



Master's thesis

Master's Programme in Computer Science

Post-Quantum Multi-Party Key Exchange: Design, Implementation and Evaluation

Sanish Gurung

July 31, 2026

FACULTY OF SCIENCE
UNIVERSITY OF HELSINKI

Contact information

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki, Finland

Email address: info@cs.helsinki.fi

URL: <http://www.cs.helsinki.fi/>

Acknowledgements

I would like to thank Prof. Valtteri Niemi and the other members of the “Secure Systems” and “Wireless Internet” research groups for their invaluable help and continuous feedback on my work. This thesis is based on the research work I did for the Business Finland project “Beyond the Limits of Post-Quantum Cryptography (BLimPQC)” (Project No. 100/31/2024). This thesis was also partially funded by the Jouni Lestelin Fund scholarship 2025-2026.

Tiedekunta — Fakultet — Faculty		Koulutusohjelma — Utbildningsprogram — Study programme	
Faculty of Science		Master's Programme in Computer Science	
Tekijä — Författare — Author			
Sanish Gurung			
Työn nimi — Arbetets titel — Title			
Post-Quantum Multi-Party Key Exchange: Design, Implementation and Evaluation			
Ohjaajat — Handledare — Supervisors			
Prof. Valtteri Niemi, Markku Kojo			
Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages	
Master's thesis	July 31, 2026	56 pages	
Tiivistelmä — Referat — Abstract			
Key establishment among multiple participants over an insecure channel is a widely adopted cryptographic operation. However, a quantum adversary can employ Shor's algorithm, compromising most key exchange protocols based on the discrete logarithm problem. Safeguarding against such quantum attacks requires transitioning to post-quantum key establishment protocols that incorporate quantum-resistant mathematical hardness assumptions. This thesis addresses the research gap on post-quantum key exchange for multiple participants and decentralized settings. It proposes the feasible solution, Multi-Party Key Exchange (MP-KEX) protocol, to derive a conference key, implemented and evaluated using OpenSSL and liboqs cryptographic libraries. The research evaluates four cryptographic algorithm combinations of key exchange and authentication, comparing their performance, identifying the protocol's bottleneck, and addressing the formulated research questions. This thesis demonstrates that the computational cost of MP-KEX protocol is largely dictated by digital signature algorithm used, with ML-DSA being inherently slower than ECDSA-P256. On the other hand, ML-KEM is computationally fast compared to X25519 for key exchange purpose. The main bottleneck of the protocol arises from the verification phase in the naive model, whose computation time is reduced by nearly 50 % with a quorum-based verification system when the group size is large. The findings demonstrate the feasibility of MP-KEX protocol, proposing novel post-quantum key exchange protocol that provides scalable performance and empirical comparison between classical and post-quantum approaches to contributory group key establishment. ACM Computing Classification System (CCS) Theory of Computation → Computational complexity and cryptography → Cryptographic protocols Security and privacy → Cryptography → Key management			
Avainsanat — Nyckelord — Keywords			
Post-Quantum Cryptography, Multi-Party Key Exchange, Key Encapsulation Mechanism			
Säilytyspaikka — Förvaringsställe — Where deposited			
Helsinki University Library			
Muita tietoja — Övriga uppgifter — Additional information			

Table 1: Abbreviations and Definitions

Abbreviation	Full Form
3GPP	3rd Generation Partnership Project
AES	Advanced Encryption Standard
BLAKE	BLAKE Cryptographic Hash Function
CCA	Chosen Ciphertext Attack
CMA	Chosen Message Attack
CNSA	Commercial National Security Algorithm Suite
CRQC	Cryptographically Relevant Quantum Computer
DH	Diffie-Hellman
DLP	Discrete Logarithm Problem
DSA	Digital Signature Algorithm
DDH	Decisional Diffie-Hellman
EC	Elliptic Curve
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem
ECDSA	Elliptic Curve Digital Signature Algorithm
ECIES	Elliptic Curve Integrated Encryption Scheme
EdDSA	Edwards-curve Digital Signature Algorithm
ETSI	European Telecommunications Standards Institute
EU	European Union
FIPS	Federal Information Processing Standard
FN-DSA	Falcon-based Digital Signature Algorithm
GSMA	Global System for Mobile Communications Association
HNDL	Harvest Now Decrypt Later
HQC	Hamming Quasi-Cyclic
HTTPS	Hypertext Transfer Protocol Secure
IEC	International Electrotechnical Commission
IETF	Internet Engineering Task Force
IFP	Integer Factorization Problem
IKEv2	Internet Key Exchange Version 2

