mathsf{eid}}[\mathsf{id}'][1:] \subseteq \{\mathrm{CT}^{\mathrm{c,e}}_{\mathsf{id}',\mathsf{eid},\mathsf{mid}} \cup \mathrm{CT}^{\mathrm{o,e}}_{\mathsf{id}',\mathsf{eid}}\} \wedge$
2. $\mathrm{AT}^{\mathrm{d}}_{\mathsf{id},\mathsf{eid}}[\mathsf{id}'] \subseteq \mathrm{AT}^{\mathrm{e}}_{\mathsf{id}',\mathsf{eid}}$

The gACCE model analyzes QUIC's security properties in various epochs. fACCE captures certain security properties which are given values corresponding to a stage of the protocol that they are guaranteed thereafter: Authentication and integrity, key compromise impersonation (KCI) resistance, forward secrecy, resistance against replay attacks, and reveals

against executions' random coins. MLS already provides guarantees for FS and replay attacks which we do not model for brevity. For gACCE, we add PCS as an additional security property denoted by the counter `pcs` to the ten presented in [187]. Similar to the `fs` counter, `pcs` defines the epoch from which post-compromise security (with respect to the group session) is reached. It should be hard, for an epoch $\mathsf{eid} \geq \texttt{pcs}$, to break the confidentiality of ciphertexts, even if all group members were corrupted previously (*unless* one of the members' random coins were revealed to the adversary). We modify the freshness notion to incorporate PCS security in Figure 7.5.

$\underline{\text{Fresh}_{\texttt{fs-pcs}}}$

1: **for** $(i, j) \in [n_p]$ **do**
2: **for** $(s, t) \in [n_s]$: **do**
3: $\texttt{ctr} \leftarrow \min(\mathsf{eid}^* : \pi_i^s.fr_{\mathsf{eid}^*} = 1)$
4: **if** $corr_i = 1$ **then**
5: $\texttt{ctr} \leftarrow \max(\texttt{ctr}, \texttt{fs})$
6: $\tau \leftarrow \pi_i^s.\mathsf{eid}$
7: $\pi_i^s.fr_{\mathsf{eid}^*} \leftarrow 0$ for all $\mathsf{eid}^* < \texttt{ctr}$
8: **if** $(\mathsf{Upd} \in \{\pi_i^s.\mathrm{CT}^{\mathrm{c,e}}_{i,\tau}, \cup\, \pi_i^s.\mathrm{CT}^{\mathrm{o,e}}_{i,\tau'}\}) \wedge ((\pi_j^t.\mathrm{CT}^{\mathrm{c,d}}_{j,\tau'}[i] = \pi_i^s.\mathrm{CT}^{\mathrm{c,e}}_{i,\tau'})$
$\vee\, (\pi_j^t.\mathrm{CT}^{\mathrm{c,d}}_{j,\tau'}[i] = \pi_i^s.\mathrm{CT}^{\mathrm{o,e}}_{i,\tau'})) \;\forall\; \tau' > \tau$ **then**
9: $\pi_i^s.fr_{\mathsf{eid}^*}, \pi_j^t.fr_{\mathsf{eid}^*} \leftarrow 1$ for all $\mathsf{eid}^* \geq \tau'$
10: $\texttt{pcs} \leftarrow \max(\texttt{pcs}, \tau')$

Figure 7.5. Freshness notion for fACCE with PCS modifications in blue. Freshness (fr) is regained after corruption through all parties reaching a new shared group state resulting from processing an update. We abuse notation on line 8 in the second clause to convey that the *commit* to π_i^s's update must appear in π_j^t's transcript (i.e. $(\mathsf{Commit}, \mathsf{Upd}) \in \mathrm{CT}^{\mathrm{c,d}}_{j,\tau+1}[i]$).

The execution environment of fACCE is also suitable to analyze QUIC. Particularly, it's worth noting that the Honest Partner definition can extend to the group channel formed by QUIC-MLS.

Definition 19 (Honest Group Partner (adapted from [187]))**.** *π_j^t is an honest group partner of π_i^s if $(i, j) \in \overrightarrow{\mathsf{G}}$ where $j \neq i$ and $\overrightarrow{\mathsf{G}} = \{i, \ldots, n\}$ are group partners and all initial variables match (i.e. $\pi_i^s.\overrightarrow{\mathsf{G}}_{\mathsf{eid}} = \pi_j^t.\overrightarrow{\mathsf{G}}_{\mathsf{eid}}$, $\pi_i^s.\mathsf{st}.\gamma = \pi_j^t.\mathsf{st}.\gamma \neq \perp$, $\pi_i^s.\mathsf{eid} = \pi_j^t.\mathsf{eid}$ where $\rho_\iota = \mathsf{a}$ for some $\iota \in \overrightarrow{\mathsf{G}}$), and the received control transcript is a prefix of the partner's sent control transcript for their common epochs, respectively, where at least one of them is not empty (i.e., for any epoch n s.t. $0 \leq n \leq m$ where m is the $\min(\pi_i^s.\mathsf{eid}, \pi_j^t.\mathsf{eid})$, where $a = \left|\pi_j^t.\mathrm{CT}^{\mathrm{c,d}}_{i,n}[j]\right|, b = \left|\pi_i^s.\mathrm{CT}^{\mathrm{c,d}}_{j,n}[i]\right|$ such that if $a > 0$ and $b > 0$ then $\forall\ 0 \leq \alpha < a$:*

$(\pi_i^s.\mathrm{CT}_{i,n}^{\mathrm{c,e}}[\alpha] = \pi_j^t.\mathrm{CT}_{j,n}^{\mathrm{c,d}}[i][\alpha])$ *and* $\forall 0 \leq \beta < b : (\pi_i^s.\mathrm{CT}_{i,\mathsf{eid}}^{\mathrm{c,d}}[j][\beta] = \pi_j^t.\mathrm{CT}_{j,n}^{\mathrm{o,e}}[\beta])$). *If* π_i^s *already received ciphertexts from any* π_j^t*, then* π_j^t *is an honest group partner of* π_i^s *only if no other honest partner* π^* *exists outside the set* $\pi_{s.G}^i$.

Our honest group partner notion changes the honest partnering from fACCE in two major ways. First, gACCE allows for partnering to be a 1-to-many relationship such that there can be up to $n-1$ honest partners for a particular initiator after decrypting a ciphertext. This reflects the fact that any member of an MLS group can encrypt and decrypt application messages and group control messages. Second, as opposed to matching conversations across all message types, we only require matching control message sequences within CT between honest group partners to enforce group state agreement.

7.6.2 Adversarial Model

We adopt the adversarial model of fACCE in [187] for our security experiment with modifications for the group setting. Like fACCE, an adversary $\mathcal{A}$ plays a security game with a challenger in which they attempt break the confidentiality (i.e. via guessing a sampled bit b) or integrity (i.e. via forgery) of messages in the gACCE protocol. At the beginning of the experiment, the challenger runs KGen to generate keys for all participants of the protocol and gives $\mathcal{A}$ the list of their public keys. $\mathcal{A}$ is given oracle access to OInit, OEnc, ODec, OReveal, and OCorrupt and can control communications between participants. One way the $\mathcal{A}$ can win the security game by is by triggering a $\texttt{win} \leftarrow 1$ (indicating breaking the authenticity or integrity of ciphertexts). Otherwise, at the end of the experiment $\mathcal{A}$ terminates and outputs a bit b' against the challenge bit $\pi_i^s.b_{\mathsf{eid}}$. The challenger checks the freshness conditions and if $\mathcal{A}$ has not violated any of the freshness conditions outputs $(b' = \pi_i^s.b_{\mathsf{eid}}) \vee \texttt{win}$. We detail the modifications to the relevant oracles below:

- $\mathsf{OInit}(i, s, \overrightarrow{\mathsf{G}}, \rho)$ initializes a session π_i^s (if not yet initialized) of party i to be in a group with parties in the list $\overrightarrow{\mathsf{G}} = (i, \ldots, n)$, invoking $\mathsf{gACCE.KGen}() \xrightarrow{\$} (\mathrm{sk}_n, \mathrm{pk}_n)$ for each party in $\overrightarrow{\mathsf{G}}$ (if not yet generated) and then $\mathsf{gACCE.Init}(sk_i, \overrightarrow{\mathsf{PK}}_{\overrightarrow{\mathsf{G}}}, \rho, ad) \xrightarrow{\$} \pi_i^s.st$ under $\$ \leftarrow \pi_i^s.rand$. It also sets $\pi_i^s.\rho \leftarrow \rho$ and $\pi_i^s.\overrightarrow{\mathsf{G}} \leftarrow \overrightarrow{\mathsf{G}}$. This oracle provides no return value. All subsequent invocations of Enc, Dec of this session participant use $\pi_i^s.rand$ for obtaining randomness. Finally freshness flags are updated by invoking $\mathrm{Fresh}_{\texttt{fs-pcs}}$ (see Figure 7.5).

- $\mathsf{OEnc}(i, s, \mathsf{ctrl}, m_0, m_1; \pi_i^s.rand)$ Upon satisfaction of freshness conditions, this triggers either the encryption of a message or a group action proposal according to the given ctrl instruction for the encryption of message m_b for $b = \pi_i^s.b_{\mathsf{eid}}$ by invoking $\mathsf{gACCE.Enc}(\pi_i^s.st, \mathsf{ctrl}, m_b) \rightarrow (st', c, \mathsf{eid}, \mathsf{mid})$ for an initialized π_i^s. It returns $(c, \mathsf{eid}, \mathsf{mid})$ to the adversary. If $m_0 \neq m_1$, the challenger checks the freshness conditions (given by Figure 7.5) and outputs $\perp$ if the $\mathcal{A}$ issued this query and $\pi_i^s.fr_{\mathsf{eid}} = 0$. Otherwise, if encryption succeeds (i.e. $c \neq \perp$), this oracle updates the session specific variables $\pi_i^s.st \leftarrow st'$, and appends the private results to the challenger held encryption CT and AT in π_i^s according to Definition 18.
- $\mathsf{ODec}(i, s, c; \pi_i^s.rand)$ triggers invocation of $\mathsf{gACCE.Dec}(\pi_i^s.st, c) \rightarrow (st', m, \mathsf{ctrl}, \mathsf{id}, \mathsf{eid}, \mathsf{mid})$ for initialized π_i^s and returns $(m, \mathsf{eid}, \mathsf{mid})$ to the adversary. If c was not generated by an honest partner and the epoch was fresh according to Definition 19 and Figure 7.5, respectively, then win = 1 and the challenger ends the game. Otherwise, if decryption was successful (i.e. $m \neq \perp$), then it appends the results to the decryption CT and AT in π_i^s according to Definition 18.

We define the (insecurity) of gACCE below:

Definition 20 (Breaking gACCE Security)**.** *Let* Π *be a flexible group ACCE protocol* $\Pi = \{\mathsf{KGen}, \mathsf{Init}, \mathsf{Enc}, \mathsf{Dec}\}$. *Let* $\mathsf{Exp}^{\mathsf{gACCE}}_{\Pi,\mathcal{A}}(\lambda)$ *be the* gACCE *security experiment described in Section 7.6.2 in which an adversary* $\mathcal{A}$ *breaks the security of* gACCE *security when* $\mathcal{A}$ *terminates and outputs* (i, s, eid, b')*, if there either exists a session* π_i^s *such that* $\pi_i^s.b_{\mathsf{eid}} = b'$ *and* $\pi_i^s.fr_{\mathsf{eid}} = 1$*, or* win = 1. *We define the advantage of an adversary* $\mathcal{A}$*that breaks a group flexible ACCE protocol* gACCE *as:* $\mathsf{Adv}^{\mathsf{gACCE}}_{\Pi,\mathcal{A}}(\lambda) = |2{*}\mathsf{Pr}(\mathsf{Exp}^{\mathsf{gACCE}}_{\Pi,\mathcal{A}}(\lambda) = 1) - 1| + \mathsf{Pr}[\mathtt{win} = 1]$.

7.6.3 Security Proof Outline

Broadly speaking, our construction is simply a composition of a CGKA protocol, Authenticated Encryption (AuthEnc) protocol, and some book-keeping functionalities. There is no change to either protocols nor their primitives. Thus, the game-hopping based proof is split into two cases which are structurally similar:

Case 1 The adversary $\mathcal{A}$ attempts to forge a message. That is, $\mathcal{A}$ succeeds in getting a party to decrypt a ciphertext c for which no honest partner produced.

Case 2 The adversary $\mathcal{A}$ guesses the correct bit b with non-negligible probability.

a) Case 1: We proceed in a series of game-hops as follows:

Game 0 (G0) This is the gACCE experiment as described in Section 7.6.2 with no modifications.

Game 1 (G1) This version is the same as G0 except that we replace all of the keys used for encryption or decryption during an epoch marked fresh by in the gACCE experiment (i.e., $\mathcal{A}$has not triggered a trivial attack) with uniformly random values from a CGKA challenger C_{CGKA}. This is sound under the standard hybrid argument since CGKA provides uniformly random keys for fresh epochs. Different CGKA challengers are extended, without loss of generality (WLOG), to any of group sessions allowed under the gACCE experiment. For each group session, a CGKA reduction is made as follows. Freshness in the gACCE game then corresponds to freshness in the CGKA game. When $\mathcal{A}$ issues a OReveal query, the challenger first checks the freshness flag of the epoch and if $fr = 1$, it returns a uniformly random value from C_{CGKA}, otherwise it returns the real key. When $\mathcal{A}$ issues a OCorrupt query against a party for a given epoch, if the epoch in question is not fresh then the real long-term keys of party are exposed, otherwise the query is not allowed. Future corruptions in new epochs against that party are allowed until pcs flips fr back to 1.

Game 2 (G2) This game is the same as G1 except that we replace all instances of encryption and decryption in OEnc and ODec with ones provided by an AuthEnc challenger $\mathcal{B}_{\mathsf{AuthEnc}}$. WLOG, this requires one fresh AuthEnc reduction per epoch per group session (each group has distinct epochs and each epoch has new keys). In each instance, the adversary's advantage in forgery reduces to the AuthEnc.auth security of the AuthEnc scheme.

b) Case 2: This proof follows the series of game hops exactly as in Case 1 except that in G2, the adversary's advantage in guessing bit b reduces the advantage of an adversary guess bit b in the AuthEnc.conf game.

Theorem 5 (gACCE Security for QUIC-MLS)**.** *Let n_s, n_G, and n_e be the number of sessions, number of groups, and number of epochs given by some security parameter λ used in the experiment. The QUIC-MLS protocol presented in Figure 7.2 is* gACCE*-secure. That is, for any PPT algorithm $\mathcal{A}$ against the* gACCE *security experiment (described in Section 7.6.2), the $\mathsf{Adv}^{\mathsf{gACCE}}_{\Pi,\mathcal{A}}(\lambda)$ is negligible under the* cgka *security of* CGKA *and the* auth *and* conf

security of the AuthEnc *primitives.*

7.7 Conclusion

This work introduces the first formalization and analysis of QUIC-MLS, an idea initially proposed for space networking security. We demonstrate feasibility of the concept and provide a cryptographic security proof. With growing needs for low-latency and efficiency, as well as security, this points a way forward for using novel key exchange as a replacement for legacy handshake approaches within networking protocols.

Appendix

7.8 QUIC-MLS Protocol Construction

From a high level overview, QUIC-MLS establishes a secure group channel among two or more parties which uses the output of the MLS continuous key agreement component as the input to the QUIC record layer used to send encrypted application layer messages. We detail our assumptions and intuition on different components of the protocol below:

- *Operational Settings* We assume that there is no centralized strongly consistent DS (e.g. server style DS). Rather, we rely on QUIC transport mechanisms to provide reliable in-order delivery of all messages. Furthermore, we assume that QUIC-MLS participants already have the certificate information (i.e. public keys and associated identities) of all other participants as provided by some AS implementation (e.g. PKI from CAs, out-of-band, preinstalled, etc.).
- *Initialization* The initiator of a session may choose one or more participants to form a group. The initiator uses the public keys of the group member(s) to start an MLS session by unilaterally committing their own add proposal and sends individualized Welcome messages out (via their public keys).
- *Group Control* Because we assume there is no DS to arbitrate valid proposals simultaneously sent by multiple group members, we assign a leader to each epoch that is allowed to commit to group control proposals to ensure no conflicts can arise. Leader assignment can be time based, schedule based, or arbitrarily selected to fit a use-case. For example, epoch leaders can be chosen based on a pre-assigned schedule upon group initialization which is exported (i.e. in the `Group Context`) and updated with any group control action. Or, perhaps more elegantly, epoch leaders can be assigned based on their position in the ratchet tree (leftmost to right).
- *Encryption and Decryption* We make no changes to the QUIC AEAD Encryption or Decryption as previously discussed in Section 7.4.3. However, if non-repudiation between group members is desired for application messages, then the use of a sign-then-encrypt procedure can be adapted to messages input to QUIC's AEAD.

Term	Description
$\mathrm{AT}^{x}_{\mathsf{id},\mathsf{eid}}[\mathsf{id}']$	The application message transcript used in the gACCE security model where $x \in (e, d)$. Note, $\mathrm{AT}^{e}_{\mathsf{id},\mathsf{eid}}$ does not maintain separate vectors for partners (i.e., $\mathrm{AT}^{e}_{\mathsf{id},\mathsf{eid}} = \mathrm{AT}^{e}_{\mathsf{id},\mathsf{eid}}[\mathsf{id}]$).
$\mathrm{CT}^{a,b}_{\mathsf{id},\mathsf{eid}}[\mathsf{id}']$	The group control message transcript used in the gACCE security model where $a \in \{c, o\}$ signifies confirmed or opportunistic, $b \in \{e, d\}$ refers to encrypted or decrypted, id refers to the transcript owner and id' refers to a partner id, and eid refers to the epoch number. Similarly, id does not maintain encryption transcripts of partners (i.e., $\mathrm{CT}^{\cdot,e}_{\mathsf{id},\mathsf{eid}} = \mathrm{CT}^{\cdot,e}_{\mathsf{id},\mathsf{eid}}[\mathsf{id}]$).
st	The gACCE state stored in memory of each group member which includes the following vectors: γ - The CGKA group state as defined in 16 ρ - The role of the st owner which can take the values of $\{\mathrm{a}, \mathrm{p}\}$ for admin or participant. Only admin can commit to proposals sent by other group members. ast - The administrative state which holds a transcript of commits and proposals, and (optional) welcome messages leading to the current epoch formatted as: [commit, $\overrightarrow{\mathrm{p}}$, welc] or a proposed epoch (where commit = $\perp$). eid - The epoch number associated with a particular id: st.eid is shorthand for $\mathsf{st.eid}_{\mathsf{st.id}}$. gid - The group identifier of the current session id - The ID of the st owner kst - The set of keys stored by the st owner indexed by eid. $\mathsf{mid}_{\mathsf{eid}}$ - The message transcript indexed by eid $\overrightarrow{\mathsf{G}}_{\mathsf{eid}}$ - The IDs of group members in a particular epoch

Table 7.3. Summary of variable terminology

Part IV

Design and Implementation for Post-Quantum Optimization

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 8:
Benchmarking of the Amortized Post Quantum Combiner for MLS

Addressing the quantum vulnerability of space systems

As established in Chapter 1, traditional space security architectures face a third fundamental inadequacy: the impending obsolescence of classical cryptography against quantum adversaries, coupled with the prohibitive overhead of naive post-quantum replacements. Satellites launched today may operate for decades into the quantum era, handling information that must remain classified for extended periods. With no upgrade path post-launch and adversaries already harvesting encrypted traffic for future decryption once quantum computers become available (the "harvest now, decrypt later" (HNDL) threat) space systems require quantum-resistant cryptography now, not later. Furthermore, quantum adversaries could forge authentication, impersonating legitimate communications and commanding spacecraft or disrupting operations. However, post-quantum algorithms impose substantial costs: public keys and ciphertexts are significantly larger (often an order of magnitude) than their classical counterparts, and encryption/decryption operations demand more computation. For bandwidth-constrained and power-limited space systems where every byte of bandwidth is precious and power budgets are tightly constrained, these overheads make naive post-quantum deployment prohibitively expensive—creating an apparent dilemma between quantum security and operational feasibility.

Pillar III: Practical Post-Quantum Deployment Through Amortization. Recall from Chapter 1 that Pillar III resolves this dilemma by developing practical mechanisms for deploying post-quantum cryptography without sacrificing operational efficiency. Rather than paying the full cost of quantum-resistant key exchanges at every update, the Amortized Post-Quantum MLS Combiner spreads quantum-resistant entropy across multiple updates over time. The key insight is that MLS's asynchronous ratcheted key schedule, the same mechanism that enables the dynamic key agreement in Pillar II, allows quantum-resistant key material to be injected periodically (via full updates) and then combined with faster classical key agreement operations in intermediate (partial) updates. This amortization

approach maintains quantum resistance while achieving bandwidth and computational costs approaching those of classical-only implementations. Importantly, the mechanism provides tunable security-efficiency tradeoffs: system designers can adjust the frequency of full post-quantum updates based on their threat model, operational constraints, and desired security margin. The amortization approach is specifically designed to exploit the asynchronous updates underlying the QUIC-MLS and BPSec-MLS integrations from Pillar II, ensuring that the operational flexibility gained through dynamic key agreement is not lost when transitioning to quantum-resistant cryptography.

Chapter Contribution and Impact. This chapter validates the amortization approach through comprehensive implementation and empirical benchmarking of the Amortized Post-Quantum MLS (APQ-MLS) Combiner. This work demonstrates that quantum resistance is within reach of space systems launched today and can be deployed at costs roughly equivalent to traditional cryptography—resolving the apparent tension between quantum security and operational feasibility that has hindered post-quantum adoption in constrained environments.

At the highest level, this work transforms post-quantum cryptography from a future aspiration into a present reality for space systems. By reducing the cumulative overhead of quantum-resistant operations to levels approaching classical cryptography, APQ-MLS removes the primary barrier preventing space agencies and commercial operators from deploying quantum-resistant security today. Satellites using BPSec-MLS or QUIC-MLS (from Pillar II) can perform full post-quantum updates during high-bandwidth contact windows and rely on lightweight partial updates during constrained periods, maintaining continuous quantum resistance throughout their operational lifetimes without exhausting bandwidth or power budgets. For multi-decade missions—deep space probes, lunar infrastructure, geosynchronous satellites—this capability ensures that communications remain secure even after cryptographically-relevant quantum computers arrive. Furthermore, the amortization mechanism is generalizable beyond the specific post-quantum algorithms currently standardized by NIST: as cryptanalysis advances and algorithms become more efficient, or as new vulnerabilities are discovered in current ciphersuites, the APQ-MLS framework can accommodate these changes through ciphersuite modifications rather than requiring wholesale protocol redesign. This future-proofing ensures that the protocol infrastructure established by Pillars I and II remains viable across the evolving post-quantum landscape.

At the protocol level, this chapter implements the APQ-MLS combiner to measure the effectiveness of driving down the cost of post-quantum operations in MLS through amortization. The combiner operates in two modes: PQ-Confidentiality mode (injecting quantum-resistant entropy for forward secrecy and post-compromise security) and PQ-Confidentiality+Authentication mode (additionally providing quantum-resistant authentication guarantees). Testing confirms that the amortized approach reduces cumulative bandwidth consumption by up to 94% and cumulative computational overhead up to 85% compared to standard post-quantum MLS implementations that perform full post-quantum operations at every update. Moreover, when compared against the leading hybrid post-quantum construction X-Wing, widely considered the state-of-the-art for combining classical and quantum-resistant cryptography in a hybrid cipehrsuite, the APQ-MLS Combiner outperforms it by over five times in cumulative bandwidth while providing even stronger security guarantees (specifically, post-quantum authentication that X-Wing does not provide). These results demonstrate that quantum-resistant cryptography is not merely feasible for space systems but can be deployed at costs roughly equivalent to traditional cryptography, eliminating the tradeoff between security and efficiency.

At the implementation level, this work provides comprehensive empirical validation across multiple dimensions. The implementation incorporates native Rust post-quantum cryptography libraries (leveraging memory-safe implementations of NIST-standardized algorithms including ML-KEM and ML-DSA), tunes the OpenMLS library for performance in constrained environments, and expands benchmarking capabilities to measure cumulative overhead across extended group lifetimes rather than focusing solely on per-operation costs. Specific metrics measured include: bandwidth consumption for MLS control messages over time, computational overhead for key derivation and cryptographic operations, and latency for processing group updates. The experiments systematically and exhaustively vary ciphersuites and amortization parameters (frequency of full post-quantum updates) to characterize the security-efficiency tradeoff space, enabling mission designers to select configurations appropriate for their specific operational constraints and threat models. Ciphersuite downselection identifies optimal combinations of classical and post-quantum algorithms that balance performance with security, informed by NIST standardization outcomes and cryptanalytic assessments.

Seamless Transition to Full Post-Quantum. As a strictly non-hybrid approach—running

two parallel MLS sessions, one classical and one post-quantum, and combining their outputs—the APQ-MLS Combiner makes transitioning to full post-quantum cryptography trivial, as posited by the National Security Agency's (NSA) Commercial National Security Algorithms (CNSA) 2.0 roadmap [1]. Rather than reinstantiating a new session with new post-quantum ciphersuites (a disruptive operation requiring all participants to renegotiate security associations), deployed systems using the combiner can transition to full-PQ without interruption by simply terminating and shedding the classical session half and leaving only the post-quantum MLS session. This graceful transition path allows space systems launched today with the APQ-MLS Combiner to seamlessly upgrade to full post-quantum as CNSA 2.0 mandates take effect, without requiring mission replanning, software patches, or service disruptions.

Real-World Impact and Standardization. The draft specification associated with this work [8] has been adopted by the IETF MLS working group and is currently in working group last call for standardization - the final review stage before publication as an RFC. This positions APQ-MLS to become part of the standardized internet protocol suite, ensuring that the amortization mechanisms developed for space use cases benefit the broader internet community. The work has been implemented and tested by PhoenixR&D in independent implementations [188], with commercial entities intending to deploy APQ-MLS in production systems. Their feedback from testing has been incorporated into the newest draft version, validating that the protocol is implementable and performs as specified. Furthermore, the US Navy's Program Executive Office for Command, Control, Communications, Computers and Intelligence (PEO C4I) will be testing APQ-MLS in the upcoming Rim of the Pacific Exercises (RIMPAC), providing operational validation in realistic military communication scenarios. This paper has been accepted to the Security Standardization Conference (SSR) 2025, where it will be presented alongside QUIC-MLS, showcasing the integrated framework spanning Pillars I, II, and III. This work was funded by the generous support of the National Reconnaissance Office, which designs and manages collection satellites for the U.S. government, underscoring the operational relevance and national security importance of practical post-quantum cryptography for space systems.

Joint Work and Contributions. This chapter presents the implementation and benchmarking of the Amortized Post-Quantum MLS (APQ-MLS) Combiner which was joint work with Britta Hale (Naval Postgraduate School) and Lee Wang (Defense Language Institute).

It has been reprinted, with permission, from all authors.[21] The candidate contributions include writing, experimental design, incorporation of native rust PQ crypto libraries, and down-selection of ciphersuites. Lee Wang and the candidate worked jointly in general code implementation including tuning OpenMLS, updating the codebase inherited from Noah Greene from prior work, and expanding benchmarking capabilities. Editing the paper and performing data analysis was joint work by all authors.

Chapter Abstract

Overhead costs associated with post quantum (PQ) algorithms, especially digital signatures, create a significant barrier to incorporation and adoption of post quantum cryptographic protocols in various settings. To counter this, the working group for the Messaging Layer Security (MLS) protocol under the Internet Engineering Task Force has proposed an approach where traditional and PQ sessions of the protocol are strategically combined in such a way as to amortize PQ-associated overhead, i.e., an Amortized Post Quantum (APQ) combiner. In this work, we implement and benchmark APQ using standardized NIST algorithms (ML-KEM and ML-DSA) integrated into OpenMLS with native Rust cryptographic libraries, presenting the first comprehensive performance evaluation of APQ to include PQ authenticity. Our evaluation encompasses execution run-time, message size, and memory consumption across various security levels and amortization ratios to compare and contrast MLS with traditional-only, APQ confidentiality-only, APQ confidentiality+authenticity, and an alternative hybrid ciphersuite. We demonstrate that APQ achieves exponential improvements in message size and memory efficiency as amortization traditional:PQ ratios decrease from 1:1 to 1:100, with optimal performance observed around 1:50 ratios. These findings establish APQ as a practical solution for deploying post quantum security in resource constrained settings.

8.1 Introduction

The rapid advancements in quantum computing pose an imminent threat to the cryptographic foundations that secure modern digital communications. While practical quantum computers

[21] 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. This paper has been accepted for publication at SSR 2025.

capable of executing Shor's Algorithm at scale remain under development, the cryptography community has proactively developed post-quantum (PQ) cryptography, which is based on alternative, quantum-resistant hardness assumptions. Standardized PQ algorithms, under National Institute of Standards and Technology (NIST), exist [5] and integration into various protocols is ongoing [189]–[191]. Among these, early efforts in PQ migration have been focused on *hybrid* approaches, which aim to attain PQ security while integrating PQ and traditional asymmetric cryptographic algorithms. Because PQ algorithms are relatively nascent and fledging, the hybrid approach allows adopters to maintain classical security even if the PQ algorithm is broken, e.g., [192], [193].

Notably, the incorporation PQ cryptographic algorithms is not without challenges. Post-quantum key encapsulation and signature operations incur significantly higher computational and bandwidth costs compared to their classical counterparts. For example, according to prior work [194] on TLS, Dilithium (ML-DSA) signature sizes are approximately forty times larger (48B vs 2044B) than ECDSA with private key sizes nearly sixty times larger (48B vs 2800B) than ECDSA. For KEMs, slowdown factors of 1.05 to 2.55 are also to be anticipated in TLS [195] when using PQ KEMs versus ECDHE and ciphertext sizes can be eleven times larger (64B for ECDH vs 736B for ML-KEM512) [196]. These increased costs can be prohibitive for resource-constrained devices with limited compute, power, or bandwidth, creating a challenge even under the necessity for PQ. Use of hybrid approaches further exacerbates this overhead.

Recognizing the different risk tolerance levels of adopters, the Internet Engineering Task Force (IETF), under the Messaging Layer Security (MLS) Working Group, has considered two distinct approaches for integrating quantum resistance, both providing variant hybrid guarantees: 1) use of direct hybrid ciphersuites and 2) strategic integration of traditional and PQ sessions to *amortize* the PQ overhead. The first approach follows a typical hybrid process, namely the combination of traditional and PQ key encapsulation mechanisms (KEMs) for every key update [118]. Meanwhile, the second approach combines two parallel sessions, one with PQ ciphersuites and one with traditional ciphersuites, where randomness from the PQ session is injected in the traditional session at controllable intervals. This is called the Amortized Post Quantum (APQ) combiner method [119]. The APQ approach aims to reduce the overall frequency of PQ operations, and therefore bandwidth overhead, while offering PQ confidentiality and optionally PQ authenticity as well. While much work has looked at

the performance of PQ protocols using hybrid ciphersuites such as in (1), in this work, we provide the first comprehensive benchmarking of performance across amortized alternatives, comparing various ciphersuites, modes of operation for the APQ combiner (including PQ confidentiality-only and PQ confidentiality+authenticity), and show comparison vs. other normalized hybrid approaches (e.g., X-Wing [197]) and simple PQ ciphersuites. We cross-compare APQ for amortization ratios of traditional:PQ from 1:1 to 1:100 with metrics for message size, memory efficiency, and time. This work demonstrates the feasibility of achieving not only PQ-confidentiality but also PQ authenticity even on resource constrained devices.

Contributions This work provides a comprehensive benchmarking of the APQ combiner for MLS from [119] using NIST-standardized algorithms. Our contributions include:

1. **Amortized Post Quantum Combiner Analysis:** We conduct benchmarking of the APQ combiner for time (in seconds), random access memory (RAM) usage (in bytes), and message output size (in bytes) in its two modes (PQ Confidentiality-Only and PQ Confidentiality+Authenticity) and provide analysis of amortization strategies overall protocol performance.
2. **Amortized Combiner vs Hybrid Combiner** We provide the first benchmarking of the APQ combiner in its two modes against the hybrid KEM combiner, X-Wing.

The remainder of this paper is organized as follows. Section 2 provides background on the MLS protocol and related work, hybridization approaches in MLS, and details on the APQ combiner. Section 3 details the implementation approach, ciphersuites used, and metrics considered. Section 4 presents performance results for time, memory usage, and message size. Finally, Section 5 concludes the paper with implications for PQ MLS deployment.

8.2 The Messaging Layer Security Protocol, Hybrids, and the APQ Combiner

Messaging Layer Security MLS provides end-to-end encryption to communicating parties. It is a type of Continuous Key Agreement (CKA), which also encompasses Signal [113] and other ratcheted protocols. Unlike its predecessors, MLS is built to be extensible to groups of larger sizes than typical 1:1 channels while also achieving lower scaling overhead. While we provide a general introduction to MLS here, inclusive of the group scenario, the functional

design of the protocol provides scaling performance improvements relative to the number of participants. Consequently, our testing focuses on the simple two-party case to demonstrate the overhead cost-savings of the flexible hybrid combiner even under the worst case scenario.

CKAs "ratchet" or update keys throughout the lifetime of communications. In MLS, this is facilitated by a subfunctionality called TreeKEM [125] which manages keys as a binary tree that is updated via a series of KEM operations. The root of the tree is used to compute the shared secret from which data encryption keys are derived. To evolve the state (or add/remove communication parties), members update keys through a propose-and-commit sequence. Proposals, which can be made by any member, are suggestions to modify the ratchet tree through adding, removing, or updating of nodes. A commit message is sent by a group member to ratify the proposal(s) and enter a new *epoch*. An *empty* commit, is when a member updates their own representative binary tree leaf node and unilaterally commits to it. All proposals and commits are authenticated through the use of digital signature algorithm (DSA) operations.

As a result of the CKA construct and epochal key evolution, the protocol achieves forward secrecy (FS) and post compromise security (PCS). FS ensures past communications remain secure upon compromise of security keys while PCS allows security to be regained after a compromise using fresh keys. Crucial for APQ, entropy can be added to or exported from the cryptographic state via pre-shared keys (PSK). When evolving the state, a proposal can include a PSK identifier, indicating that it should be included into the next commit. Thus, a PSK can be added to the key derivation function (KDF) along with the new shared secret to derive keys. PSKs can be derived to be exported from the shared secret as an *exporter key*. The PSK functionality and optional inclusion in proposals and commits will become useful in the APQ combiner mechanism, described later.

A variety of MLS implementations exist across major programming languages [120]. Relevant to this work, this includes a Rust version called OpenMLS that offers memory safety features and integrates with widely available PQ libraries. We use OpenMLS due to its clear and thorough documentation [198].

MLS Hybrids X-Wing has been proposed to the IETF as a hybrid KEM combiner [199]. It provides indcca security based on the indcca of its component KEMs, MLKEM and X25519. The IETF MLS working group has adopted X-Wing as a post-quantum secu-

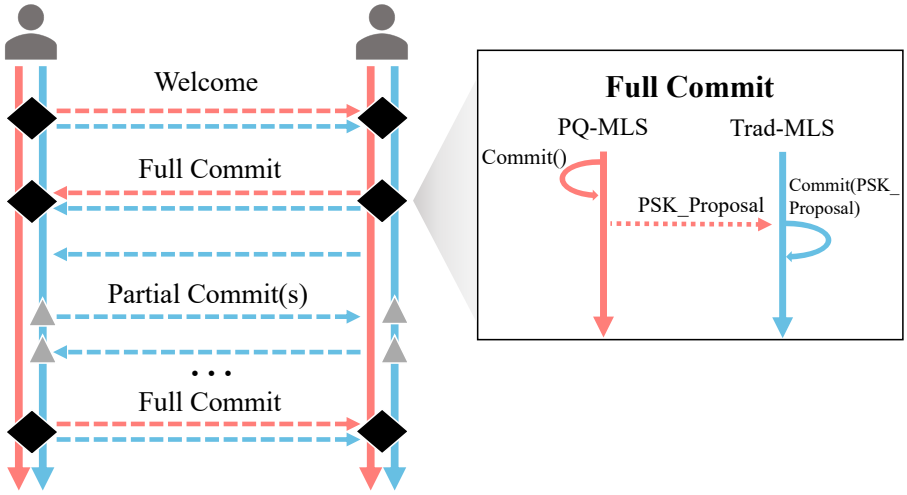


Figure 8.1. Overview of the APQ protocol using PQ and traditional MLS sessions (solid lines). Left user adds right user via Welcome messages sent in both sessions (thick dashed lines). ♦ underscores the state changes in both sessions via Full Commits while ▲ highlights state change in just one session via Partial Commits. MLS Welcome and Commit mechanics are detailed in [3]. APQ PSK_Proposal generation is defined in [119].

rity option among the set of new PQ ciphersuites in [200]. As an example of a typical hybridization approach, we include X-Wing for cross-comparison with the APQ combiner.

APQ Combiner The APQ combiner [119], illustrated in Figure 8.1, combines two MLS sessions, one with PQ algorithms, A, and the other with traditional algorithms, B. These two sessions run in parallel, with the same set of communication participants.

The traditional session B operates according to normal MLS functionality: updates to the shared keying material are proposed and *committed*, effectively ratcheting the state forward. These are performed using a traditional KEM (e.g., DHKEM), and signed with a traditional signature. We call such traditional-only commits *Partial Commits*, and the overall key update performed a *Partial Update*. The new state is used to derive data encryption keys. Application messages are then encrypted with one of a selection of Authenticated Encryption with Associated Data (AEAD) schemes, and also signed to ensure uniqueness and non-repudiation inside the session.

The PQ session A operates in two possible ways, dependent on the combiner mode selected,

PQ/T Confidentiality-Only or PQ/T Confidentiality+Authenticity:

- **PQ/T Confidentiality Only Mode:** Within the PQ session, updates to the shared keying material are performed using PQ KEM algorithm but signed with a traditional signature algorithm. No application messages are sent in the PQ session. Rather, it is used for maintaining and updating a PQ key schedule.
  From A's PQ key schedule an exporter secret can be derived using an established label. The PQ PSK is then injected in B using a PSK proposal message that is committed to yield a new B state. As this mixes the PQ PSK into the traditional session B state using a KDF within the MLS key schedule, the resultant B key schedule is also PQ. Thus when data encryption keys are derived within B, they provide for PQ confidentiality. When a proposal-commit sequence in the traditional session B uses the PQ PSK in as described here, we call the overall commit a *Full Commit* and the overall key update a *Full Update*.
- **PQ/T Confidentiality+Authenticity Mode:** This mode follows that of the PQ/T Confidentiality-Only mode, with the exception that key updates in the PQ session are signed with a PQ digital signature algorithm.

Thus, session A uses PQ KEM in the PQ/T Confidentiality-Only (PQ-Conf) mode and a combination of PQ KEM and PQ DSA in the PQ/T Confidentiality+Authenticity (Conf+Auth) mode. Session B uses traditional KEM and DSAs in both modes.

Full Commits can be interspersed or scheduled on a less frequent basis than Partial Commits, thus spreading the associated overhead over a longer period, i.e., amortizing the cost of A operations. Note that Partial Commits update the keying state (which is already PQ, based on the last Full Commit). Thus, a quantum attack on the Partial Commit does not lead to leakage of the session secrets, but only reduces the relative window of PQ key update to that of a Full Commit, i.e., PQ PCS is determined by Full Commits. Meanwhile, PQ FS is still achieved under Partial Commits.

The overall security depends on the frequency of the Full Commits and the mode of operation. In the PT/T Confidentiality+Authenticity Mode, the fact that PQ signatures are used to sign key updates in A enforces PQ authenticity on the overall PQ/T key schedule, which is then used to derive AEAD keys, resulting in PQ authenticity on application data

sent in B.[22] The full APQ description and additional bookkeeping mechanisms for signaling Full or Partial updates (to curtail running the two sessions independently, e.g., using the APQMLSInfo object) can be found in [119].

Amortization is a key feature of the APQ combiner and is accomplished through interspersing *Full* key updates with one or more *Partial* key updates. In our testing, we denote amortization with ratios $i : j$ to refer to i number of Full Updates for every j number of Partial Updates.

Other Related Work Prior work provided a proof of concept implementation and initial metrics for the APQ combiner [201], focusing on the PQ Confidentiality-Only Mode across various group sizes. In contrast, we provide cross comparison with other hybrid options as well as accounting for efficiency under the combined PQ confidentiality and authentication (PQ-Conf-Auth) mode.

Simultaneous to our implementation is that of PheonixIM [188], which also implemented the amortized combiner from [119]. While that work focused on meeting the specifications of the proposed standard in the context of interoperability with OpenMLS, we provide formal benchmarking of performance.

8.3 Implementation

Our implementation, extending that of [201], is available at github-link.[23] It is written in Rust, utilizing and modifying open-source Rust crates OpenMLS, hpke, and RustCrypto. OpenMLS implements MLS as promulgated by RFC9240. The cryptographic operations used by MLS (encryption, signing, etc.) are incorporated in a modular fashion using *crypto-providers* like OpenMLS-RustCrypto and OpenMLS-LibcruxCrypto. Crypto-providers are modular interfaces of underlying cryptographic libraries that specify how those libraries interact with OpenMLS. We chose to use OpenMLS-RustCrypto because RustCrypto, as a cryptographic library, is more widely used and does not require more modern x86 and amd64 CPUs [198]. like the high-assurance LibcruxCrypto library. In total, we utilized three

[22] Digital signatures over the AEAD inside the B session are still traditional, meaning that there is no session-internal non-repudiation. An inside attacker that possesses a quantum computer and wishes to frame another session participant can forge the requisite traditional signature. This is, however, a niche threat model, and unrelated to the PQ authenticity provided against external quantum adversaries.

[23] Link removed for reviewing anonymity; will be added for the full version.

additional cryptographic libraries: the RustCrypto MLDSA crate, RustCrypto MLKEM, and the HPKE crate (which generally implements KEMs). Tests were completed on a MacBook Pro with M2 Max (ARM CPU/GPU Combo), 64GB RAM, and macOS Ventura.

While ML-DSA is incorporated via RustCrypto's MLDSA and OpenMLS traits, support for ML-KEM required further customization of the HPKE library used by RustCrypto. Following the structure of the existing KEM implementation in hpke, we created a ml-kem handler to interface with the generic RustCrypto ML-KEM crate.

...rtization strategies. Generally, by varying the frequency of full-commits, the cost of PQ operations can be spread across the lifespan of a group session. Concretely, we tested with full-commits in occurring every 10, 50, and 100 commits to gain a sense of amortization effects. Moreover these tests were conducted across the two modes operation, PQ Confidentiality-Only and PQ Confidentiality+Authentication. To establish a baseline of comparison, we measured the performance costs of standard MLS (i.e., individual session) with all levels and appropriate combinations of PQ and traditional ciphersuites as specified by [200] and [3], respectively. Finally, we measured the performance of an MLS session running the X-Wing hybrid KEM to provide a comparison against hybridization strategies. In those tests, we compare across Full:Partial commit ratios of 1:1, 1:2, 1:5, 1:10, 1:50, and 1:100 to provide more granular comparisons against X-Wing.

8.3.1 Ciphersuites Tested

We detail which ciphersuites were selected for testing and provide rationale for their selection in this section.

Notation

Table 8.1 shows the ciphersuites used in our tests. For readability purposes, we assign aliases to the ciphersuites. Each alias begins with a prefix letter denoting which ciphersuite category they belong to: T for traditional, PQ for post-quantum, and H for combinations thereof. Within the traditional category, the second prefix represents the underlying curves used (i.e., EC for P256 and P384 and Ed for Ed25519 and Ed448) and the third prefix is an abbreviation of the authenticated encryption algorithm (i.e., AES vs CHACHA20POLY1305). For readability, we give all of the ciphersuites suffixes based on the roughly associated security

level (i.e., **L**ow, **M**edium, **H**igh).

Rationale

We chose ciphersuites for benchmarking based on the existing list of supported traditional ciphersuites from [3] as well as the list of proposed PQ ciphersuites from [200]. Following the guidance from [119], we selected to implement 'pure PQ' ciphersuites out of the proposed ciphersuites from [200] which also included PQ and Traditional combined KEMs that redundant in the APQ context. Absent from [200] is support for the lowest security levels of ML-KEM and ML-DSA: ML-KEM512 and ML-DSA44, due to lack of demand. For completeness, we constructed additional ciphersuites (listed in orange in Table 8.1). As there were no corresponding PQ ciphersuite which uses P521, we rejected the traditional counterpart from testing.

For a fair comparison with X-Wing, we add traditional and PQ ciphersuites (also in orange) that correspond to the existing X-Wing ciphersuite, as supported by OpenMLS (T-Ed-Cha-L and PQ-C-X in Table 8.1). To test X-Wing against APQ in the PQ Confidentiality+Authenticity mode, we added an additional PQ ciphersuite with comparable security level components (PQ-CA-X in Table 8.1).

To establish baselines of comparison, we test single (uncombined) MLS sessions across the traditional (T) and post-quantum (PQ) ciphersuite classes in Table 8.1. For ciphersuite choices for APQ, we pair sessions based on their approximate security level and ciphersuites from each class as also shown in Table 8.1.

H-C-L	H-C-M	H-C-H	H-CA-L	H-CA-M	H-CA-H	X-Wing	H-C-X	H-CA-X	Alias	Ciphersuite
									Traditional	
*	*								T-EC-AES-L	MLS_128_DHKEMP256_AES128GCM_SHA256_P256
		*							T-EC-AES-H	MLS_256_DHKEMP384_AES256GCM_SHA384_P384
			*						T-Ed-AES-L	MLS_128_DHKEMX25519_AES128GCM_SHA256_Ed25519
				*	*				T-Ed-AES-H	MLS_256_DHKEMX448_AES256GCM_SHA512_Ed448
						~	*	*	T-Ed-Cha-L	MLS_128_DHKEMX25519_CHACHA20POLY1305_SHA256_Ed25519
										PQ Conf-only
*									PQ-C-L	MLS_128_ML-KEM512_AES128GCM_SHA256_P256
	*								PQ-C-M	MLS_128_ML-KEM768_AES256GCM_SHA384_P256
		*							PQ-C-H	MLS_192_ML-KEM1024_AES256GCM_SHA384_P384
						~	*		PQ-C-X	MLS_192_ML-KEM768_CHACHA20POLY1305_SHA256_Ed25519
										PQ Conf+Auth
			*						PQ-CA-L	MLS_128_ML-KEM512_AES128GCM_SHA256_MLDSA44
				*					PQ-CA-M	MLS_192_ML-KEM768_AES256GCM_SHA384_MLDSA65
					*				PQ-CA-H	MLS_256_ML-KEM1024_AES256GCM_SHA512_MLDSA87
								*	PQ-CA-X	MLS_192_ML-KEM768_CHACHA20POLY1305_SHA384_MLDSA65
										Hybrid KEM-Combiner
						*			X-Wing	MLS_256_XWING_CHACHA20POLY1305_SHA256_Ed25519

Table 8.1. Selected standardized traditional ciphersuites [3], draft standard PQ ciphersuites [200], and additional ciphersuites (notated in orange) for comparison. The various hybrid combinations used in APQ testing are indicated on the left side, with ~ representing the component combinations for comparison with X-Wing.

For selection of appropriate traditional ciphersuites to pair with a PQ counterpart, we generally defer to the security levels of the PQ ciphersuite components and selected a traditional to match. Since PQ Confidentiality-Only ciphersuites use ECDSA signature algorithms for signing in the PQ session, we also choose the matching ECDSA counterpart in the traditional ciphersuite category. For the full PQ (confidentiality and authenticity) ciphersuites which uses SUF-CMA DSAs (stipulated by NIST [5]), we select classical ciphersuites that have EdDSA (which is known to be SUF-CMA [202]) and the appropriate parameter-sets (e.g., Curve25519, Curve448). For the H-C-X combiner we create PQ and traditional ciphersuites based on the ML-KEM768 and EdDSAs used in X-Wing. These combinations are summarized in Table 8.1.

8.3.2 Performance Metrics

Tests are conducted using a two-participant session size across 500 epochs which are updated via empty commits (one commit per epoch) from a single party.[24] Amortization was tested by varying the ratio of Full to Partial APQ updates.

Time is a direct usability metric that is especially relevant for delay sensitive applications (e.g., secure messaging, media streaming, VoIP). Our time measurements are composed solely of the overhead of cryptographic operations (we do not send packets across a network). Most importantly, our time measurements provide insight into the cost of certain classical ciphersuites that had cascading effects in usage in APQ that would not be revealed by message size and memory usage alone. We measure the total time for 500 epochs, based on an average of 10 samples (i.e., 500 epochs are run 10 times, with the average taken across the 10). This is to mitigate for an spurious interference on time from background processes. Time values are derived using the Rust Criterion [203] microbenching tool which runs 10 iterations of the 500 epoch test and outputs the expected (average) total time for a single 500-epoch sample. As opposed to a single sample, the average of 10 iterations mitigates the effects of background processes effects on timing.

Message sizes offer another perspective in examining the impacts of implementing PQ algorithms. It is well established that PQ signatures and ciphertexts can be several orders

[24]For clarity and simplicity of measurement, all commits are performed by a single party. In normal operation, commits are performed by the various parties in the session.

of magnitude higher than their traditional counterparts [194]. For use of PQ in bandwidth limited settings, amortization effects of APQ on cumulative message sizes over time are especially important. MLS message sizes were calculated as a byte summation of the total PSK proposals, Welcome messages, and Commit messages sent between the two parties over 500 epochs.

Peak memory (RAM) consumption helps to determine the minimum RAM requirements for deployment of the various ciphersuites. Total memory measurements across the session run enables fair comparative analysis between the ciphersuites and the amortization ratios due to the clarity provided by the sizes of their overall memory footprint. Memory considerations are relevant for embedded systems and other resource constrained devices that are especially impacted by memory usage. Memory (RAM) usage was measured using the Rust Hotpath [204] process profiling tool which calculates granular peak RAM usage during certain function calls (e.g., encrypt, sign, verify, etc.) as well as the total memory footprint of a program. We measure memory usage as a total across 500 epochs.

8.3.3 Limitations

Our APQ implementation achieves the core functionality of the combiner (e.g., two MLS sessions tied together via use of an exported/injected PSK via the full commit mechanism from [119]). Our implementation does not include the APQInfo structure and the related bookkeeping mechanisms of Extensions Specification from [119]. Furthermore, our exported PSK uses the `export_secret` as opposed to the `epoch_secret` which is not accessible via the OpenMLS API. These limitations and deviations ultimately have negligible effects to our performance measurements: the APQInfo is a small fixed length byte string that gets added to encryption and signing operations and the secret used for PSK generation are the same sizes. As GitHub and mailing list discussion on the APQ draft is on-going as of the time of this writing, we leave high fidelity testing using a fully compliant APQ implementation to future work.

8.4 Results

To establish baselines, we first compare the performance of individual MLS sessions using the traditional, PQ Confidentiality, and PQ Conf+Auth ciphersuites from Table 8.1. Then

we compare the performance of the two modes of APQ across various Full:Partial update ratios to observe the effects of amortization. Finally, we compare hybrid combiners with the X-Wing KEM combiner and our APQ combiner.

In all cases we provide performance metrics as totals across the 500 epochs to clearly illustrate the differences in overall performance. Supplemental tables for per-epoch averages of commit message sizes can be found in Table 8.3, Table 8.4, and data summary in Table 8.5 of Section 8.6.

8.4.1 Baseline Testing

In terms of cumulative total run time, base-line comparisons in Figure 8.2 show that MLS sessions using PQ ciphersuites may perform no worse than comparable traditional ciphersuites. Our results indicate that at low security levels, Edwards Curve based DSA and KEM ciphersuites (T-Ed-AES and T-Ed-Cha) are faster than the lowest security level PQ ciphersuites. At high (256-bit) security levels, shown in red in Figure 8.2, both classes of PQ ciphersuites (PQ-C-H and PQ-CA-H) outperformed their traditional counterparts.

There is also high variation *within* traditional and PQ ciphersuite classes. Within the traditional class of ciphersuites, MLS sessions using ECDSA and ECDHKEM based ciphersuite were about four times as slow as the Edwards Curve ciphersuites. Additionally, at the high security level, the MLS session using a fully PQ ciphersuite (PQ-CA-H) ran 23% faster than the PQ confidentiality-only ciphersuite (PQ-C-H).

In terms of message sizes, our baseline results shown in Figure 8.3 indicate that comparative PQ ciphersuites are several orders of magnitude higher than their traditional counterparts. This is consistent with the stated outputs on the ciphertext sizes of ML-KEM and signature sizes of ML-DSA from prior research [194] [196] as well as empirical analysis [205]. It is worth noting that out of the class of traditional ciphersuites the ECDSA and ECDHKEM based ciphersuite has the largest ciphertext sizes.

We measure memory allocation by recording the peak RAM consumption (see Table 8.2) as well as total (cumulative across all 500 epochs) RAM consumption (see Figure 8.4). ...es across all security levels and DSA/KEM types have a ceiling of 1.2MB of peak memory usage. Meanwhile, the PQ ciphersuites increase in peak memory usage with the

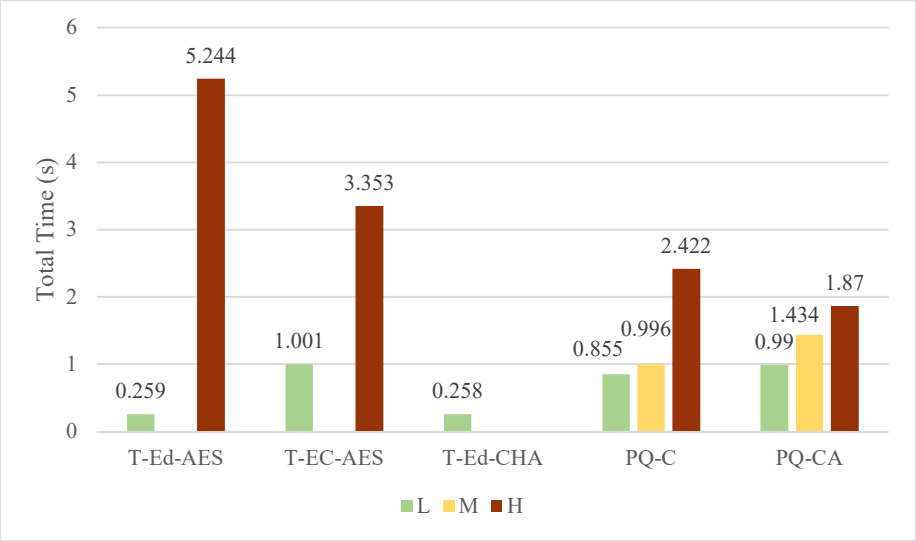


Figure 8.2. **Total time** comparison of individual MLS sessions (group size of 2) across 500 epochs (via empty commits). Clusters (Green-Yellow-Red) based on respective security level (Low, Medium, and High). Absence of a color in a cluster corresponds to a ciphersuite that was not included in testing. Calculations are based on an average of ten 500-epoch samples.

incorporation of larger KEMs and DSAs, as expected. These impacts are exacerbated when examined using total memory measurements collected across 500 MLS epochs. They reveal that PQ ciphersuites can be several orders of magnitude higher than their traditional counterparts. The difference is smaller for low security level ciphersuites (i.e., only 235% difference between T-EC-AES and PQ-C-L) than high security level ciphersuites (i.e., 750% difference between T-Ed-AES and PQ-CA-L). We expect these differences to increase further with larger group sizes and more epochs.

8.4.2 Amortization Testing using APQ

Results from amortization tests show that as the ratio of Full Updates to Partial Updates decreases, total run time, message size, and memory usage decreases which is consistent with amortization claims made in [119]. However, across all APQ modes and ratio, we observe diminishing returns of efficiency gains as the ratio goes beyond 1:50 with a limit based on the less performant component ciphersuite.

In all except H-CA-L and H-C-X, the total runtime exhibited by APQ is generally higher than

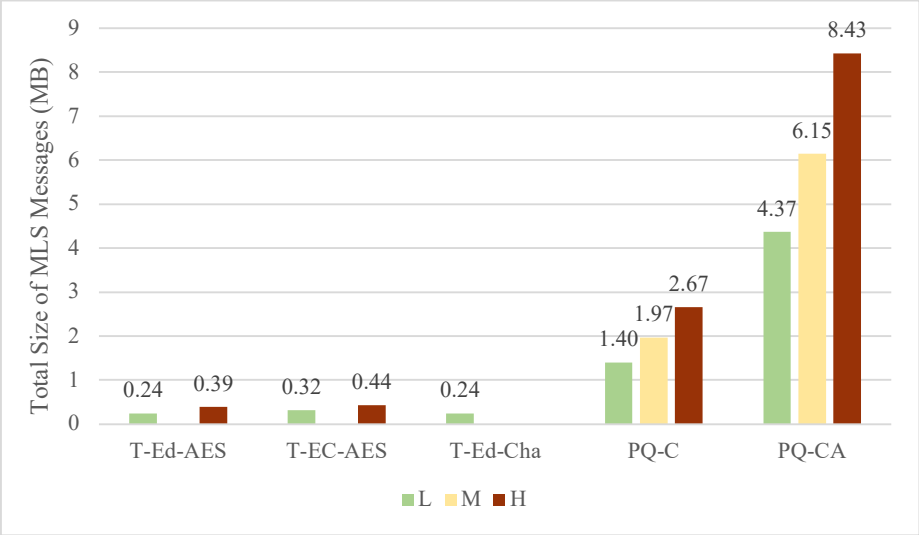


Figure 8.3. **Total message size** comparison of individual MLS sessions (group size of 2) for various traditional-only or PQ-only key update methods, totaled across 500 epochs (via empty commits). Clusters (Green-Yellow-Red) based on respective security level (Low, Medium, and High). Absence of a color in a cluster corresponds to a ciphersuite that was not included in testing.

just running the respective component PQ sessions solo – even at the lowest Full:Partial amortization ratios. This contradicts the intuition of APQ reducing the PQ costs over singular PQ sessions. However, the amortization effect of APQ is working as intended. Because partial updates are applied to the traditional session resulting in far more calls to the traditional ciphersuite components, the amortization effects of APQ likely skew toward the performance of the those traditional ciphersuites. The medium and high security APQ (in both C and CA modes) utilize the slower high security elliptic curve based KEM (X448 and P384) and DSAs (Ed448 and P384), so they set a floor for amortization that is higher than the PQ component alone. On the other hand, low security APQ (in both modes) used traditional KEM and DSAs that were faster than their PQ counterparts so the run-time performances for H-CA-L and H-C-X likely trended toward their traditional ciphersuite performance levels. In summary, because results from our solo session baselines showed PQ sessions performing faster than some of their traditional session counterparts, the APQ combiner also had mixed but correlated results. We leave testing alternative implementations of traditional ciphersuites across alternative crypto-providers and crypto-libraries to address the discrepancies to future work.