Continued on next page

Table 1 – continued from previous page

Abbreviation	Full Form
IND-CCA2	Indistinguishability under Adaptive Chosen Ciphertext Attack
IoT	Internet of Things
IPsec	Internet Protocol Security
ISO	International Organization for Standardization
KDF	Key Derivation Function
KEM	Key Encapsulation Mechanism
KEX	Key Exchange
LWE	Learning with Errors
MAC	Message Authentication Code
ML-DSA	Module-Lattice-Based Digital Signature Algorithm
ML-KEM	Module-Lattice-Based Key Encapsulation Mechanism
MLWE	Module Learning with Errors
MP-KEX	Multi-Party Key Exchange
MPC	Multi-Party Computation
MQOM	MQ on my Mind
MSIS	Module Short Integer Solution
MTI	Mandatory-To-Implement
MTU	Maximum Transmission Unit
NIS CG	Network and Information Systems Cooperation Group
NIST	National Institute of Standards and Technology
NSA	National Security Agency
NTT	Number Theoretic Transform
PKI	Public Key Infrastructure
PQ	Post-Quantum
PQC	Post-Quantum Cryptography
PQDSA	Post-Quantum Digital Signature Algorithm
QR-UOV	Quotient Ring Unbalanced Oil and Vinegar
RSA	Rivest-Shamir-Adleman
SDitH	Syndrome Decoding in the Head
SDO	Standards Development Organization
SHA	Secure Hash Algorithm

Continued on next page

Table 1 – continued from previous page

Abbreviation	Full Form
SK	Session Key
SLH-DSA	Stateless Hash-Based Digital Signature Algorithm
SNOVA	Structured Non-linear Oil and Vinegar Algorithm
SSH	Secure Shell
SUF-CMA	Strong Existential Unforgeability under Chosen Message Attack
TLS	Transport Layer Security
TNFL	Trust Now Forge Later
UOV	Unbalanced Oil and Vinegar

Contents

1	Introduction	1
2	Quantum Threat to Cryptography	4
3	Problem Statement and Objectives	8
3.1	Related work	8
3.2	Objectives and Research Questions	9
4	Preliminaries	10
4.1	Cryptography	10
4.1.1	Asymmetric Cryptography	12
4.1.2	Symmetric-Key Cryptography	13
4.1.3	Hash Functions	14
4.2	Post-Quantum Cryptography	15
4.2.1	Key Encapsulation Mechanisms	18
4.2.2	Digital Signature Algorithms	19
4.3	Hybrid Approach	21
4.4	Cryptographic Agility	22
5	Protocol Design of MP-KEX	23
5.1	Assumptions of the MP-KEX	23
5.2	Setup and Topology Alignment (Round 0)	25
5.3	Key Establishment With Neighbors (Round 1)	27
5.4	Bridges (Round 2)	30
5.5	Deriving Conference Key (Round 3)	31
6	Performance Analysis	36
6.1	Experimental Setup	36
6.2	MP-KEX Evaluation	37

6.2.1	Experiments	37
6.2.2	Results	38
6.2.3	Addressing Research Questions	43
7	Future Directions	45
8	Conclusions	47
	References	49

1 Introduction

Modern digital systems, including mobile devices and the Internet of Things (IoT), depend on secure communications to enable applications such as online banking, cross-geographical data exchange, and the protection of sensitive information assets. Such secure communications fall under the broader domain of cybersecurity, which encompasses the holistic protection of digital assets, secure key distribution over an untrusted channel, and addressing security flaws. Cybersecurity is often achieved through defensive practices that incorporate cryptography, which is one of its foundational pillars [41].

Modern communications rely on three cryptographic functionalities: public-key encryption, digital signatures, and key exchange [15]. These functionalities are based on asymmetric cryptographic systems, such as Diffie-Hellman (DH) [19], Rivest-Shamir-Adleman (RSA) [68] and Elliptic-Curve Cryptography (ECC) [15, 31]. However, as quantum computing developments accelerate, these conventional cryptographic systems face obsolescence due to the impending emergence of large-scale quantum computers, known as Cryptographically Relevant Quantum Computers (CRQCs) [15].

These computers harness the principles of quantum mechanics and transcend the computational boundaries of classical computers [42]. CRQCs are able to solve the Integer Factorization Problem (IFP) and the Discrete Logarithm Problem (DLP) by using Shor’s algorithm [15, 69]. Concurrently, Grover’s algorithm [25] enables a quantum adversary[†] to search an n -bit key space in $\mathcal{O}(2^{n/2})$ operations rather than the classical brute-force complexity of $\mathcal{O}(2^n)$, effectively halving the security level in bits. For instance, AES-256 is as secure against a quantum adversary as AES-128 is against a classical adversary.