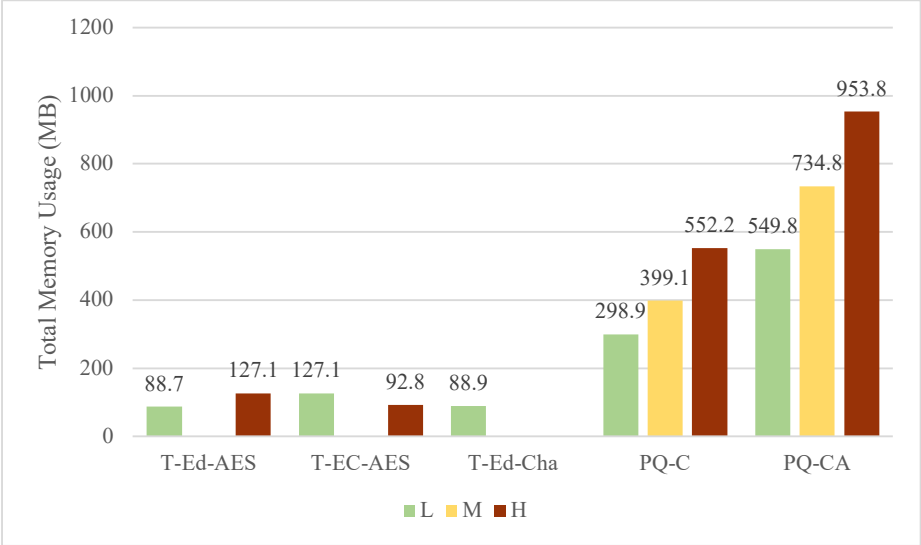


Figure 8.4. **Total memory (RAM)** comparison of individual MLS sessions (group size of 2) for various traditional-only or PQ-only key update methods, totaled across 500 epochs (via empty commits). Clusters (Green-Yellow-Red) based on respective security level (Low, Medium, and High). Absence of a color in a cluster corresponds to a ciphersuite that was not included in testing.

For both message size and memory usage, shown in Figure 8.6 and Figure 8.7, respectively, the effects of amortization are as expected. For message size, using the worst amortization strategy (1:1) results in producing a total bandwidth roughly equal to the sum of running the individual sessions separately. This is similarly so for the total memory usage. However, we observe an exponential decrease in message size and memory usage in all APQ modes as we decrease the frequency of Full:Partial commits to 1:10 which are on par with the solo traditional sessions in magnitude.

The peak RAM usage for Solo T, Solo PQ, and APQ combined MLS sessions, across the full 500-epoch test, are shown in Table 8.2. Results indicate consistent peak memory usage across all individual traditional ciphersuites. For the PQ sessions, peak memory usage generally increased with security level. This carries over to the amortized sessions, where results start high for the 1:1 Full:Partial update ratio but trend down toward their Solo PQ component peak memory measurements at lower ratios.

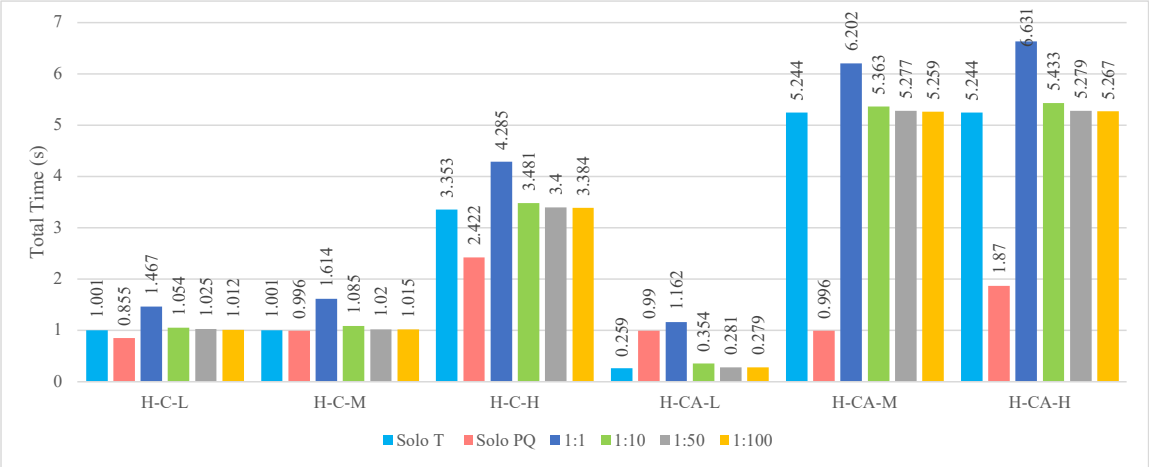


Figure 8.5. **Total time** comparison of APQ for Confidentiality-Only and Confidentiality+Authenticity at various security levels (Low, Medium, and High) across 500-epochs. Full:Partial key update ratios shown at variants of 1:1, 1:10, 1:50, and 1:100. Solo T and Solo PQ refer to individual components of the APQ mode and security level (see Table 8.1) that are ran in a non-combined configuration (e.g., the **sky blue** bar in H-C-L refers to T-EC-AES-L but the **sky blue** bar in H-CA-L refers to T-Ed-AES-L). Calculations are based on an average of ten 500-epoch samples.

Solo Traditional		Solo PQ		Combined (H-C / H-CA)				
Ciphersuite	Peak Mem (MB)	Ciphersuite	Peak Mem (MB)	Ciphersuite	Peak Mem (MB)			
					1:1	1:10	1:50	1:100
T-Ed-AES-L	1.2	PQ-C-L	1.3	H-C-L	1.9	1.3	1.3	1.3
T-Ed-AES-H	1.2	PQ-C-M	1.3	H-C-M	1.9	1.4	1.4	1.4
T-EC-AES-L	1.2	PQ-C-H	1.4	H-C-H	2.1	1.4	1.4	1.4
T-EC-AES-H	1.2	PQ-CA-L	1.5	H-CA-L	2.0	1.5	1.5	1.5
T-Ed-Cha-L	1.2	PQ-CA-M	1.6	H-CA-M	2.5	1.6	1.6	1.6
		PQ-CA-H	1.8	H-CA-H	2.6	1.8	1.8	1.8

Table 8.2. **Peak Memory (RAM)** usage across the 500 epoch test for solo traditional, solo PQ, and APQ for PQ Confidentiality-Only and PQ Confidentiality+Authenticity at various security levels (Low, Medium, and High). Full:Partial key update ratios shown at variants of 1:1, 1:10, 1:50, and 1:100.

8.4.3 Hybrid Combiners Comparison

In a head-on comparison between X-Wing and H-C-X (see **purple** and **teal** bars in Figure 8.8), which both offer PQ Confidentiality only, the APQ amortization quickly outpaces X-Wing which must call on ML-KEM768 operations for every commit (e.g., a 1:1 ratio using our terminology). As seen in Figure 8.8, at Full to Partial Update ratios smaller than

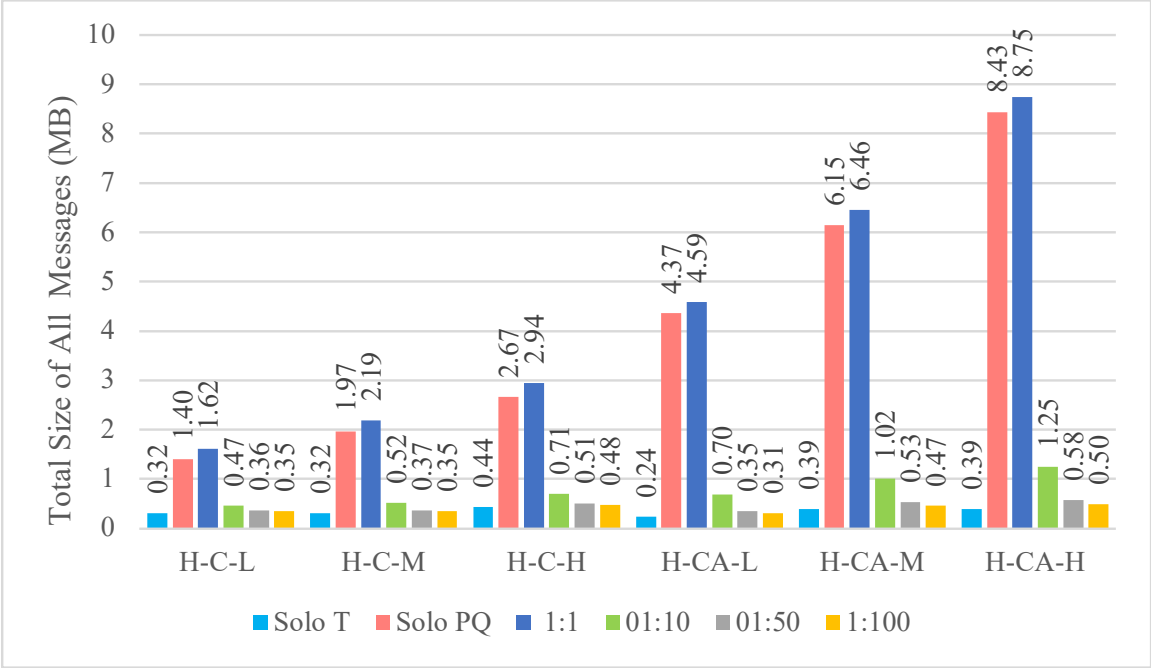


Figure 8.6. **Total message size** of all APQ messages for Confidentiality-Only and Confidentiality+Authenticity at various security levels (Low, Medium, and High), totaled across 500 epochs. Full:Partial key update ratios shown at variants of 1:1, 1:10, 1:50, and 1:100. Solo T and Solo PQ refer to individual components of APQ that are ran in a non-combined configuration.

1:2, the APQ speed-ups exhibited over X-Wing range from 26% improvement at a 1:5 ratio to upwards of 50% (diminishing) speed-ups after further reductions past 1:50. Both message size and total memory metrics follow a similar trend as shown in Figures 8.9 and 8.10, respectively.

When X-Wing run-time is compared with H-CA-X APQ variant (see **purple** and **dark-orange** bars in Figure 8.8), which has additional PQ-Authenticity with use of ML-DSA65, APQ is slower to outpace X-Wing. The eclipse in run-time performance over X-Wing occurs at 1:5 Full:Partial commit ratio for H-CA-X instead of the 1:2 ratio for H-C-X. This is notable as the APQ combiner at 1:5 outperforms X-Wing even with PQ authenticity included, which X-Wing does not provide. The run-time floor of H-CA-X is, as expected, approaching the solo run-time of T-Ed-Cha (see Figure 8.2), which provides the approximate floor for amortization.

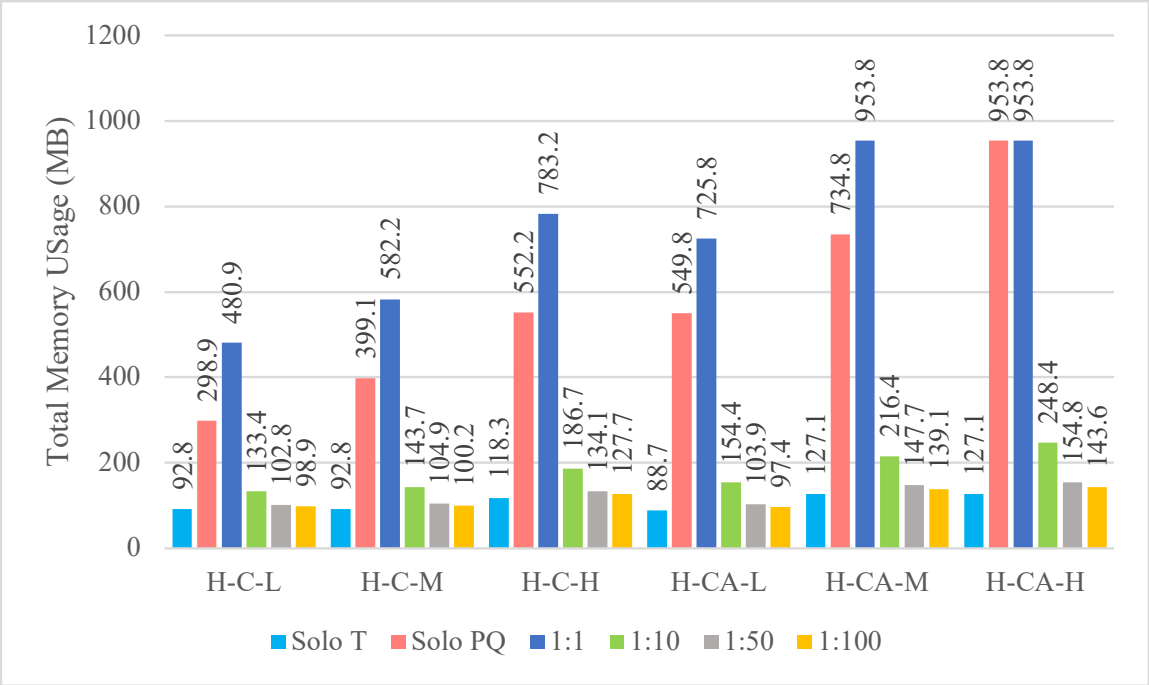


Figure 8.7. **Total memory (RAM)** comparison of APQ for Confidentiality-Only and Confidentiality+Authenticity at various security levels (Low, Medium, and High), totaled across 500 epochs. Full:Partial key update ratios shown at variants of 1:1, 1:10, 1:50, and 1:100. Solo T and Solo PQ refer to individual components of APQ that are ran in a non-combined configuration.

As shown in Figures 8.9 and 8.10, APQ becomes drastically more efficient than X-Wing in terms of message size and compute cost (total RAM usage) starting at Full:Partial commit ratios of 1:2 for H-C-X and at 1:5 for H-CA-X. The latter point is particularly significant. These results show that not only can APQ use the same ciphersuite security level as hybrids such as X-Wing (albeit with larger windows between PQ updates) for less 'bytes on the wire' and lower compute costs, but it can also provide support PQ-authenticity under such performance advantages. In fact, the message size and memory costs decrease exponentially toward a floor limit imposed by the classical ciphersuite component performance in each metric.

Although the amortization metrics shown in Figures 8.9 and 8.10 with performance advantages over X-Wing style hybrids use larger windows between PQ commits, they maintains

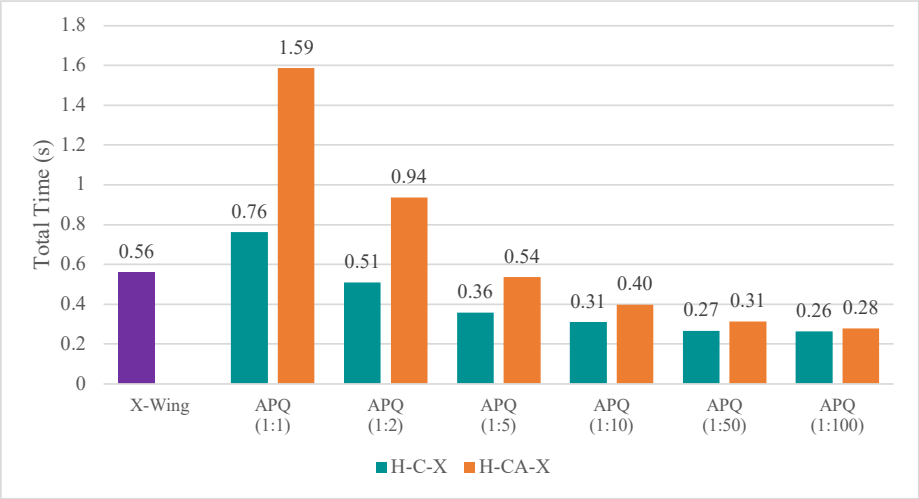


Figure 8.8. **Total execution time** comparison of Confidentiality-Only APQ vs X-Wing, at similar security levels, across 500 epochs. Full:Partial key update ratios shown at variants of 1:1, 1:10, 1:2, 1:5, 1:50, and 1:100. Calculations are based on an average of ten 500-epoch samples.

an identical traditional key update frequency. In practice, this provides for a great deal of flexibility for system developers and owners, especially as the needed frequency of quantum resistant key rotation may not be the same as for traditional in all cases. For resource constrained settings, this notably provides options for achieving quantum resistance which can still be tailored according to device constraints.

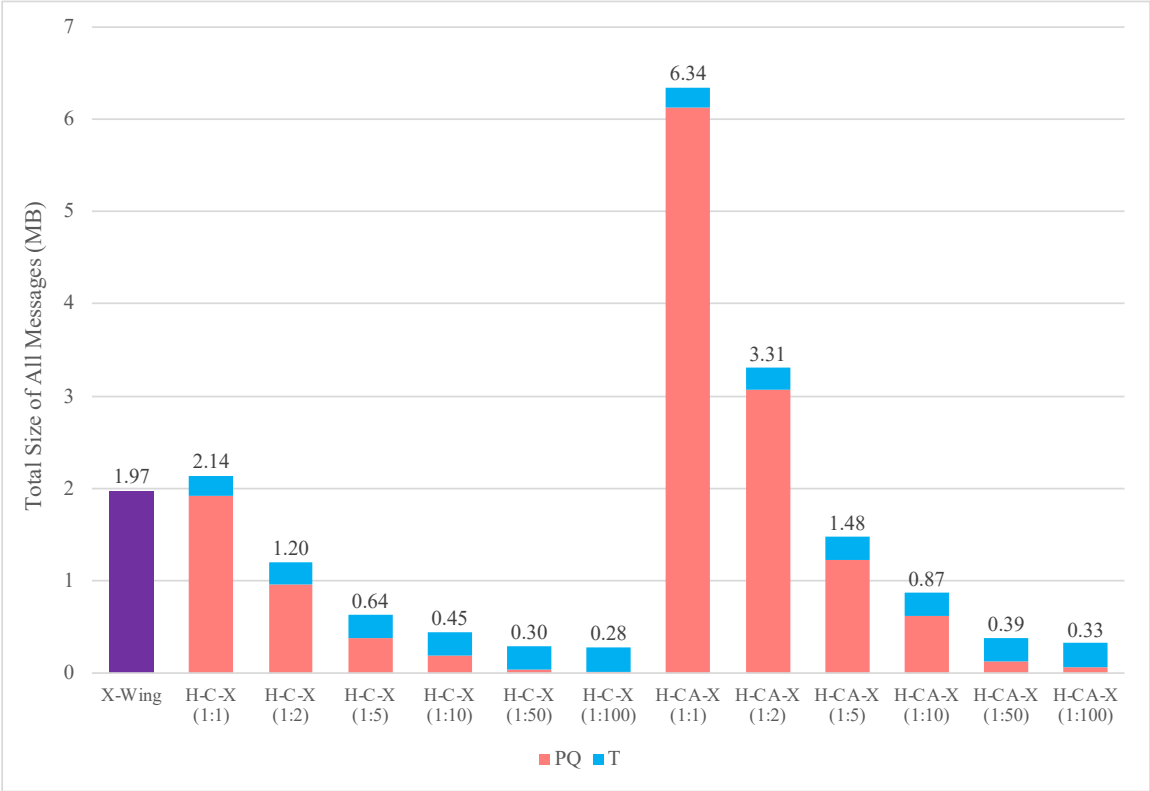


Figure 8.9. **Total size** of all APQ messages for Confidentiality-Only and Confidentiality+Authenticity APQ vs X-Wing, at similar security levels, across 500 epochs. Full:Partial key update ratios shown at variants of 1:1, 1:2, 1:5, 1:10, 1:50, and 1:100, totaled across 500 epochs. Component contributions (stacked) are shown in **pink** for PQ messages and **blue** for traditional messages.

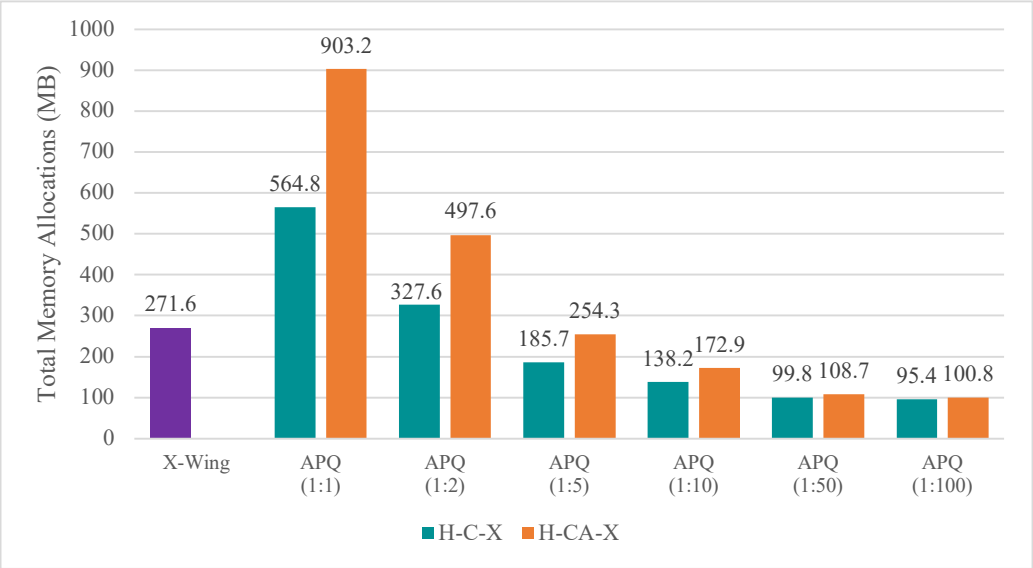


Figure 8.10. **Total memory (RAM)** comparison of Confidentiality-Only APQ vs X-Wing, at similar security levels, totaled across 500 epochs. Full:Partial key update ratios shown at variants of 1:1, 1:2, 1:5, 1:10, 1:50, and 1:100.

8.5 Conclusion

Our comprehensive benchmark evaluation of the APQ combiner demonstrates the effects of ciphersuite and amortization ratio selection on the protocol performance across time, bandwidth, and memory usage, including significant performance enhancements. Notably our results show that hybrid protocol combiner approaches for overhead amortization can, if strategically defined, substantially outperform simple PQ and hybrid KEM approaches alike. Moreover, this work points a way forward for achieving both PQ confidentiality and authenticity in resource limited use cases.

8.6 Summary Data Tables

We provide additional summary tables corresponding to our graphs here.

Table 8.3. Average per-epoch commit message size (in bytes) across 500 epochs. This is the averaged number of bytes per epoch specifically for commit messages.

Ciphersuite	Avg Commit Size (B)
T-Ed-AES-L	484
T-EC-AES-L	634
T-Ed-Cha-L	484
T-Ed-AES-H	780
T-EC-AES-H	875
PQ-C-X	3,847
PQ-C-L	2,807
PQ-C-M	3,944
PQ-C-H	5,338
PQ-CA-L	8,752
PQ-CA-M	12,307
PQ-CA-H	16,883
X-wing	3,943

Table 8.4. Average per-epoch commit message size (in bytes) for all APQ PQ and T ciphersuite combinations across 500 epochs. This is the averaged number of bytes per epoch specifically for commit messages.

Ciphersuite Pair	Avg PQ Commit Size (B)						Avg T Commit Size (B)					
	1:1	1:2	1:5	1:10	1:50	1:100	1:1	1:2	1:5	1:10	1:50	1:100
PQ Confidentiality Only												
H-C-L	2,809	2,809	2,808	2,809	2,808	2,808	220	444	579	624	660	664
H-C-M	3,945	3,946	3,946	3,946	3,946	3,946	220	444	579	624	660	664
H-C-H	5,340	5,339	5,340	5,340	5,339	5,341	284	596	784	847	897	903
H-C-X	3,849	3,849	3,849	3,849	3,849	3,849	213	366	458	488	513	516
PQ Confidentiality + Authentication												
H-CA-L	8,754	8,754	8,754	8,754	8,754	8,754	213	366	458	488	513	516
H-CA-M	12,309	12,309	12,309	12,309	12,309	12,309	330	572	718	766	805	810
H-CA-H	16,885	16,885	16,885	16,885	16,885	16,885	330	572	718	766	805	810
H-CA-X	12,260	12,260	12,260	12,260	12,260	12,260	213	366	458	488	513	516

Table 8.5. Performance metrics by ciphersuite, totaled across 500 epochs. Data summary from Figures 8.2 to 8.10.

Ciphersuite	Time (s)	RAM Allocations (MB)	Message Size (B)
T			
T-Ed-AES-L	0.259	88.7	241,894
T-EC-AES-L	1.001	92.8	316,789
T-Ed-Cha-L	0.258	88.9	241,894
T-Ed-AES-H	5.244	92.8	389,870
T-EC-AES-H	3.353	118.3	437,216
PQ Confidentiality Only			
PQ-C-X	0.567	388.4	1,921,088
PQ-C-L	0.855	298.9	1,401,817
PQ-C-M	0.996	399.1	1,969,631
PQ-C-H	2.422	552.2	2,665,805
PQ Confidentiality + Authentication			
PQ-CA-L	0.990	549.8	4,370,719
PQ-CA-M	1.434	734.8	6,145,969
PQ-CA-H	1.870	953.8	8,431,293
Hybrid and APQ Combiner			
X-wing	0.563	271.6	1,969,024
H-C-X (1:1)	0.764	564.8	2,135,464
H-C-X (1:2)	0.510	327.6	1,200,989
H-C-X (1:5)	0.359	185.7	637,739
H-C-X (1:10)	0.310	138.2	449,989
H-C-X (1:50)	0.267	99.8	299,789
H-C-X (1:100)	0.263	95.4	281,014
H-CA-X (1:1)	1.588	903.2	6,335,798
H-CA-X (1:2)	0.936	497.6	3,306,984
H-CA-X (1:5)	0.538	254.3	1,482,084
H-CA-X (1:10)	0.399	172.9	873,784
H-CA-X (1:50)	0.313	108.7	387,144
H-CA-X (1:100)	0.279	100.8	326,314

THIS PAGE INTENTIONALLY LEFT BLANK

Part V

Conclusion

THIS PAGE INTENTIONALLY LEFT BLANK

Conclusion

This dissertation has elevated the security and resiliency of space communications to new heights. From addressing challenges with tailoring key agreement for the degraded settings of space to upgrading the security of channel protocols used in space and ensuring that they are ready for quantum adversaries, this work makes significant contributions to both the fields of cryptography and network security. MLS has been enhanced to provide PCS in constrained settings and has been optimized and implemented for a post-quantum world. Popular protocols deployed and proposed for space communications have been redesigned, analyzed, and tested to utilize MLS, gaining new security guarantees and additional functionality. Significantly, our testing of the APQ-Combiner showed that it greatly outperforms the leading PQC approach (X-Wing) while providing more security. This has cemented its adoption by the IETF and industry (e.g., Phoenix R&D). Overall, the findings are supported by computational analysis using standard model assumptions as well as implementation and testing of protocols using reputable open-source libraries. In expanding the state-of-the-art in network security, a new approach of modular security that can adapt to environmental and operational constraints has been introduced to the existing single-stacked protocol paradigm. This shift in paradigm extends beyond space communications to network communications in general.

Research Impacts

Overall the scientific research impacts of this work include eighteen presentations, five accepted peer-reviewed papers, and one adopted Internet Draft by the IETF. The scientific research venues include IEEE conferences and workshops where the candidate was twice featured as an invited speaker and panelist by organizers at the 2025 IEEE Space Computing Conference Space Cyber Security workshop and at the 2025 IEEE Space-Terrestrial Internetworking Workshop, for work detailed in Chapter 5 and Chapter 6. Additionally, research from Chapters 3, 4, 7 and 8 has been accepted by the International Association for Cryptologic Research (IACR), the premier cryptography research venue. Lastly, the candidate has also presented this research to non-experts in the professional development of the Navy Engineering Duty Officer community in lecture forums to other officers and briefs to senior leaders at the Naval Research Lab, Program Executive Office for Integrated Warfare Systems, and Program Executive Office for Command, Control, Communications,

Computers and Intelligence. As a result, this research will be continued and developed into courses at the Naval Research Laboratory where the candidate will be stationed next.

This research has reached a global audience of network and security experts through six presentations at IETF meetings across four working groups as well as to engineers and researchers from all major space agencies (NASA, ESA, etc.) across four CCSDS meetings. As a result, it has gained stakeholders from major commercial entities like Amazon Web Services, Sandbox AQ, and PhoenixR&D as well as government stakeholders from the National Security Agency, NASA, and the National Reconnaissance Office (who financially supported parts of this research). These organizations value the novel security gains made in this research and have either implemented or tested our work, provided technical assistance and feedback, or provided financial support for our research to further their commercial and operational use cases.

Ongoing Work

Work is ongoing in drafting and standardizing IETF standards related to this work. Paired (Guardian) MLS will be solicited for adoption by the candidate at his next assignment at the Naval Research Laboratory. BPSec-MLS is in the process of being drafted by the candidate with help from co-authors and DTN working group members for submission to the IETF. A CCSDS Orange Book will accompany the BPSec-MLS draft. QUIC-MLS is being discussed for routing and adoption at the IETF.

Multiple government and industry partners have expressed interest in testing or adopting this work. At the encouragement of military leadership, the candidate will engage with program offices and other government research institutes to solicit further DoD testing and adoption of this work. This includes testing QUIC-MLS at Rim of the Pacific (RIMPAC) sponsored by Navy Information Warfare Center Pacific. Furthermore, the European Space Agency has requested testing of BPSec-MLS and QUIC-MLS in their CyberCube cube-sat program to be launched in 2026. The candidate will also engage in exploratory collaboration with Phoenix R&D, Anduril, and other parties who have expressed interest or have already adopted APQ-MLS and QUIC-MLS (i.e. Phoenix R&D).