Between the two quantum-enabled algorithms, Shor’s algorithm stands out as the more existential threat to Public Key Infrastructures (PKIs). In anticipation of sufficiently mature quantum computers, a quantum adversary may already intercept and store ciphertexts[‡] today, intending to decrypt them once Shor’s algorithm can be feasibly applied, a strategy known as “Harvest Now Decrypt Later (HNDL)”. Thus, HNDL transforms ciphertexts into a deferred liability, undermining cryptographic primitives, and cascading this vulnerability

[†]An adversary capable of breaking quantum-vulnerable cryptography with the help of quantum computers

[‡]Encrypted data (in transit or at rest)

to the infrastructures and archives that utilize such primitives [28].

To counteract the threat of HNDL, Standards Development Organizations (SDOs) have taken initiatives to fortify current public-key infrastructures by standardizing robust Post-Quantum Cryptography (PQC) primitives [20]. A prominent example of such an SDO is the National Institute of Standards and Technology (NIST) [50]. NIST aims to continuously support novel quantum-resistant algorithms and hosts workshops to foster collaboration among academic researchers, industry partners, and government stakeholders [57]. A cornerstone of this effort is its PQC standardization project, initiated in 2016, wherein researchers propose their PQC algorithms and such algorithms are subjected to exhaustive peer-review, focusing on new public-key standards in two categories: Key Encapsulation Mechanisms (KEMs) and Digital Signature Algorithms (DSAs) [53].

Given that the exact date of Q-Day* remains indeterminate and standardization takes considerable time, practitioners must proactively adopt hybrid cryptographic approaches[†], cryptographic agility[‡] or a combination of both as interim measures [20, 39, 42, 67]. Although these strategies significantly improve overall security, they introduce practical challenges such as increased computational overhead, larger key sizes, and interoperability issues. Addressing these challenges requires coordinated international effort to achieve global adoption [39].

A critical area where these interim measures must be applied is Key Exchange (KEX) mechanisms predicated on discrete logarithmic and integer factorization problems. Since CRQCs render these mechanisms insecure, the cryptographic paradigm must shift toward Key Encapsulation Mechanisms (KEMs) that leverage post-quantum mathematical hardness assumptions. This transition is vital because the keys generated by these mechanisms are subsequently used for session key derivation, symmetric encryption, and decryption. Consequently, compromising these keys implicitly breaches any subsequent process or archive relying upon them. This challenge is particularly pronounced in multi-party settings, where practical post-quantum group key exchange solutions remain limited.

To address this vulnerability, this thesis proposes MP-KEX, a post-quantum protocol for establishing a conference key among an arbitrary number of participants. The MP-

*A hypothetical future date on which CRQC becomes capable of breaking current public-key standards, such as RSA, DH and ECC

[†]A concept of combining a PQ algorithm with a classical cryptographic primitive for key exchange or digital signature

[‡]The capacity of a system to switch cryptographic algorithms swiftly when existing ones are broken or compromised, without fundamental redesign of the system

KEX protocol is inspired by the Burmester-Desmedt broadcast variant protocol [14] and distributes the computational workload uniformly among all participants. The protocol is designed, implemented using OpenSSL [61] and liboqs [60], and evaluated across four cryptographic algorithm combinations, providing a direct empirical comparison between classical and post-quantum approaches to group key establishment.

This thesis is organized as follows. Chapter 2 examines the quantum threat landscape, encompassing the implications of Shor’s and Grover’s algorithms on current cryptographic systems, the urgency quantified by Mosca’s Inequality, and the regulatory responses undertaken by the governmental and supranational bodies. Chapter 3 surveys related work on conference key agreement protocols, identifies the research gap in post-quantum multi-party key exchange, and formulates research objectives and contributions of this thesis. Chapter 4 introduces the foundational concepts of cryptography, providing concise descriptions of relevant cryptographic primitives and their mathematical background, alongside newly standardized post-quantum algorithms, hybrid approaches, and cryptographic agility. Chapter 5 details the design and execution rounds of the Multi-Party Key Exchange (MP-KEX) protocol to establish a conference key. Chapter 6 describes the experimental setup, presents the empirical results across four cryptographic algorithm combinations, and answers the research questions. Chapter 7 outlines the potential future research directions derived from the implementation results. Finally, Chapter 8 concludes this thesis, highlighting its core contributions and evaluating the significance of the work in the context of post-quantum preparedness.

2 Quantum Threat to Cryptography

As already mentioned, quantum computers harness the principles of quantum mechanics, enabling a quantum adversary to utilize two algorithms: Shor's and Grover's. Shor's algorithm poses a catastrophic threat by breaking cryptographic algorithms built upon integer factorization, such as RSA, and discrete logarithm problems, such as DH and Elliptic Curve (EC) cryptography, consequently undermining PKIs. Recent studies by Google Quantum AI have found that breaking RSA-2048 would require running one million noisy qubits for a week [24]. Breaking the 256-bit ECDLP over secp256k1 [70] would require either less than 1200 logical qubits and less than 90 million Toffoli gates or less than 1450 logical qubits and less than 70 million Toffoli gates, executable in minutes on a superconducting architecture [6].

Cryptographically Relevant Quantum Computers (CRQCs) pose a systemic threat to classical cryptography and its cascading effects across the telecommunications, government, defense, healthcare, and financial sectors. As 20 billion IoT devices are estimated to be active worldwide, the IoT ecosystem is at significant risk [7]. Since IoT devices utilize lightweight encryption for operational efficiency, migration to PQC primitives is challenging [7]. In telecommunications, a quantum breach could disrupt channels within communication infrastructures. Network technologies such as Transport Layer Security (TLS), Hypertext Transfer Protocol Secure (HTTPS), and email encryption are vulnerable because they are based on asymmetric cryptography [7]. In the healthcare sector, sensitive patient health records can break privacy and expose chronic medical histories due to CRQCs [7]. Quantum attacks can compromise diplomatic communications and intercept military communications, weakening geopolitical stability and security [7].

Figure 2.1 depicts how quantum threats are broadly categorized into public-key and symmetric-key cryptography, and which quantum algorithms threaten each category. The red and blue zones indicate vulnerabilities to Shor's and Grover's algorithms, respectively. Under public-key cryptography, there are two major threats: Harvest Now Decrypt Later (HNDL) and Trust Now Forge Later (TNFL).

The HNDL paradigm threatens the confidentiality of data, introducing severe disruptions across various sectors, ranging from financial transactions to satellite communications. Although the impact of HNDL spans over all these domains, the exposure window and

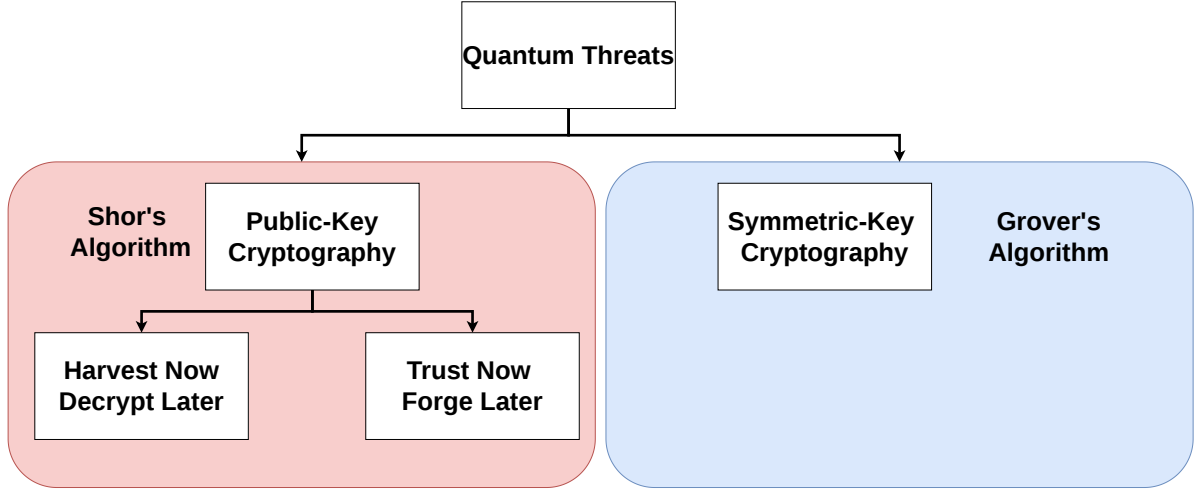


Figure 2.1: Classification of Quantum Threat

data lifetime vary significantly depending on the sector [28]. For instance, in financial transactions, data is exposed for a short period and its lifetime may be as short as a single month. In contrast, for confidential state intelligence, the exposure time of data is critical, and its lifetime can extend over more than thirty years [28]. This suggests that although HNDL impacts several sectors, quantifying the value of such encrypted data depends entirely on the context of the domain and the specific information it carries.