Future Directions

The success of integrating MLS as a handshake mechanism for specific protocols in this dissertation suggests a broader opportunity: generalize the MLS-based handshake replacement for a larger set of IP-layer protocols (e.g., Internet Key Exchange, Secure Shell, and emerging transport-layer protocols). Future work should formalize the mapping between protocol-specific logic and MLS group operations so that MLS can provide authenticated group keying with minimal changes to existing protocol logic. This may include further expansions of MLS Safe Extensions to provide interoperability for cross-layer support. A careful engineering and formal-verification effort would ensure that security properties proven for MLS could carry over after protocol integration, while also measuring real-world impacts such as latency, message size, and resource consumption.

Another direction to take is certificate-transparency-style verification for MLS key packages: space communications require strong, auditable guarantees about which cryptographic material is in use across widely distributed and intermittently connected systems. Adapting the certificate transparency (CT) model to MLS key packages offers a pragmatic path to end-to-end verifiability: this would include a set of append-only, tamper-evident logs that publish Signed KeyPackage entries and issue succinct Signed Timestamps. Endpoints, ground stations, and monitors can obtain compact Merkle proofs and gossip-verified state to detect missuance or equivocation even when connectivity is delayed as has been similarly proposed by implementations of MQTT style MLS delivery services. Research should focus on log replication strategies tolerant of long round-trip delays and partitioning, privacy-preserving redaction or selective disclosure for sensitive mission data, efficient revocation and replacement of KeyPackages, and lightweight verification primitives suitable for resource-constrained on-board systems. This may be accomplished through use of distributed hash tables. Integrating CT-style transparency with MLS's PSK export and StrongBPSec audit records would provide operators and endpoints with robust forensic evidence of key distribution and protocol transitions, while remaining practical for deployed space networks.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX: Relevant IETF Drafts

A.1 IETF Submitted Draft: Securing BPSec Against Arbitrary Packet Dropping

This draft protocol extension corresponds to the paper presented in Chapter 3 and has been submitted to the DTN working group at the IETF who have been favorable to its adoption. It has also been presented to the DTN working group at the Consultative Committee for Space Data Systems (CCSDS) who were also favorable to it's adoption. Work is ongoing in refining the details of this document before official working group adoption.

The candidate co-proposed and helped develop the initial idea to its current form. The candidate wrote COSE Context Considerations and Scope Flag sections in the draft and assisted in general editing and refinement of the internet draft.

Delay/Disruption Tolerant Networking   B. Dowling
Internet-Draft   Kings College London
Intended status: Informational   B. Hale
Expires: 3 January 2026   X. Tian
                                            Naval Postgraduate School
                                                         B. Wimalasiri
                                              University of Sheffield
                                                          2 July 2025

Securing BPSec Against Arbitrary Packet Dropping
draft-tian-dtn-sbam-00

Abstract

   In this document we describe Strong Bundle Protocol Audit Mechanism
   (SBAM), an authentication protocol designed to provide cryptographic
   auditing services for the Bundle Security protocol.

About This Document

   This note is to be removed before publishing as an RFC.

   The latest revision of this draft can be found at
   https://bwimad.github.io/draft-xxx-str-bpsec/draft-tian-dtn-
   sbam.html. Status information for this document may be found at
   https://datatracker.ietf.org/doc/draft-tian-dtn-sbam/.

   Discussion of this document takes place on the Delay/Disruption
   Tolerant Networking Working Group mailing list (mailto:dtn@ietf.org),
   which is archived at https://mailarchive.ietf.org/arch/browse/dtn/.
   Subscribe at https://www.ietf.org/mailman/listinfo/dtn/.

   Source for this draft and an issue tracker can be found at
   https://github.com/bwimad/draft-xxx-str-bpsec.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on 3 January 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal
Provisions Relating to IETF Documents (https://trustee.ietf.org/
license-info) in effect on the date of publication of this document.
Please review these documents carefully, as they describe your rights
and restrictions with respect to this document. Code Components
extracted from this document must include Revised BSD License text as
described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

This document defines additional security features for the Bundle Protocol Security (BPSec) [RFC9172] and is intended in use for Delay Tolerant Networking (DTN) environments using BPSec to provide

security guarantees, SBAM is intended to provide additional security guarantees for BPSec communication between a security source (typically a bundle source), a security acceptor, and a security destination (typically the bundle destination).

The BPSec specification [RFC9172] defines BPSec as "an end-to-end security service that operates in all of the environments where the BP operates" and claims to provide "integrity and confidentiality services for BP bundles." In particular, BPSec enables partial processing of bundles, where an intermediate node acting as a security acceptor can process and remove security services. As a result, it is possible for an intermediate malicious nodes to simply drop blocks (and associated security extension blocks). StrongBPSec Audit Mechanism (SBAM) provides in-band integrity guarantees by cryptographic auditing via ledger blocks, mitigating the risk of undetected message deletion in BPSec. Specifically, SBAM addresses a critical limitation of BPSec by enabling destination nodes to verify whether security service blocks added by the bundle source were dropped, processed, or modified during transit, while retaining compatibility with existing BPSec deployments.

1.1. Notation

This section defines terminology that either is unique to the BPSec or SBAM and is necessary for understanding the concepts defined in this specification.

1.1.1. Bundle Protocol Terms

Bundle Protocol Agent: A node component that offers the Bundle Protocol services and executes its procedures.

Bundle Destination: The Bundle Protocol Agent (BPA) that receives a bundle and delivers the payload of the bundle to an Application Agent. Also, an endpoint comprising the node(s) at which the bundle is to be delivered. The bundle destination acts as the security acceptor for every security target in every security block in every bundle it receives.

Bundle Source: The BPA that originates a bundle. Also, any node ID of the node of which the BPA is a component.

Source Node: A BPA that creates an initial bundle.

Destination Node: A security acceptor BPA that is the bundle destination and processes the bundle payload.

Forwarder: Any BPA that transmits a bundle in DTN. Also, any node ID of the node of which the BPA that sent the bundle on its most recent hop is a component.

Intermediate Node: A security acceptor BPA that is not the bundle destination.

Intermediate Receiver, Waypoint, or Next Hop: Any BPA that receives a bundle from a forwarder that is not the bundle destination. Also, any node ID of the node of which the BPA is a component.

Path: The ordered sequence of nodes through which a bundle passes on its way from source to destination. The path is not necessarily known in advance by the bundle or any BPAs in DTN.

1.1.2. Security Terms

Cipher Suite: A set of one or more algorithms providing integrity and/or confidentiality services. Cipher suites may define user parameters (e.g., secret keys to use), but they do not provide values for those parameters.

Security Acceptor: A BPA that processes and dispositions one or more security blocks in a bundle. Security acceptors act as the endpoint of a security service represented in a security block. They remove the security blocks they act upon as part of processing and disposition. Also, any node ID of the node of which the BPA is a component.

Security Block: A BPSec extension block in a bundle.

Security Context: The set of assumptions, algorithms, configurations, and policies used to implement security services.

Security Operation: The application of a given security service to a security target, denoted as OP(security service, security target). For example, OP(bcb-confidentiality, payload). Every security operation in a bundle MUST be unique, meaning that a given security service can only be applied to a security target once in a bundle. A security operation is implemented by a security block.

Security Service: A process that gives some protection to a security target. For example, the BPSec specification defines security services for plaintext integrity (bib-integrity) and authenticated plaintext confidentiality with additional authenticated data (bcb-confidentiality). This SBAM specification defines security services for cryptographic auditing of security services added by the bundle source to the bundle destination.

Security Source: A BPA that adds a security block to a bundle. Also, any node ID of the node of which the BPA is a component.

Security Target: The block within a bundle that receives a security service as part of a security operation.

Security Verifier: A BPA that verifies the data integrity of one or more security blocks in a bundle. Unlike security acceptors, security verifiers do not act as the endpoint of a security service, and they do not remove verified security blocks. Also, any node ID of the node of which the BPA is a component.

1.1.3. SBAM-Specific Terms

BAB: Bundle Audit Block – a ledger block authenticated by the source node.

BRB: Bundle Report Block – a signed/verifiable block produced by an intermediate node that processed and discarded source-added blocks.

1.2. Motivation and Problem Statement

DTN recognizes an attacker with complete network access, affording them read/write access to bundles traversing the network. Eavesdropping, modification, topological, and injection attacks are all described in [RFC9172], Section 8.2. Therein, these "on-path attackers" can be unprivileged, legitimate, or privileged nodes depending on their access to cryptographic material: unprivileged nodes only have access to publicly shared information, legitimate nodes have additional access to keys provisioned for itself, and privileged nodes have further access to keys (privately) provisioned for others. There are currently no guarantees against privileged attacks.

In an effort to distinguish malice by intermmediate nodes, these classes can be further abstracted into honest security acceptors and dishonest forwarders. Honest forwarders are privileged nodes that faithfully execute the role of a BPA as described in [RFC9171], Section 3. Dishonest forwarders are unprivileged nodes that attempt to violate the integrity or confidentiality of blocks it processes (e.g. by dropping or modifying blocks). This is the gap we address with SBAM: BPSec under the default security context [RFC9173] has no cryptographic auditing mechanism for detecting modifications to a bundle between the SN and DN. With SBAM, all security acceptors (including the intended destination) are obliged to record their modifications

2. Design Decisions

In this section we describe the design decisions of BPSec, and describe how these are impacted through the use of SBAM.

TODO: Use RFC 9172 draft as starting point, discuss how SBAM changes these, or our design does not effect.

2.1. Block-Level Granularity

2.2. Multiple Security Sources

2.3. Mixed Security Policy

2.4. User-Defined Security Contexts

2.5. Deterministic Processing

2.6. COSE-Context Considerations

In conjunction with a proper PKI mechanism, SBAM may be used in the COSE-Context [draft-ietf-dtn-bpsec-cose] to provide further authentication enhancements to auditing. Specifically, through the use of digital signature algorithms rather than message authentication codes as described herein, SBAM in the COSE-context adds source authentication as well as authentication of intermediate nodes.

2.7. Unique Key Identifiers

The Bundle Protocol Security (BPSec) and its defined security contexts, as described in RFC9172 and RFC9173 respectively, rely on the assumption that local security policies will inform Bundle Protocol Agents (BPAs) of the appropriate cryptographic keys to use for each security context. This decentralized, policy-driven approach allows flexibility but introduces ambiguity when these policies are not uniformly enforced or clearly defined across participating nodes. In the absence of standardized key selection mechanisms, there is a risk that different BPAs may select conflicting keys for the same security context or inadvertently reuse keys across incompatible contexts. Such ambiguity can lead to key collisions, where multiple security contexts reference the same key identifier or cryptographic material unintentionally, undermining the security operations BPSec is intended to enforce. To mitigate this ambiguity, our proposed SBAM mechanism introduces a key_identifier as an explicit security context parameter, enabling BPAs to uniquely identify the correct cryptographic key for each context.

2.8. Scope Flag

The Integrity Security Context BIB_HMAC-SHA2 includes Integrity Scope Flags as a parameter set (see 3.2 and 3.3.3 in RFC9173). The value of the Integrity Scope Flag describe what information is used to construct the Integrity Protected Plain Text (IPPT) for a BIB. The existing Integrity Scope Flags in bit 2 and bit 3 refer to an excessive amount of information (block type code, block number, block processing control flags). Since we explicitly only use the block number in our calculations, this scope flag is redundant and we choose to remove it.

2.9. Security Blocks

In this section we describe the different Security Blocks used in BPSec and SBAM. In particular, we note that BPSec introduced two types of security blocks: the Block Integrity Block (BIB) and the Block Confidentiality Block (BCB) providing integrity and confidentiality and integrity, respectively.

In SBAM we also introduce the Bundle Audit Block (BAB) and the Block Report Block (BRB), which (when combined) enable security targets to verify only honest security intermediate nodes have processed missing BIB or BCBs.

2.10. Block Definitions

The BPSec specification defines two types of security blocks: the Block Integrity Block (BIB) and the Block Confidentiality Block (BCB). The SBAM specification defines two additional types of security blocks: the Bundle Audit Block (BAB) and the Block Report Block (BRB).

TODO: Check references are correct

* The BIB is used to ensure the integrity of its plaintext security target(s). The integrity information in the BIB MAY be verified by any node along the bundle path from the BIB security source to the bundle destination. Waypoints add or remove BIBs from bundles in accordance with their security policy. BIBs are never used for integrity protection of the ciphertext provided by a BCB. Because security policy at BPSec nodes may differ regarding integrity verification, BIBs do not guarantee hop-by-hop authentication, as discussed in Section 1.1 (https://www.rfc-editor.org/rfc/rfc9172.html#sup_sec_svc).

* The BCB indicates that the security target or targets have been encrypted at the BCB security source in order to protect their content while in transit. As a matter of security policy, the BCB is decrypted by security acceptor nodes in the network, up to and including the bundle destination. BCBs additionally provide integrity-protection mechanisms for the ciphertext they generate.

3. StrongBPSec Audit Mechanism Protocol Overview

The core guarantee provided by SBAM is a guarantee that, after correctly verifying the Bundle Audit Block, the destination node is assured that either

* all security blocks added by the Source Node have arrived without an unprivileged node dropping or modifying security block; or

* an honest intermediary has processed a security block.

Thus, for any bundle, the source node will generate all security blocks for their destination node exactly as in BPSec. Additionally, it will provide a Bundle Audit Block, which provides a cryptographically-authenticated digest of all security services it provided for the bundle, as well as all necessary uniquely identifying information, such as the key and block identifiers. This digest is added as the security result to the Bundle Audit Block.

Any honest intermediary node that processes a security block from the source bundle will also provide a cryptographic proof that they were privileged to perform this operation (demonstrated by providing a cryptographic signature over the identifying information for the security service contained in the security block). This signature is contained within the Block Report Block, which provides a cryptographically-authenticated digest of all uniquely identifying information of the security block it just processed, such as key and block identifiers.

Finally, when the destination receives the bundle, it will collate all identifying information contained within security blocks (generated by the source node) with identifying information contained within each BRB. Verifying this information against the cryptographic digest contained within the Bundle Audit Block enables the destination node to verify that no unprivileged node modified or dropped any security block between the source node and the destination node.

3.1. Bundle Audit Blocks (BAB)

This section describes the procedure used to compute a Message Authentication Code (MAC) tag for a bundle containing one or more security blocks added by the bundle SN.

NAVAL POSTGRADUATE SCHOOL | MONTEREY, CALIFORNIA | WWW.NPS.EDU

Each SN MUST attach a BAB immediately before the Payload Block. This BAB contains metadata describing all security blocks added by the SN and a MAC computed using a key shared with the DN.

3.1.1. Inputs

plaintext: A binary string constructed as the concatenation of the payload and metadata elements from each security block added by the SN. Formally:

plaintext = payload || { block_no || key_id || security_targets } for each security block added by the SN

where: - payload: The application data block. - block_no: The identifier of the block being protected. - key_id: The identifier for the cryptographic key used as specified by the security context - security_targets: A list of target block numbers to which the security operation applies.

key: The symmetric key (e.g., a pre-shared key or key-wrapped ephemeral key) used to compute the MAC. This key is identified by the key_id and MUST be established in accordance with the security context shared between communicating nodes.

3.1.2. Output

MAC tag: A cryptographic tag that provides data origin authentication and integrity verification. The MAC is calculated as follows:

MAC = HMAC(key, plaintext)

where: - HMAC is the keyed-hash message authentication code function as defined in [RFC2104].

3.1.3. Usage

The MAC tag generated by this procedure (alongside the plaintext) is attached to the BAB security block as the security_result field and MUST be verified by the DN. Failure to validate the tag indicates tampering or corruption of the bundle or associated metadata.

TODO: Describe how to reconstruct the BAB payload from the security blocks (e.g. BIB, BCB, BRB) present in the rest of the bundle.

3.1.4. Block Specific Data

TODO: describe the exact sub-fields for the security block. Broadly it follows the following structure:

* Number of Security Blocks (Integer)

* Key identifier (BAB key shared between the SN and DN)

* For each security block represented in the BAB:

  - Block number (of the original Security Block)

- Key identifier (of the original Security Block)
- Security Targets (of the original Security Block)
- Block Type Code (of the original Security Block)
- Security Parameters (of the original Security Block)

Note that the (Block, Key, Target, Code, Parameters) field should be ordered by block number to ensure consistent ordering between the SN generating the MAC tag and the DN verifying the MAC tag.

3.2. Bundle Report Blocks (BRB)

This section defines how a Message Authentication Code (MAC) is generated by an IN when processing and discarding a security block from a bundle. The resulting tag is attached to a newly created Block Report Block (BRB). This enables the DN to verify the legitimacy of IN(s) that process and discarded SN-originated added security blocks.

An IN that processes and discards any SN-originated security block MUST add a BRB. Each BRB cryptographically authenticates metadata about the discarded block (e.g., key_id, block_id, security_targets) and is authenticated with a unique key (independent of the BAB key) shared with the DN.

3.2.1. Inputs

plaintext: A structured binary string composed of identifying information related to the discarded block:

plaintext = block_no || key_id || security_targets

where: - block_no: The unique identifier of the discarded security block. - key_id: The identifier of the key used by the SN for the original security block. - security_targets: A list of block numbers to which the discarded security block originally applied.

key: A symmetric key used for computing the MAC. This may be a pre-shared key or a key-wrapped ephemeral variant, as specified by the active security context.

3.2.2. Output

MAC tag: A cryptographic tag calculated over the plaintext:

MAC = HMAC(key, plaintext)

where: - HMAC is the keyed-hash message authentication code function defined in [RFC2104].

3.2.3. Usage

The resulting MAC tag, along with the original plaintext, is attached to the *Bundle Report Block (BRB)* as the security results field. This allows the destination node to validate that the discarded security block was processed by an authorized intermediate node, and

that its original security configuration is cryptographically verifiable.

3.2.4. Block Specific Data

TODO: describe the exact sub-fields for the security block. Broadly it follows the following structure:

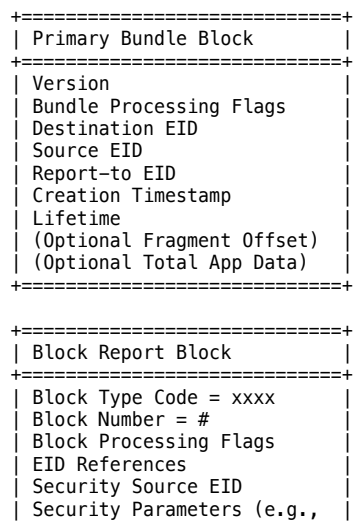
* Block number (of the original Security Block)
* Key identifier (of the original Security Block)
* Security Targets (of the original Security Block)
* Block Type Code (of the original Security Block)
* Security Parameters (of the original Security Block)
* MAC tag

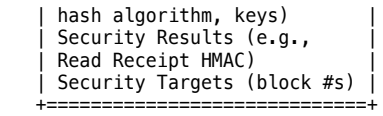
3.3. Blocks Excluded by SBAM

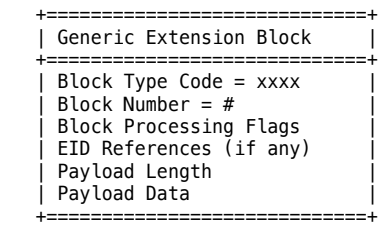
SBAM participants should exclude blocks that necessarily change throughout a bundle's life cycle from auditing. Extension blocks such as Hop-Count or Previous Node which change values SHOULD be excluded from SBAM protection to avoid unnecessary processing and overhead.

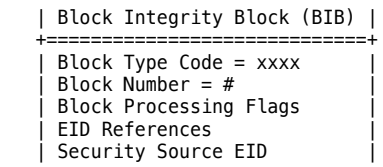
3.4. Integration with BIB/BCB

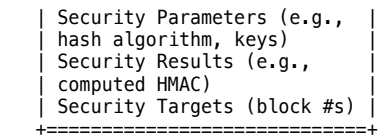
Existing BIB and BCB behavior is preserved. BAB and BRBs are *in-band* and protect against message deletion or replacement without the need for out-of-band policies.

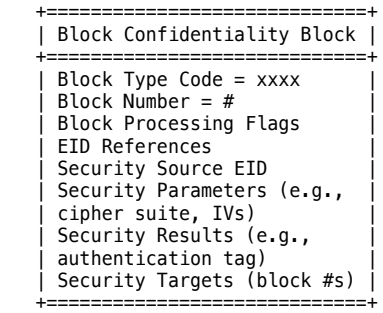


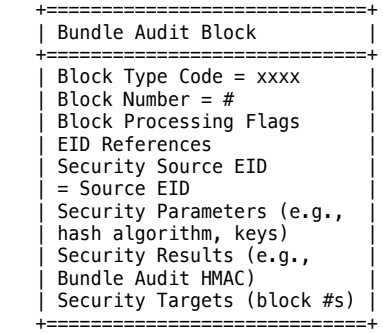

| hash algorithm, keys)
| Security Results (e.g.,
| Read Receipt HMAC)
| Security Targets (block #s)


Generic Extension Block
Block Type Code = xxxx
Block Number = #
Block Processing Flags
EID References (if any)
Payload Length
Payload Data


Block Integrity Block (BIB)
Block Type Code = xxxx
Block Number = #
Block Processing Flags
EID References
Security Source EID


| Security Parameters (e.g.,
| hash algorithm, keys)
| Security Results (e.g.,
| computed HMAC)
| Security Targets (block #s)


Block Confidentiality Block
Block Type Code = xxxx
Block Number = #
Block Processing Flags
EID References
Security Source EID
Security Parameters (e.g.,
cipher suite, IVs)
Security Results (e.g.,
authentication tag)
Security Targets (block #s)


Bundle Audit Block
Block Type Code = xxxx
Block Number = #
Block Processing Flags
EID References
Security Source EID
= Source EID
Security Parameters (e.g.,
hash algorithm, keys)
Security Results (e.g.,
Bundle Audit HMAC)
Security Targets (block #s)

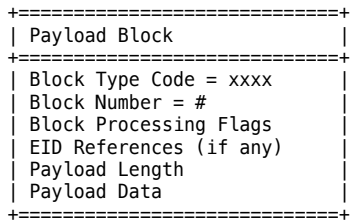


Figure 1. SBAM protected bundle

4. Processing Rules

* *SN*: Adds BAB after security blocks and before Payload.

* *IN*: Processes BIB/BCB, replaces with BRB as needed.

* *DN*: Validates all BRBs and BAB prior to accepting payload. If any validation fails, bundle MUST be discarded.

5. Security Considerations

5.1. Trivial Block Removal

SBAM allows for the detection of unauthorized deletion of SN-added BIB/BCB. This requires that the intended recipient checks for a BAB and rejects bundles without one as to avoid a trivial attack by a malicious IN of simply removing all security blocks.

5.2. Insider Attack

SBAM protected bundles are still vulnerable to *privileged insider* attacks unless asymmetric crypto is introduced. Malicious nodes with privileged access to keys associated with protected blocks may be able to modify SBAM block values undected (e.g. forge or overwrite BRB values).

6. IANA Considerations

New block types: - BAB – TBD - BRB – TBD

7. References

7.1. Normative References

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/rfc/rfc2104>.

[RFC9171] Burleigh, S., Fall, K., and E. Birrane, III, "Bundle Protocol Version 7", RFC 9171, DOI 10.17487/RFC9171, January 2022, <https://www.rfc-editor.org/rfc/rfc9171>.

[RFC9172] Birrane, III, E. and K. McKeever, "Bundle Protocol Security (BPSec)", RFC 9172, DOI 10.17487/RFC9172, January 2022, <https://www.rfc-editor.org/rfc/rfc9172>.

[RFC9173] Birrane, III, E., White, A., and S. Heiner, "Default Security Contexts for Bundle Protocol Security (BPSec)", RFC 9173, DOI 10.17487/RFC9173, January 2022, <https://www.rfc-editor.org/rfc/rfc9173>.

7.2. Informative References

[CryptoRocket]
"Cryptography is Rocket Science", n.d., <https://doi.org/10.62056/a39qudhdj>.

[draft-ietf-dtn-bpsec-cose]
Sipos, B., "Bundle Protocol Security (BPSec) COSE Context", 3 June 2025, <https://datatracker.ietf.org/doc/draft-ietf-dtn-bpsec-cose/>.

Appendix A. Abstract Security Block Representation

A.1. BAB

An example of the abstract security block structure based on Fig. 1 of the BAB is as follows. We assume the block numbers are sequential for the sake of illustration.

[4, 5, 7], / Blocks logged - BIB, BCB, Payload block number / 5, / Security Context ID - BAB-HMAC-SHA2 / 1, / Security Context Flags - Parameters Present / [2,[2, 1]], / Security Source - ipn:2.1 / [/ Security Parameters - 3 Parameters / [1, 6], / SHA Variant - HMAC 384/384 / [5, h'0xf000ba4'] / Key Identifier - Unique to BAB context /], [/ Security Results: 1 Result / [/ Target 1 Results / [1, h'deadbeefdeadbeefdeadbeefdeadbeef / MAC / deadbeefdeadbeefdeadbeefdeadbeef deadbeefdeadbeefdeadbeefdeadbeef']]]

A.2. BRB

An example of the abstract security block structure of the BRB is as follows.

[5], / Discarded Block - BCB block number / 5, / Security Context ID - BRB-HMAC-SHA2 / 1, / Security Context Flags - Parameters Present / [2,[2, 2]], / Security Source - ipn:2.2 / [/ Security Parameters - 3 Parameters / [1, 6], / SHA Variant - HMAC 384/384 / [5, h'0xcafebabe'] / Key Identifier - Unique to BRB context /], [/ Security Results: 1 Result / [/ Target 1 Results / [1, h'deadbeefdeadbeefdeadbeefdeadbeef / MAC / deadbeefdeadbeefdeadbeefdeadbeef deadbeefdeadbeefdeadbeefdeadbeef']]]

Authors' Addresses

Benjamin Dowling
Kings College London
Email: benjamin.dowling@kcl.ac.uk

Britta Hale
Naval Postgraduate School

Email: britta.hale@nps.edu

Xisen Tian
Naval Postgraduate School
Email: xisen.tian1@nps.edu

Bhagya Wimalasiri
University of Sheffield
Email: b.m.wimalasiri@sheffield.ac.uk

A.2 IETF Submitted Draft: Paired (Guardian) MLS

This draft corresponds to the paper presented in Chapter 4 and has been submitted to the MLS working group. However, adoption the working group is on hold. Work is ongoing in finding customer bases to advocate for this draft's adoptation.

This draft was jointly written by Elsie Fondevik, Britta Hale, and the candidate. The candidate specifically made initial drafts of the introduction, terminology, operational modes, explanation of operational modes, as well as tables and figures which were refined jointly. The candidate presented it to the MLS working group at IETF 117 on 28 July 2023 and has generally maintained the draft since with minor edits.

MLS
Internet-Draft
Intended status: Informational
Expires: 15 December 2025

E. Fondevik
Kongsberg Defense & Aerospace
B. Hale
X. Tian
Naval Postgraduate School
13 June 2025

Paired MLS - PCS in Limited Modes
draft-fondevik-mls-pairedmls-03

Abstract

This document describes the Paired Messaging Layer Security (MLS) extension that improves Post Compromise Security for devices that are unable to self-update using a trusted paired device.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 15 December 2025.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Paired MLS allows a trusted device to update the security parameters of another group member. The trusted paired device can be added to the group or can be another existing group member. The Paired MLS extension builds upon MLS (see [1]). This document presents two operational modes for the Paired MLS_ extension; interested readers can learn about other cases that were evaluated at [FHX23].

The Paired MLS extension describes a standard case where each device possesses its own signature key. To enable the standard Paired MLS extension, the MLS anchor node must accommodate being shared by at least two devices. If the anchor node is an MLS leaf node, this means that the leaf node would store at least two sets of signature keys. An additional operational mode is described, _Hidden_ mode, where the paired devices share a signing key and the paired device is able to issue digital signatures on behalf of the partner device in addition to PCS updates. Caveats to Hidden mode are discussed further under Security Considerations.

2. About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at _[Todo]_.

Discussion of this document takes place on the MLS Working Group mailing list (mailto:mls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/mls/. Subscribe at https://www.ietf.org/mailman/listinfo/mls/.

Source for this draft and an issue tracker can be found at https://github.com/PairedMLS/draft-pairedMLS.

3. Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on XX May 2024.

4. Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

5. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119], and [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms MLS client, MLS member, MLS group, Leaf Node, GroupContext, KeyPackage, Signature Key, Handshake Message, Private Message, Public Message, and RequiredCapabilities have the same meanings as in the [MLS protocol] https://www.rfc-editor.org/rfc/rfc9420.html (https://www.rfc-editor.org/rfc/rfc9420.html).

Generally, Paired MLS involves two paired devices, where one device can perform PCS updates in an MLS group on behalf of the other device. Without loss of generality, we define the one performing updates as the active device and the device not performing the update as the passive device:

Passive Device: A passive device is a user equipment device that is not issuing updates for a given period. Such a device may operate in receive-only mode or in another limited fashion such that sending regular keying updates is impractical or even impossible. A passive device may be offline or online, i.e., if the passive device is online it can receive application and group management messages, but is restricted from issuing updates.

Active Device: An active device is another user equipment device designated that is able to issue updates during the given period.

A passive device may be pre-paired with an active device such that the active device can issue updates on behalf of the paired passive device. The active device's updates enable a passive device to PCS heal after a compromise for improved post-compromise security. Paired devices may switch roles between active and passive. Also a device may be paired with multiple others, such that it can issue

updates on behalf of several paired passive devices.

Anchor: The MLS node that acts as a shared access node between the paired and passive device is called an anchor. The anchor can be a leaf node of a group member, where the paired devices both have access to the leaf node information but our nominally outside of the MLS tree. The anchor can also be a shared intermediate node on the path to the root of an MLS tree, and as such the the anchor may be shared with multiple only MLS members in addition to the paired devices. Any MLS members that share the anchor node in their parent tree MUST be paired.

Shared Randomness: In order for one device to perform cryptographic updates on behalf of another, they must share a source of randomness that is kept in secure storage. This is in addition to each of the devices' own randomness source. In cryptographic analysis terms, such shared randomness must be stored with similar protections as signature keys in order to not be assumed as compromised in the event of other state compromise. Practically, this is accomplished by sharing a seed for pseudorandom number generation between the two. This random seed IS RECOMMENDED be shared via secure hardware or sharing MAY be bootstrapped over a 1-to-1 channel, where the added MLS PCS guarantees from this draft are contingent on the security of the 1-to-1 channel. Readers are encouraged to see [FHX23] for security tradeoff analysis.

6. Extension Execution

The extension assumes the use of the MLS protocol where the device that desires to execute the extension is already an MLS group member and thus has access to an MLS leaf node. The group member initiating this extension MUST first negotiate the shared randomness with the device it will pair with: this SHOULD be done via secure hardware and MAY be done through an out-of-band, 1-to-1 channel. This extension assumes that the randomness is stored securely, similarly to signature private keys.

Signature key management determines whether the extension is used in standard mode or with hidden mode. In standard mode, both of the paired devices must have their own signing keys, distinct from the anchor. This is the case whether the paired devices are both MLS group members with their distinct leaf nodes, or if the anchor node is an MLS group member leaf node. In the latter case, the extension would require the ability to associate multiple signature public keys to a leaf node. In hidden mode, the paired devices may use the anchor's signing key, thus obfuscating the actions of the individual devices. The private signing key MAY be shared among the paired devices offline or out-of-band.

After sharing randomness and establishing an anchor node, the devices are considered "paired" and either device may update on the other's behalf. When the active device issues an update to the group on behalf of the passive device, it will also issue a _Notify_ notification message to the passive device to ratchet forward its group key using the shared randomness. This ensures that the passive device stays synchronized with the group epoch so it can process updates and commits made by other group members. This notification message is sent in place of a normal update to the paired device, i.e., such that the _Notify_ message does not contain secret keying material. Since, unlike the update message the _Notify_ does not

contain information about the key update, an adversary that has compromised the passive device and tries to decrypt the message learns nothing about the new epoch state, achieving PCS for the passive device. The _Notify_ notification message is formatted similarly to an update message, such that the distinction between the two is opaque to the DS.

Messages transmitted in the Paired MLS extension are those inherited from MLS [1] with the following changes:

* If an *Update* message is sent by A, such that A is an active device and is sending an update on behalf of B, then A computes the update as in MLS except for the KEM to B. Instead of the KEM for B, A computes the _Notify_ message. <!--

* A *CeasePair* message is sent from a paired device to its paired devices with whom the initiating device wishes to discontinue paired MLS extension. The command is followed by a self remove then group addition.

Optionally, if randomness between paired devices is transmitted online the following commands are additionally utilized: * A _ShareRand_ message is sent to negotiate the shared randomness between pairing devices. * A _InitPairing_ message notifies the recipient about devices that wish to pair. The message contains the identities of the pairing devices and a flag for standard or hidden mode operation. If hidden mode is set, the message MUST be sent directly to the target device in a secure 1-to-1 chanel and MUST contain the signature key pair of the anchor leaf node. If standard mode is set, the recipient MUST be the directory, and the directory will associate the public signatures of the requesting devices to the anchor node of the initiating device. * An _Accept_ message is sent back to the initiator to confirm successful paring. This means that both devices have shared randomness and that signing keys have been provisioned accordingly. -->

[TODO] Add context on pairing remove messages (if these are allowed to be transparent to the group) MLS commands such as Remove, GroupInfo KeyPackage and Welcome take the form and are processed as according to [1].

6.1. Issuing a Paired Update

Once A and B have been paired, active device A can issue an update on behalf of passive device B. A sends the update to the rest of the MLS group as a normal commit. From the perspective of other MLS group members this update will be indistinguishable from any other MLS update preformed by A. Furthermore, in hidden mode, updates by the paired devices A or B will appear to come from the anchor, due to the shared signing key. A will send the _Notify_ message to B, where

Notify is indistinguishable to other group members from a commit message to B. The _Notify_ message signals to B how it should use the shared randomness to derive the necessary update for the new group key in order to stay in sync with the new group epoch.

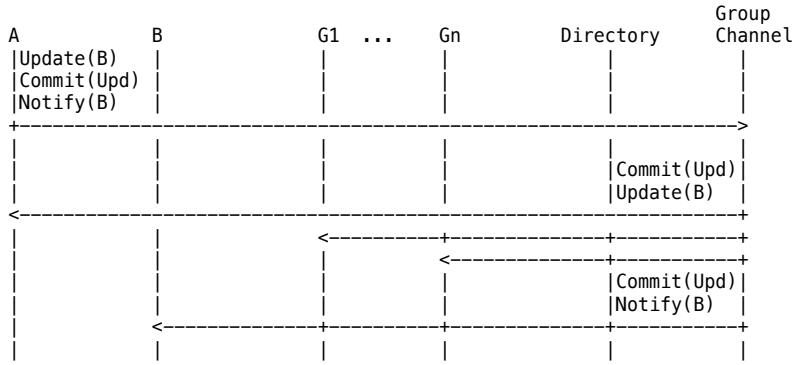


Figure 1 Active device A updates with a commit on behalf of B to the group. The commit message is process as in MLS for all members (A, B, G1, ..., Gn). The Update message is processed as in MLS, but with the change that the update for B is computed as a notify message instead. The notify message is formatted the same as a commit message form the view of the DS. Remaining MLS group members, which are labeled G1, ..., Gn, will receive the standard update messages from the DS.

If any other MLS group member sends proposals or commits to the paired devices the process will follow the flow as defined in RFC9420 [1].

6.1.1. Termination of Pairing

To end Paired MLS extension, either A or B may issue an out-of-band request to its paired device to cease paring. This request notifies the other device to stop using the shared randomness to update on the other's behalf. In standard mode, pairing termination can be enforced through a self-remove and re-add to the group.
In hidden mode, an out-of-band cease pairing request can similarly be issued, but enforcing the termination is more challenging since removing either device is opaque to the MLS given the shared signature key. This will be discussed further under Security Considerations. It is possible, however, to enforce termination of the pairing and hidden mode by removing both devices and re-adding under separate signature keys.

7. Security Considerations

The goal of the MLS extension is to reduce the PCS security risk in cases when group members are unable to update, or updates are seldom. The extension allows other MLS group member devices, or other additional devicses belonging to the same user to update on the passive device's behalf. The structure of a shared anchor node in the MLS tree and various devices under that in a subtree can be

attractive for practical operational reasons, and the hidden mode could further allow a user to have multiple devices listed under their user identity leaf node; however there are security caveats to exploiting such structures and we will summarize trade-offs here.

7.1. Transport Security

Recommendations for preventing denial of service (DoS) attacks, or restricting transmitted messages are inherited from MLS. Furthermore, message integrity and confidentiality is, as for MLS, protected. <!-- An adversary that can observe network traffic will be able to discern group membership. The MLS packets for the extension are designed to be indistinguishable from regular MLS packets for anyone but the paired devices. As such, a network observer should not be able to determine which devices are paired based solely on packet analysis, however, since paired devices communicate using a separate channel, a network observer might be able to discern general communication from pairing by observing timing and frequency. To prevent the separated communication form leaking information directly, this channel MUST be encrypted. We RECOMMEND using a TLS connection as a minimum.

7.2. Security of Shared Randomness

If the shared randomness between paired devices is leaked then any entity in possession of this information will be able to generate the group session key when either of the devices update on behalf of the other. As such, the shared randomness MUST be stored, securely and encrypted on all applicable end devices when not in use. Furthermore, we strongly RECOMMEND that the random seeds are loaded offline through hardware. If this is not possible a secure, and encrypted channel MUST by utilized to negotiate, or distribute the randomness.
-->

7.3. Post Compromise Security and Forward Secrecy

The main goal of the extension is to reduce epoch sizes when a group member is unable to update. A full security analysis pertaining PCS and FS can be found in [FHX23]. If the extension is not utilized or if paired devices are simultaneously unable to update, FS and PCS security is reduced to that of the original underlying MLS protocol. The PCS benefits from active device updates are contingent on how the shared randomness is stored; if the passive device stores the shared randomness in active memory with other MLS state, then the PCS benefits cannot be assumed. Instead, the shared randomness MUST be stored more securely as with the signature private keys. Furthermore, we strongly RECOMMEND that the random seeds are loaded offline through hardware. If this is not possible, then the out-of-band 1-to-1 channel utilized to negotiate or distribute the randomness is critical to the security benefits; compromise of that negotiation or distribution reduces the PCS guarantees to that of RFC4920 [1].

7.4. Discontinuation of Pairings

[Todo] currently operating under a single paired device. If multiple all need to be removed and then re-added later. Termination of pairing can be signaled as described above; in standard extension

NAVAL POSTGRADUATE SCHOOL | MONTEREY, CALIFORNIA | WWW.NPS.EDU

mode, if a malicious or unwitting device A ignores the signaling and continues to update on behalf of device B, there is no negative impact on security as B can still issue its own updates using its unique randomness. A can of course disrupt the key schedule if it ignores the signaling to terminate pairing and uses the shared randomness after B deletes it, but this similar as in MLS [1]. For such reasons, it is RECOMMENDED that B maintain the shared randomness after signaling termination of pairing until confirmation has been received from A. This does not affect forward secrecy.

In hidden mode, where devices share a signature key, termination of pairing requires removal and re-addition of both devices, such that they are registered with separate signature keys. It is not possible to remove only one device, as any removed device will maintain access to the signature private key in the group.

7.5. Impersonation to the Group

In Paired MLS standard mode, distinct signing keys are used by the main device and its paired device when issuing an update. Impersonation of other MLS group members is therefore not feasible given that the signature public keys are known.

In hidden mode, a single signing key is used by all paired devices. This could allow one or more paired devices to be opaque to the MLS group, which inhibits the MLS goals of transparency of group membership but supports possible user side goals of limiting tracking (e.g., if Alice possesses multiple devices that are group members). Thus, devices using the Paired MLS extension in hidden mode MUST be associated with the same group membership user identity, i.e., the paired devices may all belong to Alice but they should not belong to separate users Alice and Bob.

Without the ability to interrogate the delivery service for anonymous hidden pairings, compromised or malicious paired devices may eavesdrop undetected in hidden mode. If a group key is leaked somehow, PCS can be achieved through an update by either of the paired devices. However, if the shared randomness source is compromised on one device, then both devices are irrevocably compromised as the attacker could duplicate generation of the update secrets used on either device. Similarly, the shared signature key in hidden mode means that it is impossible to remove a hidden device member and a hidden member can easily start new groups, impersonating other members; this is similar to signature key compromise in MLS [CHK21]. It is for these reasons that it is required that hidden mode is only used for devices associated with the same MLS user.

7.6. Visability of paired devices to Delivery Service

The detection of the active/passive status of the paired devices to the rest of the group is possible in standard mode, but the detection of pairing is not. Thus other group members may see that device A has updated frequently without knowing that it is on behalf of B. This is because standard mode uses distinct signature keys for each device to issue signed updates to the group.

NAVAL POSTGRADUATE SCHOOL | MONTEREY, CALIFORNIA | WWW.NPS.EDU

TODO If there is a consideration that the lack of pairing awareness in the group may result in a devices ejection from the group, it is possible to signal to the group that devices are paired and updates have been performed on behalf of B.

In hidden mode, the DS is still aware of the devices A and B but may not be aware of the pairing status. The anchoring node's signature key is used by both devices, but whether or not they possess shared randomness to perform updates on the behalf of the other is not known to the DS.

8. Extension Requirements to MLS

8.1. Leaf Node Contents

The MLS leaf node will need to support multiple signature keys for the public guardian. The leaf node content is modified by changing signature_key to a vector of SignaturePublicKey. <!-- ## Shared Randomness Establishment The security of this extension is based upon the security of the out-of-band channel to used to establish shared randomness. We RECOMMEND the shared-randomness be installed via protected hardware in the same way that long-term signing keys stored such that it is infeasible to be accessed by an adversary. The shared-randomness MAY be shared via a secure 1-to-1 channel such as a key encapsulation mechanism between the devices.

8.2. Notifications Between Paired Devices

The notification message sent to the passive device to forward ratchet its group key must be secured from forgery and replay attacks. If an attacker were able to to prompt either devices to update, then they would fall out-of-sync and be unable to decrypt future group messages.

8.3. Multiple Signature Keys per MLS NODE

-->

9. IANA Considerations

[TODO] Determine an extension code to use

10. References

[I-D.ietf-mls-protocol] Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., and K. Cohn-Gordon. "The Messaging Layer Security (MLS) Protocol". Work in Progress, Internet-Draft, draft-ietf-mls-protocol-20, 27 March 2023. https://datatracker.ietf.org/doc/html/draft-ietf-mls-protocol-20 (https://datatracker.ietf.org/doc/html/draft-ietf-mls-protocol-20)

10.1. Normative References (i.e. RFCs)

[1] https://www.rfc-editor.org/info/rfc9420 (https://www.rfc-editor.org/info/rfc9420) "MLS RFC" [2] https://www.rfc-editor.org/info/rfc5246 (https://www.rfc-editor.org/info/rfc5246) "TLS RFC"

10.2. Informational References

[FHX23] E. M. Fondevik, B. Hale, and X. Tian. "Guardianship in Group Key Exchange for Limited Environments". Cryptology ePrint Archive, Paper 2023/1761. 11 November 2023. https://eprint.iacr.org/2023/1761 (https://eprint.iacr.org/2023/1761)

[CHK21] C. Cremers, B. Hale, K. Kohbrok. "The Complexities of Healing in Secure Group Messaging: Why Cross-Group Effects Matter". USENIX Security Symposium 2021: 1847-1864

Acknowledgments

Contributors ## Authors

Authors' Addresses

Elsie Fondevik
Kongsberg Defense & Aerospace
Email: elsie.fondevik@kongsberg.com

Britta Hale
Naval Postgraduate School
Email: britta.hale@nps.edu

Xisen Tian
Naval Postgraduate School
Email: xisen.tian1@nps.edu

A.3 IETF Submitted Draft: QUIC-MLS

This individual draft corresponds to the paper in Chapter 7 and was originally submitted to the QUIC working group (WG) at the IETF on recommendation by the Talking IP to Other Planets (TIPTOP) WG. As we explain in Chapter 5 and Chapter 7, the use of MLS as alternative key establishment mechanism for QUIC is more appropriate for the space setting. The purpose of this draft is to specify how to replace the TLS handshake in QUIC with MLS. The draft was jointly written with Benjamin Dowling (Kings College London), Britta Hale (Naval Postgraduate School), Bhagya Wimalasiri (Kings College London), Konrad Kohbrok (PhoenixR&D), and Raphael Robert (PhoenixR&D).

The candidate contributions include writing a majority of all sections and figures. The candidate did so with guidance and editorial assistance from co-authors.

QUIC
Internet-Draft
Intended status: Informational
Expires: 2 January 2026

B. Dowling
Kings College London
B. Hale
Naval Postgraduate School
K. Kohbrok
R. Robert
Phoenix R&D GmbH
X. Tian
Naval Postgraduate School
B. Wimalasiri
University of Sheffield
1 July 2025

Securing QUIC with MLS
draft-tian-quic-quicmls-00

Abstract

This document describes how Messaging Layer Security (MLS) can be used in place of Transport Layer Security (TLS) to secure QUIC.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://xisentian.github.io/ietf-quic-mls/draft-tian-quic-quicmls.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-tian-quic-quicmls/.

Discussion of this document takes place on the QUIC Working Group mailing list (mailto:quic@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/quic/. Subscribe at https://www.ietf.org/mailman/listinfo/quic/.

Source for this draft and an issue tracker can be found at https://github.com/xisentian/ietf-quic-mls.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 2 January 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the
document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal
Provisions Relating to IETF Documents (https://trustee.ietf.org/
license-info) in effect on the date of publication of this document.
Please review these documents carefully, as they describe your rights
and restrictions with respect to this document. Code Components
extracted from this document must include Revised BSD License text as
described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

Recent advances in key exchange protocol design for communications under network delays, disruption, and supporting low latency has offered new alternative key establishment techniques in the form of continuous key agreement. One such continuous key agreement protocol has been standardized by the IETF, namely Messaging Layer Security (MLS) (RFC9420). The MLS key agreement handshake can be generalized for dynamic or degraded communications settings that do not support duplex links, have highly mobile devices, and/or require asynchronous communications. QUIC's use of TLS for key agreement was primarily designed for relatively short-lived client-server connections with synchronous key initialization over reliable network availablity. MLS offers an alternative to users for cases where network reliability, attentuation, or disruption are concerns through asychronous key updates which also provide post-compromise security, enabling long-lived sessions with controllable key refresh periodicity. The MLS key agreement handshake can be slotted into

QUIC in a fairly straightforward manner, preserving other needed QUIC functionality. The combination of QUIC and MLS thus addresses a need for a robust security protocol in certain evolving communication environments.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Notation

We use terms from MLS [RFC9420], TLS [RFC8446], and QUIC [RFC9000]. Below, we have restated relevant terms and define new ones:

Application Message: The private application data payload transported and encrypted by QUIC.

Authentication Service (AS): An abstract architectural component of MLS that provides mechanisms for verifying the authenticity of group members, typically by issuing credentials (e.g., certificates) that bind identities to cryptographic keys

Delivery Service (DS): An abstract architectural component of MLS for reliably delivering MLS messages (e.g., handshake or application messages) between group members while ensuring proper ordering.

Control Message: An MLS Proposal or Commit message to change the cryptographic state, as opposed to application data.

Key Derivation Function (KDF): A Hashed Message Authentication Code (HMAC)-based expand-and-extract key derivation function (HKDF) as described in RFC5869.

Key Encapsulation Mechanism (KEM): A key transport protocol that allows two parties to obtain a shared secret based on the receiver's public key.

4. Scope

While MLS is designed for group settings, we limit discussions in this document to the two-party communications case. This choice is deliberate to avoid significant changes to QUIC and MLS.

5. Requirements for Integration

5.1. QUIC-HS Requirements

As an integrated secure transport protocol QUIC can be broken into two major components: a handshake layer (HS) responsible for key agreement and management and a record layer (RL) which provides secure (via HS keys) and reliable transport. Specifically, the HS layer requires the following from TLS as specified in RFC9001:

1. The handshake protocol must function as an authenticated key exchange (AKE) that produces distinct and unrelated keys for every connection where the server is always authenticated and the client is optionally authenticated.

2. Transport parameter authentication for both endpoints with

additional confidentiality protection for the server transport parameters.

3. Provide means for authenticated negotiation of an application protocol.

With respect to (1) MLS provides authenticated key agreement to achieve the same outcome of AKE without exchanging key material. Using existing PKI certificates, communicating partners can generate MLS Key Packages to begin MLS sessions that produce indistinguishably random keys. Mutual authentication is enabled through asymmetric

verification of credentials within MLS Key Packages. External commits which are used for "open" MLS groups can be used to allow an unauthenticated client to be added (i.e. optional client authentication). With respect to transport parameter negotiations in (2), MLS satisfies this with publication of a signed GroupInfo object inside Welcome messages that contains all the necessary information to join a group and a public key to encrypt a secret to the existing group members. For (3) Authenticated negotiation of application protocols can be achieved using the MLS Content Advertisement Extension in [I-D.ietf-mls-extensions] which would be included in the GroupInfo object.

5.2. MLS Requirements

The prerequisits for MLS are the Delivery Service (DS) and Authentication Service (AS) functionalities as specified in RFC9750. A DS ensures MLS messages are delivered to all participants and enforces ordering of commits. For example, the DS can be a centralized server or a reliable transport protocol like TCP. An AS provides the issuance and verification of user credentials. For example, the AS can be any existing web Public Key Infrastructure (PKI) (e.g. certificate authority based scheme).

For most cases using QUIC, the DS functionality is satisfied by QUIC RL through guarantees for reliable ordered delivery of messages just as it provides TLS. The AS functionality is provided through existing PKI certificates already used by TLS. In non-traditional settings (e.g. dynamic topologies, delay/disruption prone environments) where the DS functionality cannot be met by QUIC RL alone, other mechanisms to ensure (ordered) delivery MUST be employed. Likewise, the AS functionality must also exist. Details and recommendations for alternative strategies relating to either functionalities are outside the scope of this document.

6. Protocol Overview

At a high level, MLS provides the traffic keys used to encrypt and authenticate QUIC's secure channel. In turn, QUIC's transport logic which handles ordering and reliable delivery of packets helps to maintain the MLS state. Note, additional mechanisms may be used to maintain MLS state in the event that QUIC transport is insufficient.

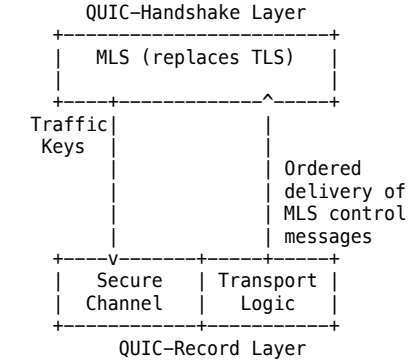


Figure 1: QUIC-MLS layer interactions

6.1. MLS Functionality

As a stateful protocol, MLS leverages a cryptographic structure called a ratchet tree to establish and maintain a shared cryptographic group state. The tree's root is the group's shared secret and each MLS group member is represented by a unique leaf node containing their HPKE encryption public key and signature public key from the AS. Intermediate nodes contain the HPKE encryption public key pairs calculated by members of the subgroup they cover. This property enables efficiency in updates to the ratchet tree. MLS group operations (e.g. add, remove, update) trigger modifications from each leaf along the path of intermediate nodes from the leaf to the root, thereby ratchetting the group state forward into a new epoch. Group operations are solicited through the DS via MLS Proposal messages and are processed (aka committed) by MLS Commit messages. Therefore, a QUIC-MLS session with respect to the HS functionality, is the creation and modification of the MLS ratchet state (the shared state).

Each epoch of an MLS session has a distinct asymmetric ratchet tree as previously described that seeds the MLS Key Schedule used to derive shared secrets used for key generation (e.g. via a Key Derivation Function). Values from the asymmetric ratchet tree and key schedule are used to derive a symmetric ratchet tree called the Secret Tree which is used to generate keys used for encrypting and authenticating application messages. QUIC-MLS provides keys from this symmetric ratchet tree to the QUIC record layer.

6.2. QUIC-MLS Execution

An initiator may unilaterally start a QUIC-MLS session with a partner. They then update the keys throughout their communications by sending MLS updates with each message until either parties terminates the session by issuinng a (self) removal proposal and commit. An important distinction from QUIC-TLS is that key updates here are ratcheted in accordance with the MLS Key Schedule (see Section 8 of [RFC9420]) as to provide forward secrecy and post-

compromise security rather than using QUIC's native key update mechanism (see Section 6.1 of [RFC9001]) which only provides forward secrecy. Furthermore, by attaching a configurable series of previous MLS Commit messages to each sending stream, we ensure that the protocol is both robust enough to handle truly asynchronous settings under eventually consistent DS but also flexible enough to adapt to synchronous settings with strongly constistent DS assumptions.

6.2.1. Initialization

An initiating client can create MLS Key Packages based off of preloaded credentials obtained from the AS (i.e. via digital certificates) or through direct communications. In the absence of an AS, initators may use preinstalled long term signature keys from itself and the intended partner(s) to create the Key Packages necessary to start a session. The initiator uses the Key Packages to create an MLS Add proposal, which once committed to, starts the MLS session with the corresponding partner. In order for the other member to join and initialize their own cryptographic state, they will need to recieve their personalized MLS Welcome message. The initiator can wait to receive a MLS Commit from the partner or unilaterally commit their own Add proposal and thereby asynchronously intiate a session. If they unilaterally commmit their own Add proposal, then 0-RTT data may be sent along with these first batch of packets using the encryption key derived from the MLS group secret in accordance with the MLS Key Schedule (see Section 8 RFC9420). After the recipient verifies the initiator's Key Package and uses the Welcome message to construct the shared MLS state, the shared key is established and can be used by the record layer to commence secure communications via 1-RTT packets.

6.2.2. Updates

When a member chooses to update the group key, they send an Update proposal and the other member commits to that proposal to ratchet the group state into the next epoch. Alternatively, one party may serve as the session admin with commit authority or may unilaterally update (via an empty commit): in either case, when an eventually consistent DS is used, they MUST attach the antecedent series of commits leading

the the current epoch in order for the other member(s) to reconcile state agreement. In the two party case the series of commits reduces to simply the last commit.

To tighten the FS/PCS compromise windows, senders MAY attach an update proposal with every data message sent.

6.2.3. Termination

Session termination occurs when any party sends a (self) removal proposal and commit it. This MAY be sent multiplexed with the last data message or as an independent stream with no data message. For self removals in a two-party group, the proposer may terminate the session without waiting for commitment depending on application needs. The remove proposal shall be sent as the payload of a Crypto frame inside of a Handshake packet.

6.2.4. Resumption

Similar to TLS where 0RTT keys are used to resume a session and quickly send encrypted application messages, MLS can make use of resumption pre-shared key (PSK) from historical epochs to send 0RTT encrypted data. With TLS, data sent encrypted using 0RTT key alone

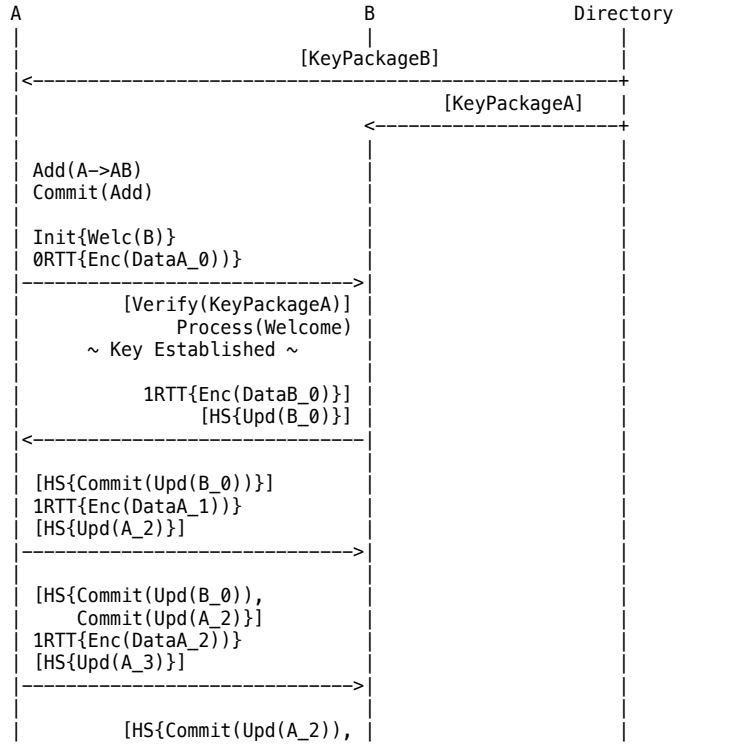
NAVAL POSTGRADUATE SCHOOL | MONTEREY, CALIFORNIA | WWW.NPS.EDU

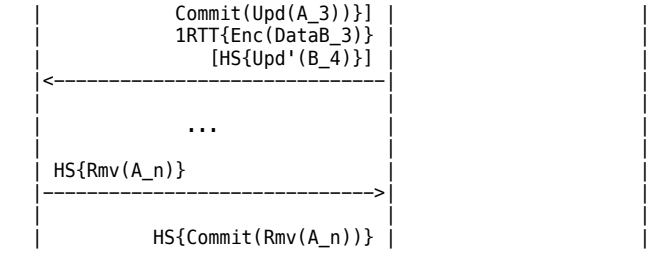
is not forward secret and suffers from replay attacks. With MLS, a member uses a Reinit proposal which describes the historical group state they belonged to to rejoin the group with their resumption PSK. Once the Reinit proposal is committed it cannot be duplicated (i.e. conflicting commit) making QUIC-MLS immune from the types of replay attacks that 0RRT TLS keys are vulnerable to. The Reinit proposal for QUIC-MLS is sent in in the Crypto frame of a Initial Packet, behaving similar to an External Join. With an encryption key derived from the new epoch secret (e.g. via HKDF Expand) of the new state, the 0-RTT data can be sent along with the Reinit proposal and commits.

6.2.5. Example Execution

The following two-party QUIC-MLS execution illustrates initialization, updates, termination, and reinitialization of a session with per-message updates for the strictest security and commit transcripts for the an eventually consistent DS. What is shown can be relaxed to accommodate longer update windows and/or streamlined by reducing or eliminating commit transcripts for highly available and reliable networks which support a strongly consistent DS. Specifically, the protocol can be adapted to the synchronous and strongly consistent DS setting by removing the positive acknowledgement of latest commit HS packet (e.g. removing A's second Commit(Upd(B_0) message and B's HS{Commit(Upd(A_2)} message). The

bracketed (e.g. optional) per-message ratchetting updates which follow each 1RTT data message shown may also be removed to allow more longer security windows (see Security Considerations (Section 9.2)).





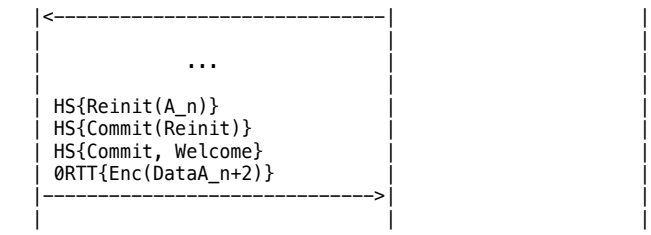


Figure 2: Client A creates a two-party QUIC-MLS session with Client B, and then (optionally) updates the group state until termination (via self- removal). Each MLS message sent is accompanied by a configurable series of commits to proposals. Brackets ('[]') indicate implicit/ optional communications steps. Handshake (HS), Init, and 0RTT packets wrap MLS messages indicated by braces ('{}'). Values following underscores ('_') indicate the associated epoch number.