In contrast to confidentiality risks, the TNFL paradigm targets system integrity and authenticity [27]. Under this threat model, a quantum adversary uses Shor’s algorithm to derive the signature signing key [7]. This allows the adversary to forge digital signatures on malicious software or documents [7]. This directly breaks non-repudiation, a property that prevents a signer from denying that they signed a message. Consequently, it destroys the underlying trust of all authenticated messages. If an adversary can derive these keys, any signed message could be a forgery. Ultimately, it creates a systemic trust crisis, making it nearly impossible to distinguish genuine software, firmware, or documents from malicious copies [27, 7].

As the post-quantum threats are imminent, the urgency of migrating to quantum-safe defense mechanisms is gauged by the gold-standard formula, called Mosca Inequality, proposed by Dr. Michele Mosca [41]. The Mosca Inequality is defined with three timeframes: shelf-life time^{*}, migration time[†], and the threat time[‡] [41, 42]. The Mosca Inequality is

^{*}Remaining time that the data must still stay confidential

[†]Anticipated time it takes practitioners to deploy quantum-resistant alternatives to protect information

[‡]Remaining time until the adversary is able to utilize quantum computers to break into the system for malicious purposes

given by:

$$T_{\text{shelf-life}} + T_{\text{migration}} < T_{\text{threat}}, \quad (2.1)$$

where all variables are expressed in years [40]. This inequality states that the total duration required to protect information assets, $T_{\text{shelf-life}}$ and to transition the systems to quantum-resistant solutions, $T_{\text{migration}}$ must precede “Q-Day”, represented by T_{threat} [41, 42].

Figure 2.2 visualizes the timeline of Mosca’s Inequality and the potential exposure window. It illustrates that when $T_{\text{shelf-life}}$ is projected backward from Q-Day, any overlap with $T_{\text{migration}}$ initiated today defines the exposure threat window, indicated by red slashes. This overlap leaves ample time for an adversary to initiate quantum attacks.

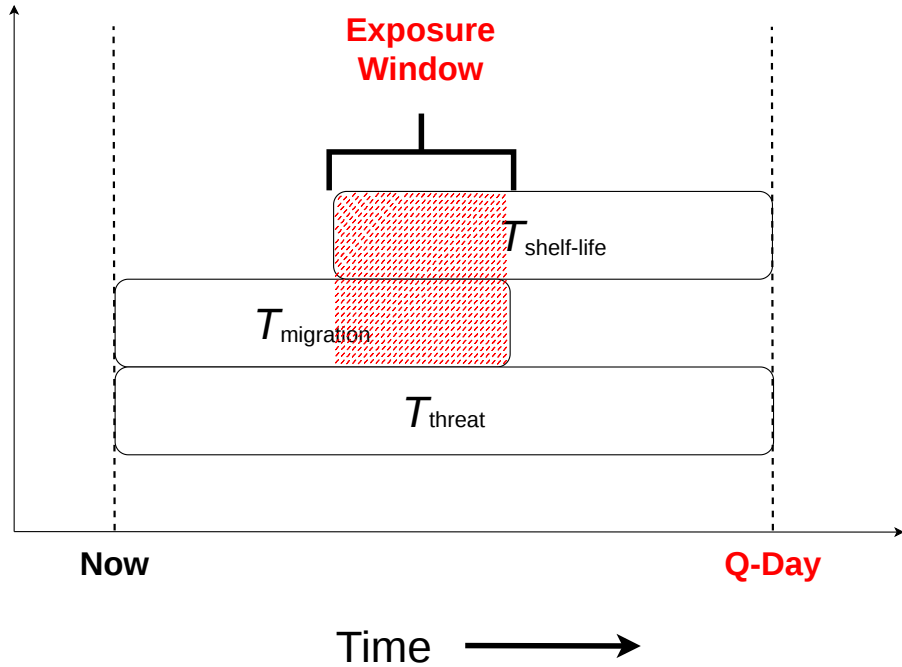


Figure 2.2: Timeline Visualization of Mosca’s Inequality and the Potential Exposure Window

The Global Risk Institute report [42], a comprehensive survey report, emphasizes a steady transition, as precipitous migration can inadvertently introduce security flaws, consequently expanding the system’s attack surface. In addition, potential challenges such as the interoperability of components and backward compatibility issues may arise in implementations, ultimately delaying migration procedures [42]. Therefore, structured migration should be adopted in which the entity, such as the system or protocol, is stress-tested prior to real-world deployment, to outpace the narrowing threat window.