TODO: Double check the reinit process -- is it better to just restart a new session? Seems like a new MLS session would have fewer steps in the two party case.

7. Packet Protection

As with QUIC with TLS, QUIC-MLS protects packets with keys derived from the MLS state, using an AEAD algorithm selected common to the communicating partners from their KeyPackage objects. Some key changes to how QUIC with TLS does packet protection (see Section 5 of [RFC9001]) are as follows:

* Initial Packets, which should be used to send MLS Welcome messages that are already encrypted with the joiner's public key retain their AEAD encryption using the secret derived from initial salt and connection ID as specified in Section 5.2 of [RFC9001]. Initial packets, which previously did not provide confidentiality or authentication to the payload, now carries a payload that DOES have confidentiality and authentication guarantees from MLS. NOTE: It may be redundant to keep the AEAD protection but removing it has downstream effects on Retry Packet usage/abuse. *TODO:* Decide if we want to even keep the Retry mechanism since we can form a MLS session unilaterally. ``

* Handshake Packets are proctected by keys derived from the MLS epoch secret.

* 0-RTT Packet Protection is provided by the traffic key computed from with the MLS Welcome message sent concurrently.

7.1. Key Derivation

For QUIC-MLS, traffic keys are derived the same as they are for QUIC with TLS except that the MLS Epoch Secret is used in place of the TLS Master Secret. Furthermore, the Early Secret, an artifact of the TLS key schedule, is never derived as it is unused (see Section 7.1 of [RFC8446]). All other keys used to encrypt or sign public and private MLS group control messages are derived according to the MLS Key Schedule (see Section 8 of [RFC9420]) using the appropriate lables. For example the following shows how traffic secrets used to encrypt 1RTT packets are derived from the MLS Epoch Secret:

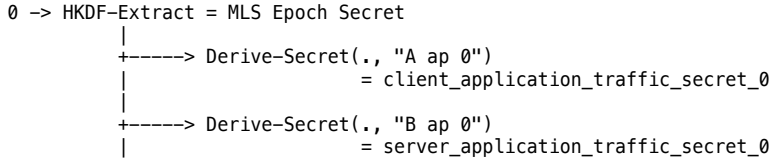


Figure 3: QUIC traffic key derivation from MLS Epoch Secret

TODO: Technically MLS doesn't expose access to the epoch secret, if we want to abide by safe extensions then we ought to use exporter secret here right?

7.2. Key Deletion

In general, keys should be deleted immediately after they are consumed. The key deletion schedule of MLS aligns with QUIC's policies for discarding keys in that members may keep unconsumed values for handling out-of-order message delivery. Therefore, keys should be deleted in accordance first with the QUIC policies for discarding keys and then the MLS Key Deletion Schedule (Section 9.2 of [RFC9420]) for MLS specific values.

8. Message Framing

The main interface from QUIC used by MLS are the CRYPTO and NEW_TOKEN frames. As payloads of Initial, Handshake, and 0RTT packets, the CRYPTO frame will carry a majority of MLS Handshake messages that facilitate group control. Initial packets only carry the MLS Welcome and External Join messages to add new users to a group. Previous group members who wish to rejoin a group (using their MLS Resumption Preshared Keys) may use the NEW_TOKEN frame inside of Initial or

Retry packets to resume membership in the session with their PreSharedKey proposal as tokens. Handshake packets carry CRYPTO frames that have as payloads all other MLS proposals and commits in accordance with RFC9420.

```
Initial_Packet{
  [...]
  Packet Payload = CRYPTO_FRAME{}
}
```

```
Handshake_Packet{
  [...]
  Packet Payload = CRYPTO_FRAME{}
}
Retry_Packet{
  [...]
  Retry_Token = NEW_TOKEN_FRAME{}
}
CRYPTO_FRAME{
  Type (i) = tbd,
  Offset (i),
  Length (i),
  Crypto Data = <MLS_Message>
}
NEW_TOKEN_FRAME{
  Type (i) = 0x07,
  Token Length (i),
  Token = MLS_PreSharedKey_Proposal
}
```

Figure 4: QUIC-MLS packet and frame structures for carrying MLS messages (e.g. proposals, commits, welcome, etc.)

9. Security Considerations

9.1. Attacks on Authentication

While message integrity is provided by the symmetric key used in AEAD, insider attacks on non-repudiation (e.g., source forgery) on application messages may still be possible because QUIC headers and payloads aren't signed. While the content (including sender information) is protected by AEAD which uses the symmetric group key, a malicious insider may still be able decrypt, modify, and re-encrypt the content using the same symmetric group key that they can compute. Thus, application messages sent by one member may be reordered or modified by another. However, in terms of group control, non-repudiation is unaffected because handshake messages are protected as signed MLS private messages.

9.2. Ratchet Window

The frequency of updates to the MLS group state can be adjusted as desired based on either time or number of messages. Following the maximum 604800 seconds (7 day) limit of the ticket_lifetime from TLS that forces a new DH handshake to establish fresh secrets, QUIC-MLS epochs must similarly not exceed 7 days in duration without an update.

9.3. Use of 0RTT

QUIC-MLS use of 0-RTT differs from QUIC-TLS in that a fresh key generated by the MLS state is used instead of using a pre-shared key generated from a previous TLS session between the communicating parties. QUIC-TLS uses 0-RTT and 1-RTT to explicitly differentiate the security levels but in QUIC-MLS their security levels are the same since they both are based on fresh randomness (akin to how 1-RTTs are protected by ephemeral Diffie Hellman keys). Unlike the caveats to using 0-RTTs for QUIC-TLS to thwart against certain types of replay attacks, QUIC-MLS has none.

10. IANA Considerations

TODO

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9001] Thomson, M., Ed. and S. Turner, Ed., "Using TLS to Secure QUIC", RFC 9001, DOI 10.17487/RFC9001, May 2021, <https://www.rfc-editor.org/rfc/rfc9001>.

[RFC9420] Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., and K. Cohn-Gordon, "The Messaging Layer Security (MLS) Protocol", RFC 9420, DOI 10.17487/RFC9420, July 2023, <https://www.rfc-editor.org/rfc/rfc9420>.

11.2. Informative References

[I-D.ietf-mls-extensions] Robert, R., "The Messaging Layer Security (MLS) Extensions", Work in Progress, Internet-Draft, draft-ietf-mls-extensions-06, 19 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-extensions-06>.

Acknowledgments

TODO acknowledge.

Authors' Addresses

Benjamin Dowling
Kings College London
Email: benjamin.dowling@kcl.ac.uk

Britta Hale
Naval Postgraduate School
Email: britta.hale@nps.edu

Konrad Kohbrok
Phoenix R&D GmbH
Email: konrad.kohbrok@datashrine.de

Raphael Robert
Phoenix R&D GmbH
Email: ietf@raphaelrobert.com

Xisen Tian
Naval Postgraduate School
Email: xisen.tian1@nps.edu

Bhagya Wimalasiri
University of Sheffield
Email: bmwimalasiri1@sheffield.ac.uk

A.4 IETF Adopted Draft: Amortized PQ MLS Combiner

This draft originated from joint work with Britta Hale (Naval Postgraduate School), Joël Alwen (AWS Wikr), and Marta Mularczyk (AWS Wickr) and led to the paper in Chapter 8. It has been adopted by the MLS working group where it is currently on a last call for IETF standardization. There have been two other independent implementations of this draft protocol by PhoenixR&D and bitbltr (active MLS WG member) in addition to our own in Chapter 8.

The original idea for this draft came from Britta Hale. It was jointly refined and analyzed by all coauthors involved. The candidate wrote the majority of this specific draft, to include all sections and figures, with the help and guidance from the co-authors.

Messaging Layer Security   J. Alwen
Internet-Draft   AWS
Intended status: Standards Track   B. Hale
Expires: 20 December 2025   Naval Postgraduate School
                                                       M. Mularczyk
                                                              AWS
                                                            X. Tian
                                           Naval Postgraduate School
                                                     18 June 2025

Flexible Hybrid PQ MLS Combiner
draft-ietf-mls-combiner-01

Abstract

This document describes a protocol for combining a traditional MLS session with a post-quantum (PQ) MLS session to achieve flexible and efficient hybrid PQ confidentiality and authenticity that amortizes the computational cost of PQ Key Encapsulation Mechanisms and Digital Signature Algorithms. Specifically, we describe how to use the exporter secret of a PQ MLS session, i.e., an MLS session using a PQ ciphersuite, to seed PQ guarantees into an MLS session using a traditional ciphersuite. By supporting on-demand traditional-only key updates (a.k.a. PARTIAL updates) or hybrid-PQ key updates (a.k.a. FULL updates), we can reduce the bandwidth and computational overhead associated with PQ operations while meeting the requirement of frequent key rotations.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://mlswg.github.io/mls-combiner/draft-ietf-mls-combiner.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-mls-combiner/.

Discussion of this document takes place on the Messaging Layer Security Working Group mailing list (mailto:mls@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/mls/. Subscribe at https://www.ietf.org/mailman/listinfo/mls/.

Source for this draft and an issue tracker can be found at https://github.com/mlswg/mls-combiner.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 20 December 2025.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction

A fully capable quantum adversary has the ability to break fundamental underlying cryptographic assumptions of traditional Key Encapsulation Mechanisms (KEMs) and Digital Signature Algorithms (DSAs). This has led to the development of post-quantum (PQ) cryptographically secure KEMs and DSAs by the cryptographic research community which have been formally adopted by the National Institute

of Standards and Technology (NIST), including the Module Lattice KEM (ML-KEM) and Module Lattice DSA (ML-DSA) algorithms. While these provide PQ security, ML-KEM and ML-DSA have significant overhead in terms of public key size, signature size, ciphertext size, and CPU time compared to their traditional counterparts. This has made achieving PQ entity and data authenticity particularly challenging. The hybrid approach in this draft amortizes the PQ overhead costs enabling practical PQ confidentiality or PQ confidentiality _and_ PQ authenticity.

Moreover, research arms on side-channel attacks, etc., have motivated uses of hybrid-PQ combiners that draw security from both the underlying PQ and underlying traditional components. A variety of hybrid security treatments have arisen across IETF working groups to bridge the gap between performance and security to encourage the adoption of PQ security in existing protocols, including the MLS protocol [RFC9420].

Within the MLS working group, there are various ways to approach PQ security extensions:

1. A single MLS ciphersuite for a hybrid post-quantum/traditional KEM. The ciphersuite can act as a drop-in replacement for the KEM, focusing on hybrid confidentiality but not authenticity, and does not incur changes elsewhere in the MLS stack. As a

confidentiality focus, it addresses the the harvest-now / decrypt-later threat model. However, every key epoch incurs a PQ overhead cost.

2. Mechanisms that leverage hybridization as a means to not only address the security balance between PQ and traditional components and achieve resistance to harvest-now / decrypt-later attacks, but also use it as a means to improve performance of PQ use while achieving PQ authenticity as well.

This document addresses the second topic of these work items.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The terms MLS client, MLS member, MLS group, Leaf Node, GroupContext, KeyPackage, Signature Key, Handshake Message, Private Message, Public Message, and RequiredCapabilities have the same meanings as in the MLS protocol [RFC9420].

3. Notation

We use terms from from MLS [RFC9420] and PQ Hybrid Terminology [I-D.ietf-pquip-pqt-hybrid-terminology]. Below, we have restated relevant terms and define new ones:

Application Message: : A PrivateMessage carrying application data.

Handshake Message: : A PublicMessage or PrivateMessage carrying an MLS Proposal or Commit object, as opposed to application data.

Key Derivation Function (KDF): : A Hashed Message Authentication Code

NAVAL POSTGRADUATE SCHOOL | MONTEREY, CALIFORNIA | WWW.NPS.EDU

(HMAC)-based expand-and-extract key derivation function (HKDF) as described in [RFC5869].

Key Encapsulation Mechanism (KEM): : A key transport protocol that allows two parties to obtain a shared secret based on the receiver's public key.

Post-Quantum (PQ) MLS Session: : An MLS session that uses a PQ-KEM construction, such as described by FIPS 203 from NIST. It may optionally also use a PQ-DSA construction, such as described by FIPS 204 from NIST.

Traditional MLS Session: : An MLS session that uses a Diffie-Hellman (DH) based KEM as described in [RFC9180].

PQ/T: : A Post-Quantum and Traditional hybrid (protocol).

4. The Combiner Protocol Execution

The hybrid PQ MLS (HPQMLS) combiner protocol runs two MLS sessions in parallel, synchronizing their group memberships. The two sessions are combined by exporting a secret from the PQ session and importing it as a Pre-Shared Key (PSK) into the traditional session. This combination process is mandatory for Commits of Add and Remove proposals in order to maintain synchronization between the sessions. However, it is optional for any other Commits (e.g. to allow for less computationally expensive traditional key rotations). Due to the higher computational costs and output sizes of PQ KEM (and signature) operations, it may be desirable to issue PQ combined (a.k.a. FULL) Commits less frequently than the traditional-only (a.k.a. PARTIAL) Commits. Since FULL Commits introduce PQ security into the MLS key schedule, the overall key schedule remains PQ-secure even when PARTIAL Commits are used. The FULL Commit rate establishes the post-quantum Post-Compromise Security (PCS) window, while the PARTIAL Commit rate can tighten the traditional PCS window even while maintaining PQ security more generally. The combiner protocol design treats both sessions as black-box interfaces so we only highlight operations requiring synchronizations in this document.

The default way to start a HPQMLS combined session is to create a PQ MLS session and then start a traditional MLS session with the exported PSK from the PQ session, as previously mentioned. Alternatively, a combined session can also be created after a traditional MLS session has already been running. This is done through creating a PQ MLS session with the same group members, sending a Welcome message containing the HPQMLSInfo struct in the GroupContext, and then making a FULL Commit as described in in the Section 4.1 section.

4.1. Commit Flow

Commits to proposals MAY be _PARTIAL_ or _FULL_. For a PARTIAL Commit, only the traditional session's epoch is updated following the Propose-Commit sequence from Section 12 of [RFC9420]. For a FULL Commit, a Commit is first applied to the PQ session and another Commit is applied to the traditional session using a PSK derived from the PQ session using the DeriveExtensionSecret and hpqmls_psk label (see Section 6.1). To ensure the correct PSK is imported into the traditional session, the sender includes information about the PSK in a PreSharedKey proposal for the traditional session's Commit list of

proposals. The information about the exported PSK is captured (shown '=' in the figures below for illustration purposes) by the PreSharedKeyID struct as detailed in [RFC9420]. Receivers process the PQ Commit to derive a new epoch in the PQ session and then the traditional Commit (which also includes the PSK proposal) to derive the new epoch in the traditional session.

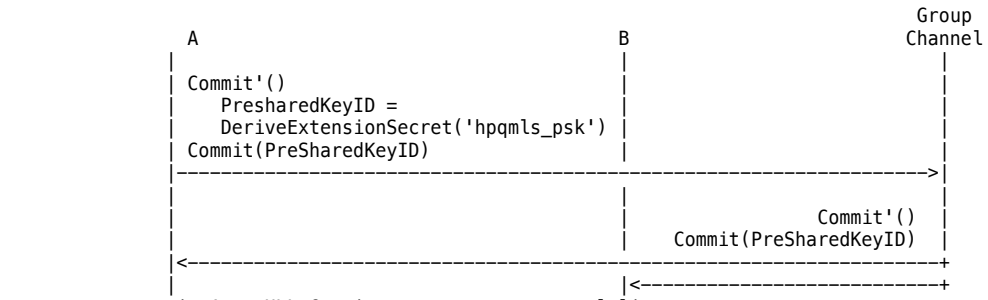
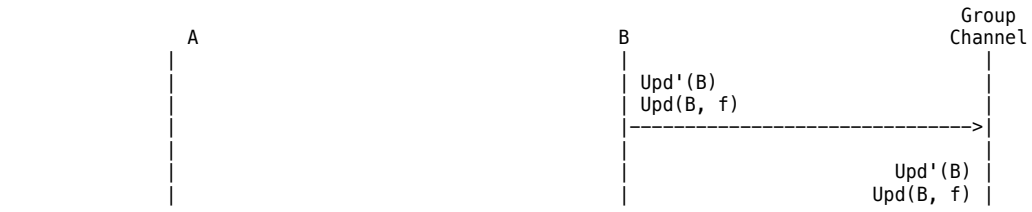


Fig 1a. FULL Commit to an empty proposal list.
Messages with ' are sent in the the PQ session.
PreSharedKeyID identifies a PSK exported from the PQ session in the new epoch following a Commit'(). The PreSharedKeyID is implicitly included in the commit in the classical session via the PreSharedKey Proposal.



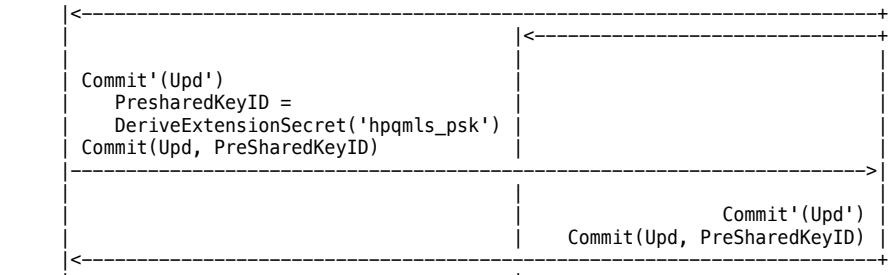
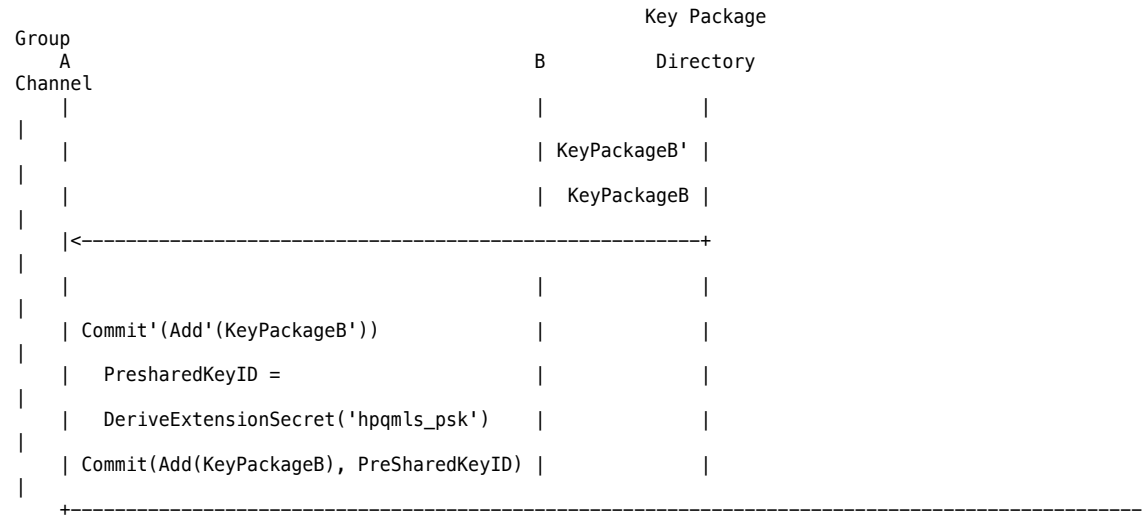


Fig 1b. FULL Commit to an Update proposal from Client B.
Messages with ' are sent in the the PQ session.

> REMARK: Fig 1b shows Client A accepting the update proposals from Client B as a FULL Commit. The flag f in the classical update proposal Upd(B, f) indicates B's intention for a FULL Commit to whomever Commits to its proposal.

4.2. Adding a User

User leaf nodes are first added to the PQ session following the sequence described in Section 3 of [RFC9420] except using PQ algorithms where HPKE algorithms exist. For example, a PQ-DSA signed PQ KeyPackage, i.e. containing a PQ public key, must first be published via the Distribution Service (DS). Then the associated Commit and Welcome messages will be sent and processed in the PQ session according to Section 12 of [RFC9420]. The same sequence is repeated in the standard session except following the FULL Commit combining sequence where a PreSharedKeyID proposal is additionally committed. The joiner MUST issue a FULL Commit as soon as possible after joining to achieve PCS.



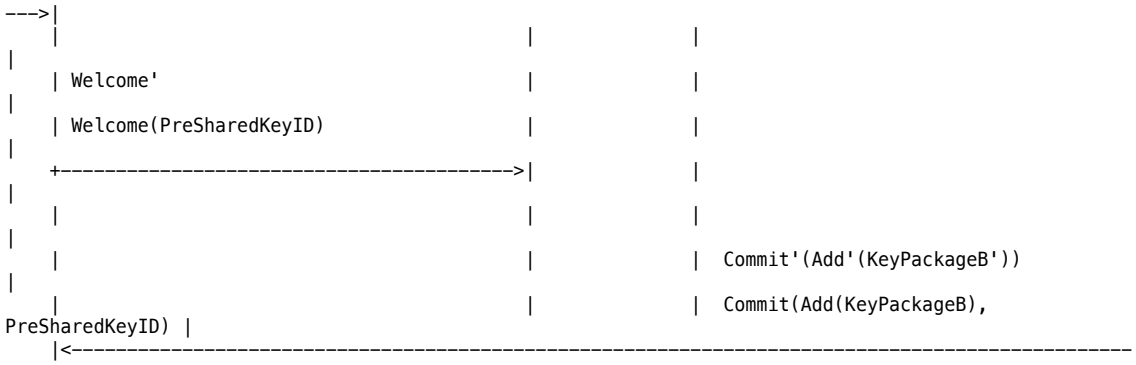


Figure 2:
Client A adds client B to the group.
Messages with ' come from the PQ session. Processing Welcome and Commit in the traditional sessio requires the PSK exported exported from the PQ session.

4.2.1. Welcome Message Validation

Since a client must join two sessions, the Welcome messages it receives to each session must indicate that it is not sufficient to join only one or the other. Therefore, the HPQMLSInfo struct indicating the GroupID and ciphersuites of the two sessions MUST be included in the Welcome message via serialization as a GroupContext Extension in order to validate joining the combined sessions. All members MUST verify group membership is consistent in both sessions after a join and the new member MUST issue a FULL Commit as described in Fig 1b.

4.2.2. External Joins

External joins are used by members who join a group without being explicitly added (via an Add-Commit sequence) by another existing member. The external user MUST join both the PQ session and the traditional session. As stated previously, the GroupInfo used to create the External Commit MUST contain the HPQMLSInfo struct. After joining, the new member MUST issue a FULL Commit as described in Fig 1b.

4.3. Removing a Group Member

User removals MUST be done in both PQ and traditional sessions followed by a FULL Commit Update as as described in Fig 1b. Members MUST verify group membership is consistent in both sessions after a removal.

4.4. Application Messages

HPQMLS combiner provides PQ security to the traditional MLS session. Application messages are therefore only sent in the traditional session using the encryption_secret provided by the key schedule of the traditional session according to Section 15 of [RFC9420].

5. Modes of Operation

Security needs vary by organizations and system-specific risk

tolerance and/or constraints. While this combiner protocol targets combining a PQ session and a traditional session the degree of PQ security may be tuned depending on the use-case: i.e., as PQ/T Confidentiality Only or both PQ/T Confidentiality and PQ/T Authenticity. For PQ/T Confidentiality Only, the PQ session MUST use a PQ KEM, while for PQ authenticity, the PQ session MUST use both a PQ KEM and a PQ DSA. The modes of operation are specified by the mode flag in HPQMLSInfo struct and are listed below.

5.1. PQ/T Confidentiality Only

The default mode of operation is PQ/T Confidentiality Only mode. This mode provides confidentiality and limited authenticity against quantum attackers. More precisely, it provides PQ authenticity against "outsiders", that is, against quantum attackers who do not have acces to (signature) secret keys of any group member. (Authenticity comes from the fact that the traditional session adds AEAD / MAC tags which are not available to outsiders with CRQC.) This mode does not prevent quantum impersonation attacks by other group members. That is, a group member with a CRQC can successfully impersonate another group member.

Note that an active attacker with access to a CRQC can become a group member by impersonating members in the moment they are added. As such, the authenticity guarantees outlined above only hold as long as the adversary is passive during the addition of new group members.

5.2. PQ/T Confidentiality + Authenticity

The elevated mode of operation is the PQ/T Confidentiality + Authenticity mode. Under a use environment of a cryptographically relevant quantum computer (CRQC), the threat model used in the default mode would be too weak and assurance about update authenticity is required. Recall that authenticity in MLS refers to three types of guarantees: 1) that messages were sent by a member of the group provided by the computed symmetric group key used in AEAD, 2) that key updates were performed by a valid member of the group, and 3) that a message was sent by a particular user (i.e. non-repudiation) provided by digital signatures on messages. While the symmetric group key used for AEAD in the traditional session remains protected from a CRQC adversary through the PSK from the PQ session, signatures would not be secure against forgery without using a PQ DSA to sign handshake messages nor are application messages assured to have non-repudiation against a CRQC adversary. Therefore, in the PQ/T Confidentiality + Authenticity mode, the PQ session MUST use a PQ DSA in addition to PQ KEM ciphersuites for handshake messages (the traditional session remains unchanged).

This version of PQ authenticity provides PQ authenticity to the PQ session's MLS commit messages, strengthening assurance for (1) and ensuring (2). These in turn provide PQ assurance for the key schedule from which application keys are derived in the traditional session. Application keys are used in an AEAD for protection of MLS application messages and thereby inherit the PQ security. However, it should be noted that PQ non-repudation security for application messages as described by (3) is not achieved by this mode. Achieving PQ non-repudiation on application messages would require hybrid signatures in the traditional session, with considerations to options

described in [I-D.hale-pquip-hybrid-signature-spectrums].

6. Extension Requirements to MLS

The HPQMLSInfo struct contains characterizing information to signal to users that they are participating in a hybrid session. This is necessary both functionally to allow for group synchronization and as a security measure to prevent downgrading attacks to coax users into parcipating in just one of the two sessions. The group_id, cipher_suite, and epoch from both sessions (t for the traditional session and pq for the PQ session) are used as bookkeeping values to validate and synchronize group operations. The mode is a boolean value: 0 for the default PQ/T Confidentiality Only mode and 1 for the PQ/T Confidentiality + Authenticity mode.

The HPQMLSInfo struct conforms to the Safe Extensions API (see [I-D.ietf-mls-extensions]). Recall that an extension is called _safe_ if it does not modify base MLS protocol or other MLS extensions beyond using components of the Safe Extension API. This allows security analysis of our HPQMLS Combiner protocol in isolation of the security guarantees of the base MLS protocol to enable composability of guarantees. The HPMLSInfo extension struct SHALL be in the following format:

```
struct{
    ExtensionType HPQMLS;
    opaque extension_data<V>;
    } ExtensionContent;

struct{
    opaque t_session_group_id<V>;
    opaque PQ_session_group_id<V>;
    bool mode;
    CipherSuite t_cipher_suite;
    CipherSuite pq_cipher_suite;
    uint64 t_epoch;
    uint64 pq_epoch;
} HPQMLSInfo
```

6.1. Key Schedule

The hpqmls_psk exporter key derived in the PQ session MUST be derived in accordance with the Safe Extensions API guidance (see Exporting Secrets in [I-D.ietf-mls-extensions]). In particular, it SHALL NOT use the extension_secret and MUST be derived from only the epoch_secret from the key schedule in [RFC9420]. This is to ensure forward secrecy guarantees (see Section 8).

Even though the hpqmls_psk PSK is not sent over the wire, members of the HPQMLS session must agree on the value of which PSK to use. In alignment with the Safe Extensions API policy for PSKs, HPQMLS PSKs used SHALL set PSKType = 3 and extension_type = HPQMLS (see Section 2.1.6 Pre-Shared Keys in [I-D.ietf-mls-extensions]).

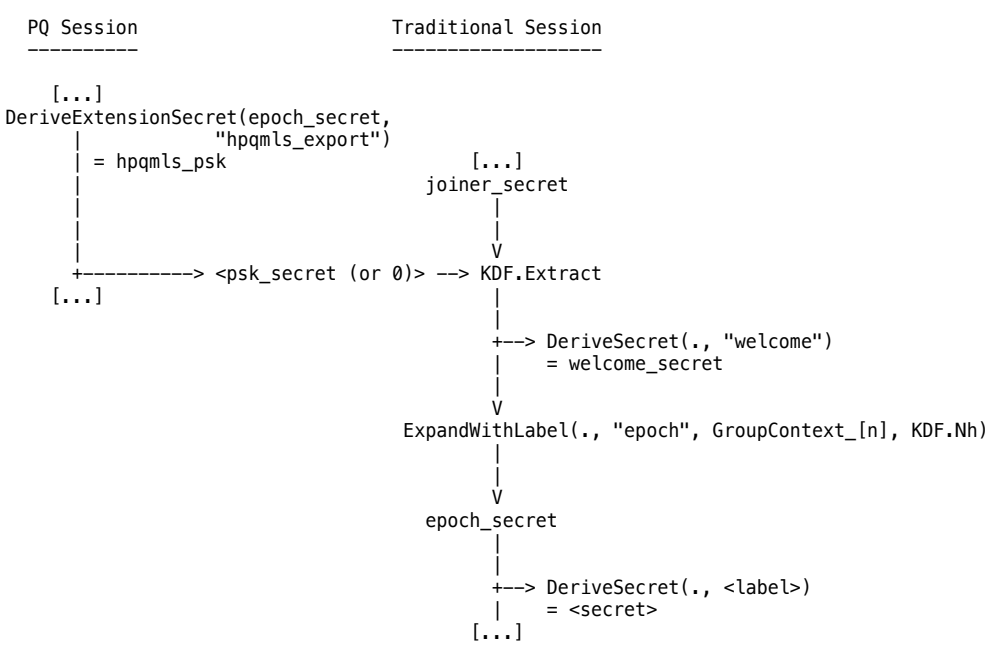


Fig 3: The hpqmls_psk of the PQ session is injected into the key schedule of the traditional session using the safe extensions API DeriveExtensionSecret.

7. Cryptographic Objects

7.1. Cipher Suites

There are no changes to _how_ cipher suites are used to perform group key computations from RFC9420 (https://www.rfc-editor.org/rfc/rfc9420#name-cipher-suites). However, the choice of _which_ primitives are used by the traditional and PQ subsessions must be explicitly stated by the CipherSuite objects within HPQMLSInfo. So long as the traditional session only uses classical primitives and the PQ session uses PQ primitives for KEM, a HPQMLS session is valid. Specifically, the PQ primitives for HPQMLS must be 'pure' (fully) PQ: PQ cost is already being amoritized at the protocol level so allowing hybrid PQ cipher suites to be used in the PQ session only adds extra overhead and complexity. Furthermore, the pq_cipher_suite may contain a classical digital signature algorithm used if mode is set to 0 (PQ Confidentiality-Only) but MUST be fully PQ if mode is set to

1 (PQ Confidentiality+Authenticity). These cipher suite combinations and modes MUST not be toggled or modified after a HPQMLS session has commenced. Clients MUST reject a HPQMLS session with invalid or

duplicate cipher suites (e.g. two traditional cipher suites).

7.1.1. Key Encapsulation Mechanism

For HPQMLS sessions, the PQ subsession MUST use a Key Encapsulation Mechanism (KEM) that is standardized by NIST for post-quantum cryptography. Specifically, only KEMs that have been selected and published by NIST as part of their post-quantum cryptography standardization process (e.g., ML-KEM as specified in FIPS 203) are permitted for use in the PQ session. The use of experimental, non-standardized, or hybrid KEMs in the PQ session is NOT RECOMMENDED and MUST be rejected by compliant clients. This requirement ensures interoperability and a consistent security baseline across all HPQMLS deployments.

7.1.2. Signing

For HPQMLS sessions, the choice of digital signature algorithm in the PQ subsession depends on the selected mode of operation. If the mode is set to 1 (PQ Confidentiality+Authenticity), the PQ session MUST use a digital signature algorithm that is standardized by NIST for post-quantum cryptography, such as ML-DSA as specified in FIPS 204. The use of experimental, non-standardized, or hybrid signature algorithms in the PQ session is NOT RECOMMENDED and MUST be rejected by compliant clients in this mode. If the mode is set to 0 (PQ Confidentiality-Only), the PQ session MAY use a classical digital signature algorithm, but the use of a NIST-standardized PQ signature algorithm is RECOMMENDED. These requirements ensure that the authenticity guarantees of HPQMLS sessions are aligned with the intended security level and provide a consistent baseline for interoperability and security across deployments.

8. Security Considerations

8.1. FULL Commit Frequency

So long as the FULL Commit flow is followed for group administration actions, PQ security is extended to the traditional session. Therefore, FULL Commits can occur as frequently or infrequently as desired by any given security policy. This results in a flexible and efficient use of compute, storage, and bandwidth resources for the host by mainly calling partial updates on the traditional MLS session, given that the group membership is stable. Thus, our protocol provides PQ security and can maintain a tighter PCS window against traditional attackers as well as forward secrecy window

against traditional or quantum attackers with lower overhead when compared to running a single MLS session that only uses PQ KEMs or PQ KEM/DSAs. Furthermore, the PQ PCS window against quantum attackers can be selected based on an application and even variable over time, ranging from e.g. a single FULL Commit in PQ/T Confidentiality Only mode followed by PARTIAL Commits from that point onwards (enabling general PQ/traditional confidentiality, traditional update authenticity, traditional PCS, and PQ/traditional forward secrecy) to frequent FULL Commits in the same mode (enabling general PQ/traditional confidentiality, traditional update authenticity, PQ/traditional PCS, and PQ/traditional forward secrecy). In PQ/T Confidentiality + Authenticity mode with frequent FULL Commits, the latter case would enable general PQ/traditional confidentiality, PQ/traditional update authenticity, PQ/traditional PCS, and PQ/traditional forward secrecy.

8.2. Attacks on Non-Repudiation

While PQ message integrity is provided by the symmetric key used in AEAD, attacks on non-repudiation (e.g., source forgery) on application messages may still be possible by a CRQC adversary since only traditional signatures on used after the AEAD. However, in terms of group key agreement, this is insufficient to mount anything more than a denial-of-service attack (e.g. via group state desynchronization). In terms of application messages, a traditional DSA signature may be forged by an external CRQC adversary, but the content (including sender information) is still protected by AEAD which uses the symmetric group key. Thus, an external CRQC adversary can only conduct a false-framing attack, where group members are assured of the authenticity of a message being sent by a group member for the adversary has changed the signature to imply a different sender; it would require an insider CRQC adversary to actually mount a masquerading or forgery attack, which is beyond the scope of this protocol.

If this is a concern, hybrid PQ DSAs can be used in the traditional session to sign application messages. Since this would negate much of the efficiency gains from using this protocol and denial-of-service attacks can be achieve through more expeditious means, such a option is not considered here.

8.3. Forward Secrecy

Recall that one of the ways MLS achieves forward secrecy is by deleting security sensitive values after they are consumed (e.g. to encrypt or derive other keys/nonces) and the key schedule has entered a new epoch. For example, values such as the init_secret or epoch_secret are deleted at the _start_ of a new epoch. If the MLS exporter_secret or the extension_secret from the PQ session is used directly as a PSK for the traditional session, against the requirements set above, then there is a potential scenario in which an adversary can break forward secrecy because these keys are derived _during_ an epoch and are not deleted. Therefore, the hpqmls_psk MUST be derived from the epoch_secret created at the _start_ of an epoch from the PQ session (see Figure 3) to ensure forward secrecy.

8.4. Transport Security

Recommendations for preventing denial-of-service attacks or restricting transmitted messages are inherited from MLS.

9. IANA Considerations

The MLS sessions combined by this protocol conform to the IANA registries listed for MLS [RFC9420].

10. Normative References

[I-D.hale-pquip-hybrid-signature-spectrums]
Bindel, N., Hale, B., Connolly, D., and F. D, "Hybrid signature spectrums", Work in Progress, Internet-Draft, draft-hale-pquip-hybrid-signature-spectrums-04, 21 March

2024, <https://datatracker.ietf.org/doc/html/draft-hale-pquip-hybrid-signature-spectrums-04>.

[I-D.ietf-mls-extensions]
Robert, R., "The Messaging Layer Security (MLS) Extensions", Work in Progress, Internet-Draft, draft-ietf-mls-extensions-06, 19 February 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-mls-extensions-06>.

[I-D.ietf-pquip-pqt-hybrid-terminology]
D, F., P, M., and B. Hale, "Terminology for Post-Quantum Traditional Hybrid Schemes", Work in Progress, Internet-Draft, draft-ietf-pquip-pqt-hybrid-terminology-06, 10 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-pquip-pqt-hybrid-terminology-06>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <https://www.rfc-editor.org/rfc/rfc5869>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9180] Barnes, R., Bhargavan, K., Lipp, B., and C. Wood, "Hybrid Public Key Encryption", RFC 9180, DOI 10.17487/RFC9180, February 2022, <https://www.rfc-editor.org/rfc/rfc9180>.

[RFC9420] Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., and K. Cohn-Gordon, "The Messaging Layer Security (MLS) Protocol", RFC 9420, DOI 10.17487/RFC9420, July 2023, <https://www.rfc-editor.org/rfc/rfc9420>.

Acknowledgments

Contributors

Konrad Kohbrok Phoenix R&D Email: konrad.kohbrok@datashrine.de

Authors' Addresses

Joël Alwen
AWS
Email: alwenjo@amazon.com

Britta Hale
Naval Postgraduate School
Email: britta.hale@nps.edu

Marta Mularczyk
AWS
Email: mulmarta@amazon.ch

Xisen Tian
Naval Postgraduate School

Email: xisen.tian1@nps.edu

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] "Commercial National Security Algorithm Suite 2.0," NSA PP-22-1338 Ver. 1.0, Fort Meade, MD, USA, Sep. 2022 [Online]. Available: https://media.defense.gov/2025/May/30/2003728741/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS.PDF

[2] C. Jaikaran, "Salt typhoon hacks of telecommunications companies and federal response implications," CRS Report No. IF12798, Washington, DC, USA, Oct. 2025 [Online]. Available: https://www.congress.gov/crs-product/IF12798

[3] R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon, *The Messaging Layer Security (MLS) Protocol*, IETF Request for Comments 9420, Jul. 2023 [Online]. Available: https://www.rfc-editor.org/info/rfc9420

[4] S. Burleigh, K. Fall, and E. J. Birrane, *Bundle Protocol Version 7*, IETF Request for Comments 9171, Jan. 2022 [Online]. Available: https://www.rfc-editor.org/info/rfc9171

[5] I. T. L. Computer Security Division. "Post-Quantum Cryptography FIPS Approved | CSRC." https://csrc.nist.gov/News/2024/postquantum-cryptography-fips-approved. Aug. 2024.

[6] M. Mosca, "Cybersecurity in an era with quantum computers: will we be ready?" Cryptology ePrint Archive, Paper 2015/1075, 2015 [Online]. Available: https://eprint.iacr.org/2015/1075

[7] B. Dowling, B. Hale, X. Tian, and B. Wimalasiri, "Securing BPSec Against Arbitrary Packet Dropping," Internet Engineering Task Force, Internet-Draft draft-tian-dtn-sbam-00, Jul. 2025. Work in Progress [Online]. Available: https://datatracker.ietf.org/doc/draft-tian-dtn-sbam/00/

[8] X. Tian, B. Hale, M. Mularczyk, and J. Alwen, "Amortized PQ MLS Combiner," Internet Engineering Task Force, Internet-Draft draft-ietf-mls-combiner-02, Oct. 2025. Work in Progress [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-mls-combiner/02/

[9] E. J. Birrane and K. McKeever, *Bundle Protocol Security (BPSec)*, IETF Request for Comments 9172, Jan. 2022 [Online]. Available: https://www.rfc-editor.org/info/rfc9172

[10] E. Birrane, *Securing Delay-Tolerant Networks with BPSec*. Wiley, 2023 [Online]. Available: https://ieeexplore.ieee.org/servlet/opac?bknumber=10015530

[11] E. J. Birrane, A. White, and S. Heiner, *Default Security Contexts for Bundle Protocol Security (BPSec)*, IETF Request for Comments 9173, Jan. 2022 [Online]. Available: https://www.rfc-editor.org/info/rfc9173

[12] E. J. Birrane and K. McKeever, "Bundle Protocol Security (BPSec)," RFC 9172, Jan. 2022 [Online]. Available: https://doi.org/10.17487/RFC9172

[13] S. Frankel and S. Krishnan, "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap," RFC 6071, Feb. 2011 [Online]. Available: https://datatracker.ietf.org/doc/html/rfc6071

[14] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Internet Engineering Task Force (IETF). RFC 8446, Rep., 2018.

[15] F. Z. Benhamida, A. Bouabdellah, and Y. Challal, "Using delay tolerant network for the Internet of Things: Opportunities and challenges," in *2017 8th International Conference on Information and Communication Systems (ICICS)*, 2017, pp. 252–257 [Online]. Available: https://doi.org/10.1109/IACS.2017.7921980

[16] C. G. Manning, "Frequently Asked Questions," 2023 [Online]. Available: https://www.nasa.gov/technology/space-comms/delay-disruption-tolerant-networking-faq/

[17] M. Bellare, T. Kohno, and C. Namprempre, "Authenticated encryption in SSH: provably fixing the SSH binary packet protocol," in *Proceedings of the 9th ACM conference on Computer and Communications Security (CCS)*, 2002, pp. 1–11.

[18] P. Rogaway, "Authenticated-encryption with associated-data," in *Proceedings of the 9th ACM Conference on Computer and Communications Security* (CCS '02). Association for Computing Machinery, 2002, p. 98–107 [Online]. Available: https://doi.org/10.1145/586110.586125

[19] M. Fischlin, F. Günther, G. A. Marson, and K. G. Paterson, "Data is a stream: Security of stream-based channels," in *Advances in Cryptology–CRYPTO 2015, Proceedings, Part II 35*. Springer, 2015, pp. 545–564.

[20] C. Boyd, B. Hale, S. F. Mjølsnes, and D. Stebila, "From Stateless to Stateful: Generic Authentication and Authenticated Encryption Constructions with Application to TLS," in *Proceedings of the RSA Conference on Topics in Cryptology - CT-RSA 2016 - Volume 9610*, 2016, p. 55–71 [Online]. Available: https://doi.org/10.1007/978-3-319-29485-8_4

[21] F. Günther and S. Mazaheri, "A Formal Treatment of Multi-key Channels," in *Advances in Cryptology–CRYPTO 2017 Proceedings, Part III 37*. Springer, 2017, pp. 587–618.

[22] M. Fischlin, F. Günther, and C. Janson, "Robust Channels: Handling Unreliable Networks in the Record Layers of QUIC and DTLS 1.3," *J. Cryptol.*, vol. 37, no. 2, jan 2024 [Online]. Available: https://doi.org/10.1007/s00145-023-09489-9

[23] C. Boyd and B. Hale, "Secure Channels and Termination: The Last Word on TLS," in *Latincrypt*, 2017 [Online]. Available: https://api.semanticscholar.org/CorpusID:3648242

[24] C. Badertscher, C. Matt, U. Maurer, P. Rogaway, and B. Tackmann, "Augmented Secure Channels and the Goal of the TLS 1.3 Record Layer," in *Proceedings of the 9th International Conference on Provable Security - Volume 9451* (ProvSec 2015). Springer-Verlag, 2015, p. 85–104 [Online]. Available: https://doi.org/10.1007/978-3-319-26059-4_5

[25] P. Syverson, R. Dingledine, and N. Mathewson, "Tor: The Second Generation Onion Router," in *Usenix Security*. USENIX Association Berkeley, CA, 2004, pp. 303–320.

[26] S. Burleigh, K. Fall, and E. J. Birrane, "Bundle Protocol Version 7," RFC 9171, Jan. 2022 [Online]. Available: https://doi.org/10.17487/RFC9171

[27] L. Torgerson *et al.*, *Delay-Tolerant Networking Architecture*, IETF Request for Comments 4838, Apr. 2007 [Online]. Available: https://www.rfc-editor.org/info/rfc4838

[28] P. Rogaway and T. Shrimpton, "A Provable-Security Treatment of the Key-Wrap Problem," in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2006, pp. 373–390.

[29] Consultative Committee for Space Data Systems, "Overview of Space Communications Protocols," Washington DC, USA, Rep. CCSDS 130.0-G-4, Apr. 2023 [Online]. Available: https://public.ccsds.org/Pubs/130x0g4e1.pdf

[30] The Consultative Committee for Space Data Systems. Recommended Standard, "Space Packet Protocol," Rep. CSDS 133.0-B-2, 2020 [Online]. Available: https://public.ccsds.org/Pubs/133x0b2e1.pdf

[31] The Consultative Committee for Space Data Systems. Recommended Standard., "CCSDS File Delivery Protocol (CFDP)," Rep. CSDS 727.0-B-5, 2020 [Online]. Available: https://public.ccsds.org/Pubs/727x0b5.pdf

[32] C. G. Manning, "Delay/Disruption Tolerant Networking Overview," 2023 [Online]. Available: https://www.nasa.gov/technology/space-comms/delay-disruption-tolerant-networking-overview/

[33] S. A. Menesidou, V. Katos, and G. Kambourakis, "Cryptographic Key Management in Delay Tolerant Networks: A Survey," *Future Internet*, vol. 9, no. 3, p. 26, 2017.

[34] N. Asokan, K. Kostiainen, P. Ginzboorg, J. Ott, and C. Luo, "Applicability of identity-based cryptography for disruption-tolerant networking," in *Proceedings of the 1st International MobiSys Workshop on Mobile Opportunistic Networking* (MobiOpp '07). Association for Computing Machinery, 2007, p. 52–56 [Online]. Available: https://doi.org/10.1145/1247694.1247705

[35] A. Kate, G. M. Zaverucha, and U. Hengartner, "Anonymity and security in delay tolerant networks," in *2007 Third International Conference on Security and Privacy in Communications Networks and the Workshops - SecureComm 2007*, 2007, pp. 504–513 [Online]. Available: https://doi.org/10.1109/SECCOM.2007.4550373

[36] S. Rüsch, D. Schürmann, R. Kapitza, and L. Wolf, "Forward Secure Delay-Tolerant Networking," in *Proceedings of the 12th Workshop on Challenged Networks* (CHANTS '17). Association for Computing Machinery, 2017, p. 7–12 [Online]. Available: https://doi.org/10.1145/3124087.3124094

[37] X. Lv, Y. Mu, and H. Li, "Non-Interactive Key Establishment for Bundle Security Protocol of Space DTNs," *IEEE Transactions on Information Forensics and Security*, vol. 9, no. 1, pp. 5–13, 2014 [Online]. Available: https://doi.org/10.1109/TIFS.2013.2289993

[38] N. Ahmad, H. Cruickshank, Z. Sun, and M. Asif, "Pseudonymised communication in delay tolerant networks," in *2011 Ninth Annual International Conference on Privacy, Security and Trust*, 2011, pp. 1–6 [Online]. Available: https://doi.org/10.1109/PST.2011.5971956

[39] J. Zhou, M. Song, J. Song, X.-W. Zhou, and L. Sun, "Autonomic Group Key Management in Deep Space DTN," *Wirel. Pers. Commun.*, vol. 77, no. 1, p. 269–287, July 2014 [Online]. Available: https://doi.org/10.1007/s11277-013-1505-1

[40] S. C. Burleigh, S. Farrell, and M. Ramadas, "Licklider Transmission Protocol - Specification," RFC 5326, Sep. 2008 [Online]. Available: https://doi.org/10.17487/RFC5326

[41] B. Sipos, "DTN Bundle Protocol Security (BPSec) COSE Context," Internet Engineering Task Force, Internet-Draft draft-ietf-dtn-bpsec-cose-04, July 2024. Work in Progress [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-dtn-bpsec-cose/04/

[42] R. Housley and J. Schaad, "Advanced Encryption Standard (AES) Key Wrap Algorithm," RFC 3394, Oct. 2002 [Online]. Available: https://doi.org/10.17487/RFC3394

[43] M. Dworkin, "Request for Review of Key Wrap Algorithms," Cryptology ePrint Archive, Paper 2004/340, 2004. https://eprint.iacr.org/2004/340 [Online]. Available: https://eprint.iacr.org/2004/340

[44] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis, "Security analysis and improvements for the ietf mls standard for group messaging," pp. 248–277, 08 2020 [Online]. Available: https://doi.org/10.1007/978-3-030-56784-2_9

[45] Joël, K. Kohbrok, and R. Robert, "MLS Virtual Clients," Internet Engineering Task Force, Internet-Draft draft-kohbrok-mls-virtual-clients-00, Dec. 2023. Work in Progress [Online]. Available: https://datatracker.ietf.org/doc/draft-kohbrok-mls-virtual-clients/00/

[46] B. Hale and C. Komlo, "On end-to-end encryption," Cryptology ePrint Archive, Paper 2022/449, 2022. https://eprint.iacr.org/2022/449 [Online]. Available: https://eprint.iacr.org/2022/449

[47] M. Marlinspike, "Advanced cryptographic ratcheting," https://signal.org/blog/advanced-ratcheting/, Nov. 2013.

[48] M. Marlinspike and T. Perrin, "The Signal Protocol," Working Draft, Signal.org, Rep., November 2016 [Online]. Available: https://signal.org/docs/specifications/x3dh/

[49] C.-E. Bogos, R. Mocanu, and E. Simion, "A security analysis comparison between Signal, WhatsApp and Telegram," Cryptology ePrint Archive, Paper 2023/071, 2023. https://eprint.iacr.org/2023/071 [Online]. Available: https://eprint.iacr.org/2023/071

[50] M. Marlinspike, "Facebook Messenger deploys Signal Protocol for end-to-end encryption," July 2016 [Online]. Available: https://signal.org/blog/facebook-messenger/

[51] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the signal messaging protocol," in *2017 IEEE European Symposium on Security and Privacy, (Euro S&P)*, 2017, pp. 451–466.

[52] B. Poettering and P. Rösler, "Asynchronous Ratcheted Key Exchange," in *CRYPTO, '18*, 2018.

[53] B. Dowling and B. Hale, "Authenticated Continuous Key Agreement: Active MitM Detection and Prevention," Cryptology ePrint Archive, Paper 2023/228, 2023 [Online]. Available: https://eprint.iacr.org/2023/228

[54] D. Jost, U. Maurer, and M. Mularczyk, "Efficient ratcheting: Almost-optimal guarantees for secure messaging," in *Advances in Cryptology - EUROCRYPT 2019* (Lecture Notes in Computer Science). Springer, 2019, vol. 11476, pp. 159–188.

[55] P. Rösler, C. Mainka, and J. Schwenk, "More is less: On the end-to-end security of group chats in Signal, WhatsApp, and Threema," in *2018 IEEE European Symposium on Security and Privacy, (Euro S&P)*, 2018, pp. 415–429.

[56] B. Dowling and B. Hale, "Secure Messaging Authentication against Active Man-in-the-Middle Attacks," in *2021 IEEE European Symposium on Security and Privacy, (Euro S&P)*, 2021.

[57] R. Barnes, J. Millican, E. Omara, K. Cohn-Gordon, and R. Robert, "The Messaging Layer Security (MLS) Protocol," Working Draft, IETF Secretariat, Internet-Draft draft-ietf-mls-protocol-20, May 2023. https://datatracker.ietf.org/doc/draft-ietf-mls-protocol/ [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-mls-protocol/

[58] C. Brzuska, E. Cornelissen, and K. Kohbrok, "Cryptographic security of the mls rfc, draft 11," *IACR Cryptol. ePrint Arch.*, vol. 2021, p. 137, 2021.

[59] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis, "Security analysis and improvements for the ietf mls standard for group messaging," in *Advances in Cryptology – CRYPTO 2020*, D. Micciancio and T. Ristenpart, Eds. Cham: Springer International Publishing, 2020, pp. 248–277.

[60] K. K. Cas Cremers, Britta Hale, "The complexities of healing in secure group messaging: Why cross-group effects matter," *USENIX*, pp. 1847–1864, 2021.

[61] Alwen, Joël and Sandro Coretti and Yevgeniy Dodis and Yiannis Tselekounis, "Modular Design of Secure Group Messaging Protocols and the Security of MLS," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (CCS '21), 2021, p. 1463–1483 [Online]. Available: https://doi.org/10.1145/3460120.3484820

[62] J. Alwen, D. Jost, and M. Mularczyk, "On the insider security of mls," in *Advances in Cryptology – CRYPTO 2022*, Y. Dodis and T. Shrimpton, Eds. Cham: Springer Nature Switzerland, 2022, pp. 34–68.

[63] I. Levy and C. Robinson, "Principles for a more informed exceptional access debate," Lawfare, Nov. 2019 [Online]. Available: https://www.lawfareblog.com/principles-more-informed-exceptional-access-debate

[64] S. Meredith, "Apple, Google and WhatsApp condemn UK proposal to eavesdrop on encrypted messages," CNBC, May 2019 [Online]. Available: https://www.cnbc.com/2019/05/30/apple-google-and-whatsapp-condemn-gchq-ghost-proposal.html

[65] C. Chevalier, G. Lebrun, and A. Martinelli, "Quarantined-treekem: a continuous group key agreement for mls, secure in presence of inactive users," Cryptology ePrint Archive, Paper 2023/1903, 2023. https://eprint.iacr.org/2023/1903 [Online]. Available: https://eprint.iacr.org/2023/1903

[66] J. Alwen, S. Coretti, and Y. Dodis, "The double ratchet: Security notions, proofs, and modularization for the signal protocol," in *Advances in Cryptology – EUROCRYPT 2019*, Y. Ishai and V. Rijmen, Eds. Cham: Springer International Publishing, 2019, pp. 129–158.

[67] J. Alwen, S. Coretti, and Y. Dodis, "The double ratchet: Security notions, proofs, and modularization for the signal protocol," Cryptology ePrint Archive, Paper 2018/1037, 2018. https://eprint.iacr.org/2018/1037 [Online]. Available: https://eprint.iacr.org/2018/1037

[68] H. Krawczyk, "Cryptographic extraction and key derivation: The hkdf scheme," in *Advances in Cryptology – CRYPTO 2010*, T. Rabin, Ed. Springer Berlin Heidelberg, 2010, pp. 631–648.

[69] K. Kohbrok, "Subject: [mls] improve fs granularity at a cost," MLS Mailing List, Jan 2019 [Online]. Available: https://mailarchive.ietf.org/arch/msg/mls/WRdXVr8iUwibaQu0tH6sDnqU1no

[70] "Space System Protection Standard." NASA Technical Standard. NASA-STD-1006A https://explorers.larc.nasa.gov/2023ESE/pdf_files/2022-07-15-NASA-STD-1006A-Approved.pdf.

[71] "Secure satellite communications encryption unit." L3Harris Fast. Forward. https://www.l3harris.com/all-capabilities/secure-satellite-communications-encryption-unit-cdu-200 [Online]. Available: https://www.l3harris.com/all-capabilities/secure-satellite-communications-encryption-unit-cdu-200

[72] M. Maidenberg, "The SpaceX Advantage That Rivals Are Trying to Emulate," *The Wall Street J.*, Oct. 2024. Section: Business [Online]. Available: https://www.wsj.com/science/space-astronomy/the-spacex-advantage-that-rivals-are-trying-to-emulate-ebde2568

[73] "High-Speed Internet Around the World." Starlink https://www.starlink.com/us.

[74] F. M. Asrar *et al.*, "Can space-based technologies help manage and prevent pandemics?" *Nature Medicine*, vol. 27, no. 9, pp. 1489–1490, Sep. 2021 [Online]. Available: https://doi.org/10.1038/s41591-021-01485-5

[75] C. Cookson, "Will space tourism really lift off?" *Financial Times*, Oct. 2024. Available at https://www.ft.com/content/4ec9715d-c151-4b91-ba1f-5c9bfb3363f7.

[76] T. Fernholz. "AstroForge Picks Up First Commercial Deep Space License." Oct. 2024. Available at https://payloadspace.com/astroforge-picks-up-first-commercial-deep-space-license/.

[77] J.-P. Menez, "Exploring the Business of Space | Yale Insights," Oct. 2022. Available at https://insights.som.yale.edu/insights/exploring-the-business-of-space.

[78] A. Hughes, "Satellites in orbit can now be hacked. Here's why that's really (really) bad," *BBC Sci. Focus*, Jun. 2024 [Online]. Available: https://www.sciencefocus.com/space/satellites-cyber-attack

[79] C. C. for Space Data Systems, "Space Data Link Security Protocol–Extended Procedures," vol. CCSDS 355.1-B-1, no. 1, pp. 3–1, Feb. 2020.

[80] Consultative Committee for Space Data Systems, "Security Architecture for Space Data Systems," Washington, DC, USA, Recommended Practice, 2012 [Online]. Available: https://public.ccsds.org/Pubs/351x0m1.pdf

[81] C. Huitema and M. Blanchet, "QUIC in Space," Internet Engineering Task Force, Internet-Draft draft-huitema-quic-in-space-00, Sep. 2023 [Online]. Available: https://datatracker.ietf.org/doc/draft-huitema-quic-in-space/00/

[82] M. Troncoso and B. Hale, "The Bluetooth CYBORG: Analysis of the Full Human-Machine Passkey Entry AKE Protocol," in *Proc. 2021 Netw. and Distributed System Security Symposium*. Virtual: Internet Soc., 2021 [Online]. Available: https://doi.org/10.14722/ndss.2021.24401

[83] L. Li, P. Podder, and E. Hoque, "A formal security analysis of ZigBee (1.0 and 3.0)," in *Proc. of the 7th Symp. on Hot Topics in the Sci. of Security* (HotSoS '20). New York, NY, USA: Assoc. for Comput. Machinery, 2020 [Online]. Available: https://doi.org/10.1145/3384217.3385617

[84] "Zigbee Specification," Davis, CA USA, Mar. 2023. Available at https://csa-iot.org/wp-content/uploads/2023/04/05-3474-23-csg-zigbee-specification-compressed.pdf.

[85] "Bluetooth Core Specification," Aug. 2024. Available at https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=588232.

[86] A. Sultan, "5G System Overview," Aug. 2022 [Online]. Available: https://www.3gpp.org/technologies/5g-system-overview

[87] M. Joras and Y. Chi. "How Facebook is bringing QUIC to billions." Oct. 2020 [Online]. Available: https://engineering.fb.com/2020/10/21/networking-traffic/how-facebook-is-bringing-quic-to-billions/

[88] "GitHub - youtube/cobalt at 25.lts.10" [Online]. Available: https://github.com/youtube/cobalt

[89] B. Hale, "Cryptography, chaos, and the cutting edge," CROSSFYRE, Dec. 2021.

[90] B. Hale, "The future of comms," CRUSER, 2023.

[91] B. Hale, "UxS communication links: Planning for the future," CRUSER, Feb. 2024.