The report analyzes surveys from 26 experts worldwide who have worked on the develop-

ment, commercialization, and research of quantum computing [42]. The report indicates that in the first five-year horizon (from 2025 to 2030), half of the polled experts estimated that there is less than 1% probability that a CRQC could break RSA-2048 within a 24-hour window [42]. In stark contrast, when looking at the ten-year horizon (by 2035), half of the surveyed experts estimated that the likelihood is at least 50%. This upward shift in probability suggests that T_{threat} is shrinking aggressively within the first ten-year horizon, underscoring the urgency of transition to quantum-resistant alternative solutions [42].

The U.S. National Security Agency (NSA) issued a cybersecurity advisory to mandate a transition to Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) [59]. This advisory introduces a suite of PQC algorithms designed to protect U.S. national security systems from threats posed by CRQCs [58, 28]. Similarly, the European Union (EU) released its recommendation, “A Coordinated Implementation Roadmap for the Transition to Post-Quantum Cryptography” [22], which provides concrete recommended steps for EU member states. The EU strategically divides its milestones into two target horizons: the end of 2026 and the end of 2030, respectively [22]. The action plan for the first milestone focuses on engaging stakeholders in PQC activities, participating in PQC initiatives through the Network and Information Systems Cooperation Group (NIS CG) workstream, and establishing a deliberate implementation plan [22]. The subsequent milestone involves completing the PQC transition for high-risk use cases and facilitating automated upgrades for quantum-safe software and firmware [22].

3 Problem Statement and Objectives

This chapter reviews prior research on key exchange protocols, both in traditional and multi-party contexts. The chapter also outlines the core contributions of this thesis and formulates the primary research questions.

3.1 Related work

In the realm of cryptography, secure key exchange among multiple participants is a well-studied problem. The keys established through such a process underpin critical operations such as encryption, decryption, rekeying, and session key derivation, meaning any compromise of these keys directly endangers all dependent processes.

The establishment of keys requires a secure key exchange method. Traditionally, keys could be transmitted over secure channels such as private courier or registered mail, however such means are impractical in modern settings. A widely adopted approach for securely establishing keys over insecure channels is the Diffie-Hellman (DH) key exchange. Building on this, Burmester and Desmedt [14] introduced a prominent conference key establishment method based on DH, proposing four topology configurations: star, tree, cyclic, and broadcast systems. Among these, the broadcast system is the most efficient, requiring only two communication rounds while eliminating leader-based computation, thereby distributing the workload evenly and avoiding a single point of failure.

Katz and Yung proposed a compiler-based approach for authenticated group key exchange [29]. This approach provides a general framework for transforming any unauthenticated group key exchange protocol secure against a passive eavesdropper into an authenticated one, though their construction relies on the Decisional Diffie-Hellman (DDH) assumption and remains vulnerable to quantum attacks. In the context of multiparty key exchange, Boneh and Zhandry proposed a multiparty non-interactive key exchange (NIKE) protocol constructed from indistinguishability obfuscation (iO) and pseudorandom generators, achieving key exchange without trusted setup [13]. Their construction nevertheless relies on iO which is a largely theoretical primitive with no known efficient and practical instantiation, limiting its real-world applicability. Alwen et al. introduced mKyber which is a multi-recipient KEM based on Kyber, where a single compact cipher-

text can be decapsulated by multiple recipients [4]. Their work is fundamentally a key encapsulation primitive rather than a full contributory group key agreement protocol, though the authors highlight its application to reducing bandwidth in existing group key agreement protocols such as Message Layer Security (MLS).

In the post-quantum setting, Apon et al. [5] constructed a constant-round unauthenticated group key exchange based on the Ring Learning with Errors (RLWE) assumption. The resulting protocol can be authenticated via the Katz-Yung compiler using any post-quantum signature scheme, though no empirical implementation or evaluation is presented. On the implementation side, Escribano Pablos and González Vasco [21] implemented a post-quantum group authenticated key exchange protocol based on Kyber, demonstrating practical feasibility up to 2000 entities. However, their construction predates NIST standardization of ML-KEM, avoids post-quantum signatures entirely, and does not evaluate cryptographic agility across multiple algorithm combinations.

Therefore, there is a research gap motivated by the need for a practical and PQ-secure protocol to establish a conference key, particularly in decentralized environments where no group participant bears a significantly heavier computational workload than the others.

3.2 Objectives and Research Questions

The primary goal of this thesis is to design a MP-KEX protocol that establishes a conference key in contributory and symmetric manner, and to evaluate its performance. The protocol performs key exchanges with digital signatures for message authentication. The benchmarks are evaluated across four algorithm combinations: ECDH+ECDSA, ECDH+ML-DSA, ML-KEM+ECDSA, and ML-KEM+ML-DSA, providing a direct comparison between classical and post-quantum approaches.

This thesis addresses three main Research Questions (RQ):

- RQ1: To what extent does the MP-KEX protocol satisfy the security requirements of a post-quantum contributory group key agreement protocol?
- RQ2: How do classical and post-quantum algorithm combinations compare in terms of computational efficiency within the MP-KEX protocol?
- RQ3: How does the MP-KEX protocol scale in terms of computational complexity as the number of participants increases?

4 Preliminaries

This chapter explains the foundations of cryptography and provides a systematic taxonomy of the discipline. It presents technical overviews of selected cryptographic algorithms and provides a critical synthesis of hybrid models and cryptographic agility.

4.1 Cryptography

Historically, cryptography was viewed as an art of secret writing [18]. Although its formal definition has evolved over time, its foundational objective remains immutable: ensuring secure communications and protecting data both at rest and in transit. In the contemporary paradigm, cryptography is the systematic practice and study of methods and frameworks to protect information against adversarial intervention.

Figure 4.1 presents the taxonomy of cryptography. At the root level, cryptography is trifurcated into three core categories: symmetric systems, asymmetric systems, and cryptographic hash functions. While both symmetric and asymmetric cryptography require cryptographic keys, the hash functions operate as keyless primitives. The classification of classical cryptography often varies depending on whether a cryptographic key serves as the taxonomic anchor. This thesis adopts a classification based on the objectives of cryptography, namely, to secure communications and protect data, thereby including hash functions as a part of cryptography.

Asymmetric cryptography is categorized into Key Exchange (KEX)/ Key Encapsulation Mechanisms (KEMs) and Digital Signature Algorithms (DSAs). Both categories include classical and PQC variants. Since most widely deployed classical primitives are quantum-vulnerable algorithms, PQC aims to address two paramount vulnerabilities within asymmetric cryptography upon the emergence of a CRQC: namely, securing key establishment mechanisms and hardening digital signatures which correspond to PQC under KEMs and PQC under DSAs, respectively. The algorithms highlighted in the green boxes are used in our protocol implementations.

Figure 4.2 illustrates the key cryptographic goals, conceptually derived from the cryptographic objectives delineated by Delfs and Knebl [18]. The outer nodes of the model