[92] C. Bader, "On Requirements and Concepts for TT&C Link Key Management," Mar 2024 [Online]. Available: https://doi.org/10.14722/spacesec.2024.23053

[93] D. J. Israel *et al.*, "Lunanet: a flexible and extensible lunar exploration communications and navigation infrastructure," in *2020 IEEE Aerospace Conference*, 2020, pp. 1–14 [Online]. Available: https://doi.org/10.1109/AERO47225.2020.9172509

[94] C. G. Manning, "Frequently Asked Questions - NASA," Sep. 2023. Section: Communicating and Navigating with Missions [Online]. Available: https://www.nasa.gov/technology/space-comms/delay-disruption-tolerant-networking-faq/

[95] Consultative Committee for Space Data Systems, "Rationale, Scenarios, and Requirements for DTN in Space," vol. CCSDS 734.0-G-1, no. 1, pp. 5–7, 2010.

[96] D. Bhattacherjee and A. Singla, "Network topology design at 27,000 km/hour," in *Proc. of the 15th Int. Conf. on Emerging Netw. Experiments And Technologies* (CoNEXT '19). New York, NY, USA: Assoc. for Comput. Machinery, 2019, pp. 341–354 [Online]. Available: https://doi.org/10.1145/3359989.3365407

[97] I. del Portillo, B. G. Cameron, and E. F. Crawley, "A technical comparison of three low earth orbit satellite constellation systems to provide global broadband," *Acta Astronautica*, vol. 159, pp. 123–135, 2019 [Online]. Available: https://doi.org/https://doi.org/10.1016/j.actaastro.2019.03.040

[98] S. Floyd, *Congestion Control Principles*, IETF Request for Comments 2914, Sep. 2000 [Online]. Available: https://datatracker.ietf.org/doc/rfc2914

[99] P. Muri and J. McNair, "A performance comparison of dtn protocols for high delay optical channels," in *2013 IEEE Wireless Commun. and Netw. Conf. Workshops (WCNCW)*, 2013, pp. 183–188 [Online]. Available: https://doi.org/10.1109/WCNCW.2013.6533337

[100] National Aeronautics and Space Administration, "Mars Communications Disruption and Delay," White Paper, 2023 [Online]. Available: https://www.lpi.usra.edu/lunar/strategies/resources/M2M-ACR2023_MarsCommunicationDisruptionDelay.pdf

[101] M. Parisi, T. Panontin, S.-C. Wu, K. Mctigue, and A. Vera, "Effects of Communication Delay on Human Spaceflight Missions," 2023 [Online]. Available: https://doi.org/10.54941/ahfe1003920

[102] National Aeronautics and Space Administration and European Space Agency, "Draft Lunanet Interoperability Specification," Rep., Aug. 2023 [Online]. Available: https://www.nasa.gov/wp-content/uploads/2023/09/lunanet-interoperability-specification-v5-draft.pdf?emrc=6f4483

[103] B. Sipos, "DTN Bundle Protocol Security (BPSec) COSE Context," Internet Engineering Task Force, Internet-Draft draft-ietf-dtn-bpsec-cose-05, Nov. 2024 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-dtn-bpsec-cose/05/

[104] E. Rescorla, *The Transport Layer Security (TLS) Protocol Version 1.3*, IETF Request for Comments 8446, Aug. 2018 [Online]. Available: https://www.rfc-editor.org/info/rfc8446

[105] J. Iyengar and M. Thomson, *QUIC: A UDP-Based Multiplexed and Secure Transport*, IETF Request for Comments 9000, May 2021 [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[106] T. Shreedhar, R. Panda, S. Podanev, and V. Bajpai, "Evaluating quic performance over web, cloud storage, and video workloads," *IEEE Trans. on Netw. and Service Management*, vol. 19, no. 2, pp. 1366–1381, 2021.

[107] C. Boyd, A. Mathuria, and D. Stebila, *Protocols for authentication and Key Establishment*. Springer, 2020.

[108] K. Cohn-Gordon, C. Cremers, and L. Garratt, "On post-compromise security," in *2016 IEEE 29th Comput. Security Foundations Symp. (CSF)*, 2016, pp. 164–178 [Online]. Available: https://doi.org/10.1109/CSF.2016.19

[109] K. Yagna, B. Odayarkoil, and A. Ellis. "Pushy to the Limit: Evolving Netflix's WebSocket proxy for the future." Sep. 2024 [Online]. Available: https:

//netflixtechblog.com/pushy-to-the-limit-evolving-netflixs-websocket-proxy-for-the-future-b468bc0ff658

[110] M. Thomson and S. Turner, *Using TLS to Secure QUIC*, IETF Request for Comments 9001, May 2021 [Online]. Available: https://www.rfc-editor.org/info/rfc9001

[111] S. Yang, H. Li, and Q. Wu, "Performance analysis of quic protocol in integrated satellites and terrestrial networks," in *2018 14th Int. Wireless Commun. and Mobile Comput. Conf. (IWCMC)*, 2018, pp. 1425–1430 [Online]. Available: https://doi.org/10.1109/IWCMC.2018.8450388

[112] D. Werner, "How long should a satellite last: five years, ten years, 15, 30?" *SpaceNews*, May 2018. Available at https://spacenews.com/how-long-should-a-satellite-last/.

[113] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A Formal Security Analysis of the Signal Messaging Protocol," *J. of Cryptol.*, vol. 33, pp. 1914–1983, 2020.

[114] D. Balbás, D. Collins, and P. Gajland, "WhatsUpp with sender keys? Analysis, improvements and security proofs," in *ASIACRYPT 2023, Part V* (LNCS), J. Guo and R. Steinfeld, Eds. Springer, Singapore, Dec. 2023, vol. 14442, pp. 307–341 [Online]. Available: https://doi.org/10.1007/978-981-99-8733-7_10

[115] J. Alwen, D. Jost, and M. Mularczyk, "On the insider security of MLS," in *CRYPTO 2022, Part II* (LNCS), Y. Dodis and T. Shrimpton, Eds. Springer, Cham, Aug. 2022, vol. 13508, pp. 34–68 [Online]. Available: https://doi.org/10.1007/978-3-031-15979-4_2

[116] Cisco, "Zero-Trust Security for Webex," *Cisco Webex Public: Whitepaper*, 2021.

[117] A. Leon, C. Britt, and B. Hale, "Multi-device security application for unmanned surface and aerial systems," *Drones*, vol. 8, no. 5, 2024 [Online]. Available: https://doi.org/10.3390/drones8050200

[118] R. Mahy, "Messaging Layer Security Ciphersuite using XWing Key Exchange Mechanism," Internet Engineering Task Force, Internet-Draft draft-mahy-mls-xwing-00, Mar. 2024 [Online]. Available: https://datatracker.ietf.org/doc/draft-mahy-mls-xwing/00/

[119] J. Alwen, B. Hale, M. Mularczyk, and X. Tian, "Flexible Hybrid PQ MLS Combiner," Internet Engineering Task Force, Internet-Draft draft-hale-mls-combiner-01, Sep. 2024 [Online]. Available: https://datatracker.ietf.org/doc/draft-hale-mls-combiner/01/

[120] K. Cohn-Gordon, B. Beurdouche, and R. Robert. “MLS Implementations.” 2024. Available at: https://github.com/mlswg/mls-implementations.

[121] S. C. Burleigh, D. Horres, K. Viswanathan, M. Benson, and F. Templin, “Architecture for Delay-Tolerant Key Administration,” Internet Engineering Task Force, Internet-Draft draft-burleigh-dtnwg-dtka-02, Aug. 2018 [Online]. Available: https://datatracker.ietf.org/doc/draft-burleigh-dtnwg-dtka/02/

[122] J. Smailes, S. Köhler, S. Birnbach, M. Strohmeier, and I. Martinovic, “KeySpace: Public Key Infrastructure Considerations in Interplanetary Networks,” Nov. 2024. arXiv:2408.10963 [cs] [Online]. Available: https://doi.org/10.48550/arXiv.2408.10963

[123] S. C. Burleigh, S. Farrell, and M. Ramadas, *Licklider Transmission Protocol - Specification*, IETF Request for Comments 5326, Sep. 2008 [Online]. Available: https://www.rfc-editor.org/info/rfc5326

[124] H. Krawczyk, “A unilateral-to-mutual authentication compiler for key exchange (with applications to client authentication in tls 1.3),” in *Proc. of the 2016 ACM SIGSAC Conf. on Comput. and Commun. Security* (CCS ’16). New York, NY, USA: Assoc. for Comput. Machinery, 2016, pp. 1438–1450 [Online]. Available: https://doi.org/10.1145/2976749.2978325

[125] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis, “Security Analysis and Improvements for the IETF MLS Standard for Group Messaging,” in *Adv. in Cryptol. – CRYPTO 2020*. Springer-Verlag, 2020, pp. 248–277 [Online]. Available: https://doi.org/10.1007/978-3-030-56784-2_9

[126] R. Robert, “The Messaging Layer Security (MLS) Extensions,” Internet Engineering Task Force, Internet-Draft draft-ietf-mls-extensions-04, Apr. 2024 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-mls-extensions/04/

[127] J. Schier, “Interoperability in the Lunar Sphere through LunaNet and International Cooperation,” Interop Tokyo, Jun. 2024.

[128] Starlink, “Starlink 2024 progress report,” Starlink, Rep., 2024 [Online]. Available: https://www.starlink.com/public-files/starlinkProgressReport_2024.pdf?srsltid=AfmBOooiLQgj0mVyawOHU4bkhemznkk4FX0WKn8hR0YcgFw65TLZAc01

[129] E. S. Agency, “Report on the Space Economy 2024,” Rep., Dec. 2024 [Online]. Available: https://space-economy.esa.int/documents/b61btvmeaf6Tz2osXPu712bL0dwO3uqdOrFAwNTQ.pdf

[130] D. J. Israel *et al.*, "Lunanet: a flexible and extensible lunar exploration communications and navigation infrastructure," in *2020 IEEE Aerospace Conf.*, 2020, pp. 1–14 [Online]. Available: https://doi.org/10.1109/AERO47225.2020.9172509

[131] M. Anastos *et al.*, "The cost of maintaining keys in dynamic groups with applications to multicast encryption and group messaging," in *Theory of Cryptogr.: 22nd Int. Conf., TCC 2024, Milan, Italy, December 2–6, 2024, Proc., Part I*. Berlin, Heidelberg: Springer-Verlag, 2024, pp. 413–443 [Online]. Available: https://doi.org/10.1007/978-3-031-78011-0_14

[132] J. Alwen, D. Hartmann, E. Kiltz, and M. Mularczyk, "Server-aided continuous group key agreement," in *ACM CCS 2022*, H. Yin, A. Stavrou, C. Cremers, and E. Shi, Eds. ACM Press, Nov. 2022, pp. 69–82 [Online]. Available: https://doi.org/10.1145/3548606.3560632

[133] D. Soler, C. Dafonte, M. Fernández-Veiga, A. F. Vilas, and F. J. Nóvoa, "Experimental analysis of efficiency of the messaging layer security for multiple delivery services," *arXiv preprint arXiv:2502.18303*, 2025.

[134] B. Dowling, B. Hale, X. Tian, and B. Wimalasiri, "Key establishment in the space environment," 2025 [Online]. Available: https://arxiv.org/abs/2503.06785

[135] J. Alwen, S. Coretti, and Y. Dodis, "The double ratchet: Security notions, proofs, and modularization for the Signal protocol," in *EUROCRYPT 2019, Part I* (LNCS), Y. Ishai and V. Rijmen, Eds. Springer, Cham, May 2019, vol. 11476, pp. 129–158 [Online]. Available: https://doi.org/10.1007/978-3-030-17653-2_5

[136] T. Wallez, J. Protzenko, and K. Bhargavan, "TreeKEM: A Modular Machine-Checked Symbolic Security Analysis of Group Key Agreement in Messaging Layer Security," 2025. Publication info: Preprint. [Online]. Available: https://eprint.iacr.org/2025/410

[137] C. . Hale and B. Kohbrok, "The complexities of healing in secure group messaging: Why cross-group effects matter," pp. 1847–1864, 2021 [Online]. Available: https://www.usenix.org/conference/usenixsecurity21/presentation/cremershttps://www.usenix.org/conference/usenixsecurity21/presentation/cremers

[138] C. Brzuska, E. Cornelissen, and K. Kohbrok, "Security analysis of the mls key derivation," in *2022 IEEE Symp. on Security and Privacy (SP)*, 2022, pp. 2535–2553 [Online]. Available: https://doi.org/10.1109/SP46214.2022.9833678

[139] N. Sullivan and S. Turner. "Messaging layer security: Secure and usable end-to-end encryption." IETF Blog. 2024 [Online]. Available: https://www.ietf.org/blog/mls-secure-and-usable-end-to-end-encryption/

[140] B. Beurdouche, E. Rescorla, E. Omara, S. Inguva, and A. Duric, "The Messaging Layer Security (MLS) Architecture," Internet Engineering Task Force, Internet-Draft draft-ietf-mls-architecture-15, Aug. 2024 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-mls-architecture/15/

[141] A. Bienstock, Y. Dodis, S. Garg, G. Grogan, M. Hajiabadi, and P. Rösler, "On the worst-case inefficiency of cgka," in *Theory of Cryptogr.: 20th Int. Conf., TCC 2022, Chicago, IL, USA, November 7–10, 2022, Proc., Part II*. Berlin, Heidelberg: Springer-Verlag, 2022, pp. 213–243 [Online]. Available: https://doi.org/10.1007/978-3-031-22365-5_8

[142] M. Anastos *et al.*, "The Cost of Maintaining Keys in Dynamic Groups with Applications to Multicast Encryption and Group Messaging," in *Theory of Cryptogr.: 22nd International Conference, TCC 2024, Milan, Italy, December 2–6, 2024, Proceedings, Part I*. Berlin, Heidelberg: Springer-Verlag, Dec. 2024, pp. 413–443 [Online]. Available: https://doi.org/10.1007/978-3-031-78011-0_14

[143] C. Chevalier, G. Lebrun, A. Martinelli, and J. Plût, "The art of bonsai: How well-shaped trees improve the communication cost of MLS," Cryptology ePrint Archive, Paper 2024/746, 2024 [Online]. Available: https://eprint.iacr.org/2024/746

[144] B. Dowling, B. Hale, X. Tian, and B. Wimalasiri, "Cryptography is rocket science," *cic*, vol. 1, Issue 4, 2025 [Online]. Available: https://doi.org/10.62056/a39qudhdj

[145] M. Sommer, A. Sterz, M. Vogelbacher, H. Bellafkir, and B. Freisleben, "Quicl: A quic convergence layer for disruption-tolerant networks," in *Proc. of the Int'l ACM Conf. on Modeling Analysis and Simulation of Wireless and Mobile Syst.* (MSWiM '23). New York, NY, USA: Assoc. for Comput. Machinery, 2023, pp. 37–46 [Online]. Available: https://doi.org/10.1145/3616388.3617525

[146] B. Dowling, B. Hale, K. Kohbrok, R. Robert, X. Tian, and B. Wimalasiri, "Securing QUIC with MLS," Internet Engineering Task Force, Internet-Draft draft-tian-quic-quicmls-00, Jul. 2025 [Online]. Available: https://datatracker.ietf.org/doc/draft-tian-quic-quicmls/00/

[147] A. Hülsing, T. Lange, and F. Weber, "Key-Update Mechanism for SDLSP," European Space Research and Technology Centre, May 2024.

[148] S. A. Menesidou and V. Katos, "Authenticated Key Exchange (AKE) in Delay Tolerant Networks," in *Inf. Security and Privacy Research*, vol. 376, D. Gritzalis, S. Furnell, and M. Theoharidou, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 49–60. Series Title: IFIP Advances in Information and Communication Technology [Online]. Available: https://doi.org/10.1007/978-3-642-30436-1_5

[149] F. Fuchs, F. Walter, and F. Tschorsch, "BERMUDA: A BPSec-compatible key management scheme for DTNs," Cryptology ePrint Archive, Paper 2025/806, 2025 [Online]. Available: https://doi.org/tbd

[150] W. L. Van Besien, "Dynamic, non-interactive key management for the bundle protocol," in *Proc. of the 5th ACM Workshop on Challenged Networks* (CHANTS '10). New York, NY, USA: Assoc. for Comput. Machinery, 2010, pp. 75–78 [Online]. Available: https://doi.org/10.1145/1859934.1859951

[151] K. Cohn-Gordon, C. Cremers, and L. Garratt, "On Post-compromise Security," in *2016 IEEE 29th Comput. Security Foundations Symp. (CSF)*. IEEE, Jun. 2016, pp. 164–178 [Online]. Available: https://doi.org/10.1109/CSF.2016.19

[152] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, Nov. 1984 [Online]. Available: https://doi.org/10.1145/357401.357402

[153] B. Sipos, *Bundle Protocol Version 7 Administrative Record Types Registry*, IETF Request for Comments 9713, Jan. 2025 [Online]. Available: https://www.rfc-editor.org/info/rfc9713

[154] NASA Jet Propulsion Laboratory, "Interplanetary Overlay Network (ION-DTN)," GitHub, 2025. [Source code; Accessed: 12-Sept-2025] [Online]. Available: https://github.com/nasa-jpl/ION-DTN/tree/current

[155] K. Mckeever, *Bundle Protocol Security (BPSec)*, IETF Request for Comments 9172, 2022 [Online]. Available: https://www.rfc-editor.org/info/rfc9172

[156] Cisco Systems, "MLSpp: Implementation of Messaging Layer Security," https://github.com/cisco/mlspp, 2025. C++ implementation of the Messaging Layer Security protocol [Online]. Available: https://github.com/cisco/mlspp

[157] M. Blanchet, W. Eddy, and T. Li, "An Architecture for IP in Deep Space," Internet Engineering Task Force, Internet-Draft draft-many-deepspace-ip-architecture-01, Nov. 2024 [Online]. Available: https://datatracker.ietf.org/doc/draft-many-deepspace-ip-architecture/01/

[158] U. N. O. for Outer Space Affairs (UNOOSA), "Online Index of Objects Launched into Outer Space," 2025. Retrieved from https://www.unoosa.org/oosa/osoindex/search-ng.jspx.

[159] E. Vyncke, "Taking IP To Other Planets Charter," 2025. Retrieved from https://datatracker.ietf.org/doc/charter-ietf-tiptop/.

[160] J. Smailes, R. David, S. Kohler, S. Birnbach, and I. Martinovic, "Poster: spacequic: Securing communication in computationally constrained spacecraft," 2023 [Online]. Available: https://arxiv.org/abs/2305.12948

[161] M. Sommer, A. Sterz, M. Vogelbacher, H. Bellafkir, and B. Freisleben, "QUICL: A QUIC Convergence Layer for Disruption-tolerant Networks," in *Proceedings of the Int'l ACM Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems* (MSWiM '23). Association for Computing Machinery, 2023, p. 37–46 [Online]. Available: https://doi.org/10.1145/3616388.3617525

[162] G. Amponis *et al.*, "Channel-Aware QUIC Control for Enhanced CAM Communications in C-V2X Deployments Over Aerial Base Stations," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 7, pp. 9320–9333, 2024 [Online]. Available: https://doi.org/10.1109/TVT.2024.3393614

[163] Sandvine, "The Global Internet Phenomena Report," Jan. 2023.

[164] N. Aviram, K. Gellert, and T. Jager, "Session Resumption Protocols and Efficient Forward Security for TLS 1.3 0-RTT," *J. Cryptol.*, vol. 34, no. 3, July 2021 [Online]. Available: https://doi.org/10.1007/s00145-021-09385-0

[165] F. Günther, B. Hale, T. Jager, and S. Lauer, "0-rtt key exchange with full forward secrecy," in *Advances in Cryptology – EUROCRYPT 2017*. Springer International Publishing, 2017, pp. 519–548.

[166] K. Cohn-Gordon, C. Cremers, and L. Garratt, "On Post-Compromise Security," in *2016 IEEE 29th Computer Security Foundations Symposium (CSF)*. IEEE, 2016, pp. 164–178.

[167] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis, "Modular design of secure group messaging protocols and the security of mls," in *Proc. of the 2021 ACM SIGSAC Conf. on Comput. and Commun. Security* (CCS '21). Assoc. for Comput. Machinery, 2021, pp. 1463–1483 [Online]. Available: https://doi.org/10.1145/3460120.3484820

[168] C. Brzuska, E. Cornelissen, and K. Kohbrok, "Security analysis of the MLS key derivation," in *2022 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2022, pp. 2535–2553 [Online]. Available: https://doi.org/10.1109/SP46214.2022.9833678

[169] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A Formal Security Analysis of the Signal Messaging Protocol," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2017, pp. 451–466 [Online]. Available: https://doi.org/10.1109/EuroSP.2017.27

[170] B. Dowling and B. Hale, "Secure Messaging Authentication against Active Man-in-the-Middle Attacks," in *2021 IEEE European Symposium on Security and Privacy (EuroS&P)*, 2021, pp. 54–70 [Online]. Available: https://doi.org/10.1109/EuroSP51992.2021.00015

[171] R. Fiedler and F. Günther, "Security Analysis of Signal's PQXDH Handshake," in *Public-Key Cryptography – PKC 2025*. Springer-Verlag, 2025, p. 137–169 [Online]. Available: https://doi.org/10.1007/978-3-031-91823-0_5

[172] A. Bienstock, J. Fairoze, S. Garg, P. Mukherjee, and S. Raghuraman, "A More Complete Analysis of the Signal Double Ratchet Algorithm," in *Advances in Cryptology – CRYPTO 2022*. Springer-Verlag, 2022, p. 784–813 [Online]. Available: https://doi.org/10.1007/978-3-031-15802-5_27

[173] M. Fischlin, F. Günther, and C. Janson, "Robust Channels: Handling Unreliable Networks in the Record Layers of QUIC and DTLS 1.3," *J. Cryptol.*, vol. 37, no. 2, Jan. 2024 [Online]. Available: https://doi.org/10.1007/s00145-023-09489-9

[174] A. Delignat-Lavaud *et al.*, "A Security Model and Fully Verified Implementation for the IETF QUIC Record Layer," in *2021 IEEE Symposium on Security and Privacy (SP)*, 2021, pp. 1162–1178 [Online]. Available: https://doi.org/10.1109/SP40001.2021.00039

[175] S. Chen, S. Jero, M. Jagielski, A. Boldyreva, and C. Nita-Rotaru, "Secure Communication Channel Establishment: TLS 1.3 (over TCP Fast Open) versus QUIC," *J. Cryptol.*, vol. 34, no. 3, July 2021 [Online]. Available: https://doi.org/10.1007/s00145-021-09389-w

[176] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru, "How secure and quick is quic? provable security and performance analyses," in *2015 IEEE Symp. on Security and Privacy*, 2015, pp. 214–231 [Online]. Available: https://doi.org/10.1109/SP.2015.21

[177] B. Beurdouche, E. Rescorla, E. Omara, S. Inguva, and A. Duric, "The Messaging Layer Security (MLS) Architecture," RFC 9750, Apr. 2025 [Online]. Available: https://doi.org/10.17487/RFC9750

[178] R. Robert, "The Messaging Layer Security (MLS) Extensions," Internet Engineering Task Force, Internet-Draft draft-ietf-mls-extensions-07, July 2025. Work in Progress [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-mls-extensions/07/

[179] Marten Seemann, "quic-go," 2025. https://pkg.go.dev/github.com/quic-go/quic-go.

[180] Cisco Systems, “go-mls,” 2021. https://pkg.go.dev/github.com/cisco/go-mls.

[181] Google, “crypto/tls,” 2025. https://pkg.go.dev/crypto/tls.

[182] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, “On the Security of TLS-DHE in the Standard Model,” in *Annual Cryptology Conference*. Springer, 2012, pp. 273–293.

[183] T. Jager, F. Kohlar, S. Schäge, and J. Schwenk, “On the security of TLS-DHE in the standard model,” in *CRYPTO 2012* (LNCS), R. Safavi-Naini and R. Canetti, Eds. Springer, Berlin, Heidelberg, Aug. 2012, vol. 7417, pp. 273–293 [Online]. Available: https://doi.org/10.1007/978-3-642-32009-5_17

[184] R. Lychev, S. Jero, A. Boldyreva, and C. Nita-Rotaru, “How secure and quick is quic? provable security and performance analyses,” in *Proc. of the 16th Annu. Inf. Security Symp.* (CERIAS ’15). West Lafayette, IN: CERIAS - Purdue Univ., 2015.

[185] M. Fischlin and F. Günther, “Multi-stage key exchange and the case of Google’s QUIC protocol,” in *ACM CCS 2014*, G.-J. Ahn, M. Yung, and N. Li, Eds. ACM Press, Nov. 2014, pp. 1193–1204 [Online]. Available: https://doi.org/10.1145/2660267.2660308

[186] M. Bellare and P. Rogaway, “Entity authentication and key distribution,” in *CRYPTO’93* (LNCS), D. R. Stinson, Ed. Springer, Berlin, Heidelberg, Aug. 1994, vol. 773, pp. 232–249 [Online]. Available: https://doi.org/10.1007/3-540-48329-2_21

[187] B. Dowling, P. Rösler, and J. Schwenk, “Flexible authenticated and confidential channel establishment (facce): Analyzing the noise protocol framework,” in *Public-Key Cryptogr. – PKC 2020: 23rd IACR Int. Conf. on Practice and Theory of Public-Key Cryptogr., Edinburgh, UK, May 4–7, 2020, Proc., Part I*. Berlin, Heidelberg: Springer-Verlag, 2020, pp. 341–373 [Online]. Available: https://doi.org/10.1007/978-3-030-45374-9_12

[188] K. Kohbrok, “Phnx-im/hpqmls,” https://github.com/phnx-im/hpqmls.

[189] D. Stebila, S. Fluhrer, and S. Gueron, “Hybrid Key Exchange in TLS 1.3,” Internet Engineering Task Force, Internet-Draft draft-ietf-tls-hybrid-design-16, Sep. 2025 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/16/

[190] A. Banerjee, T. Reddy.K, D. Schoinianakis, T. Hollebeek, and M. Ounsworth, “Post-Quantum Cryptography for Engineers,” Internet Engineering Task Force, Internet-Draft draft-ietf-pquip-pqc-engineers-14, Aug. 2025 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-pquip-pqc-engineers/14/

[191] T. Reddy.K, D. Wing, B. S, and K. Kwiatkowski, "Adapting Constrained Devices for Post-Quantum Cryptography," Internet Engineering Task Force, Internet-Draft draft-ietf-pquip-pqc-hsm-constrained-01, Jul. 2025 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-pquip-pqc-hsm-constrained/01/

[192] W. Castryck and T. Decru, "An Efficient Key Recovery Attack on SIDH," in *Adv. in Cryptol. – EUROCRYPT 2023*. Springer-Verlag, 2023, pp. 423–447 [Online]. Available: https://doi.org/10.1007/978-3-031-30589-4_15

[193] W. Beullens, "Breaking Rainbow Takes a Weekend on a Laptop," in *Adv. in Cryptol. – CRYPTO 2022*. Springer-Verlag, 2022, pp. 464–479 [Online]. Available: https://doi.org/10.1007/978-3-031-15979-4_16

[194] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-Quantum Authentication in TLS 1.3: A Performance Study," Network and Distributed Systems Security (NDSS) Symposium 2020, 2020 [Online]. Available: https://doi.org/10.14722/ndss.2020.24203

[195] N. Alnahawi, J. Müller, J. Oupický, and A. Wiesmaier, "A Comprehensive Survey on Post-Quantum TLS," *IACR Communications in Cryptology*, vol. 1, no. 2, 2024 [Online]. Available: https://doi.org/10.62056/ahee0iuc

[196] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking post-quantum cryptography in TLS," in *Int. Conf. on Post-Quantum Cryptogr.* Springer, 2020, pp. 72–91.

[197] D. Connolly, P. Schwabe, and B. Westerbaan, "X-Wing: General-Purpose Hybrid Post-Quantum KEM," Internet Engineering Task Force, Internet-Draft draft-connolly-cfrg-xwing-kem-09, Sep. 2025 [Online]. Available: https://datatracker.ietf.org/doc/draft-connolly-cfrg-xwing-kem/09/

[198] P. R&D and Cryspen, "OpenMLS Book," https://book.openmls.tech/.

[199] M. Barbosa *et al.*, "X-Wing: The Hybrid KEM You've Been Looking For," Cryptology ePrint Archive, Paper 2024/039, 2024 [Online]. Available: https://doi.org/10.62056/a3qj89n4e

[200] R. Mahy and R. Barnes, "ML-KEM and Hybrid Cipher Suites for Messaging Layer Security," Internet Engineering Task Force, Internet-Draft draft-ietf-mls-pq-ciphersuites-00, Jul. 2025 [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-mls-pq-ciphersuites/00/

[201] N. Greene and B. Hale, "Making post quantum key exchange efficient: An implementation with the MLS protocol," Cryptology ePrint Archive, Paper 2025/1881, 2025 [Online]. Available: https://eprint.iacr.org/2025/1881

[202] J. Brendel, C. Cremers, D. Jackson, and M. Zhao, "The provable security of ed25519: Theory and practice," Cryptology ePrint Archive, Paper 2020/823, 2020 [Online]. Available: https://eprint.iacr.org/2020/823

[203] B. Heisler, "bheisler/criteron," https://github.com/bheisler/criterion.rs.

[204] P. Urbanek, "pawurb/hotpath," https://github.com/pawurb/hotpath.

[205] B. Westerbaan and L. Valenta. "A Look at the Latest Post-Quantum Signature Standardization Candidates." https://blog.cloudflare.com/another-look-at-pq-signatures/. Nov. 2024.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

DUDLEY KNOX LIBRARY

NAVAL POSTGRADUATE SCHOOL

WWW.NPS.EDU

WHERE SCIENCE MEETS THE ART OF WARFARE