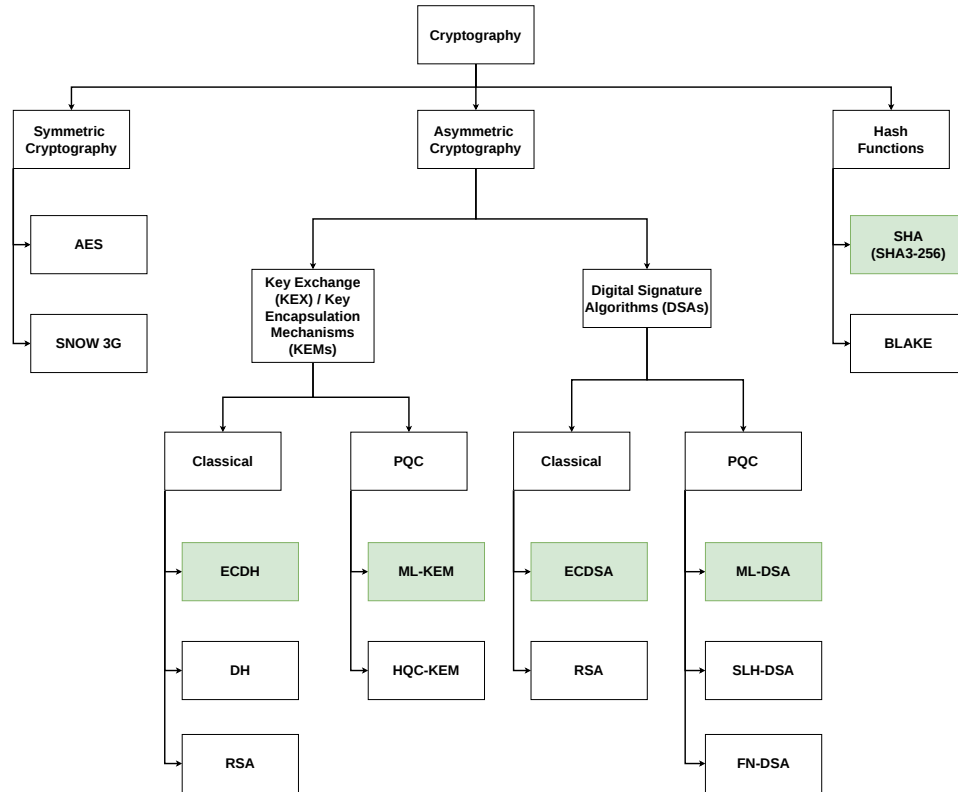


Figure 4.1: Taxonomy of Cryptography

represent the following core objectives:

- **Authenticity:** Property that allows a party to verify the identity of the other party and ensures the origin of the message.
- **Confidentiality:** Property in which data are kept secret from unauthorized access. It encompasses the property of secrecy, which refers to the protection of sensitive information such as ephemeral session keys or long-term master keys.
- **Integrity:** Property that ensures the original message has not been tampered with or altered in any unauthorized way.
- **Non-repudiation:** Property that prevents a party from denying their signature or the origin of the message.

However, achieving all objectives at once is usually not necessary, and for many contexts, it suffices to ensure a subset of above objectives. Thus, practitioners often prioritize certain properties depending on scalability and application use-cases.

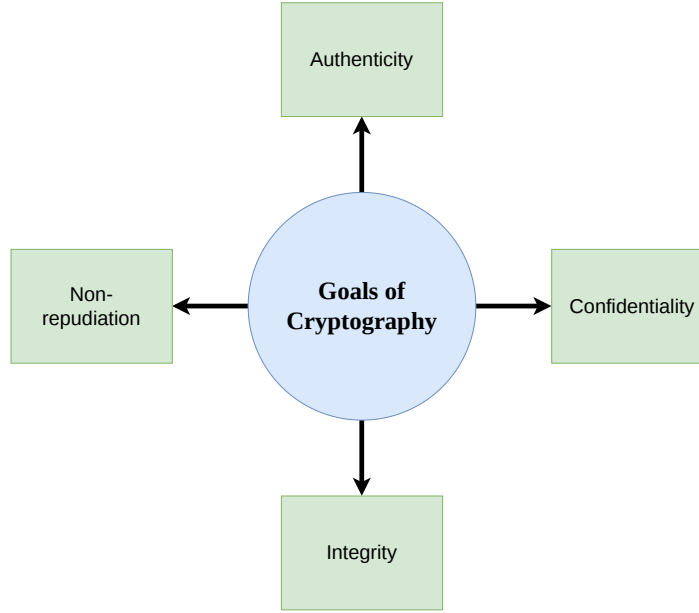


Figure 4.2: Objectives of Cryptography [18]

4.1.1 Asymmetric Cryptography

Asymmetric cryptography, also known as public key cryptography, is a primitive in which each party possesses a pair of keys, a public key pk and a private key sk . Each party secretly possesses their private key while a public key is publicly available to others.

Table 4.1 presents three main ways to use asymmetric cryptography. In a public-key encryption scheme, pk is used to encrypt a message and sk is used for decryption. In a digital signature scheme, sk is used to sign a message and pk is used to verify the signature. For key establishment, parties exchange public parameters and combine their own sk with the other party's pk to derive a shared secret key k .

Table 4.1: Comparison of Asymmetric Cryptographic Primitives

Feature	Encryption	Digital Signature	Key Establishment
Public Key	Encrypts the message	Verifies the signature	Exchanges parameters
Private Key	Decrypts the ciphertext	Signs the message	Derives shared secret
Goal	Confidentiality	Authenticity & Data Integrity	Secure key setup
Algorithms	RSA, ECIES, ElGamal	RSA, ECDSA, EdDSA	DH, ECDH

Next we present two well-established algorithms that are used in the experiments of the thesis.

X25519

X25519 is an Elliptic Curve Diffie-Hellman (ECDH) key agreement function built on Curve25519, introduced by Bernstein and standardized in Request for Comments (RFC) 7748 [10, 33]. It allows two or more parties to establish a shared secret key over an insecure communication channel while this secret does not need to ever be transmitted. It is widely adopted in cryptographic protocols such as TLS 1.3, SSH, and the Signal protocol due to its strong security guarantees and computational efficiency [62, 2, 73]. NIST formally recognized Curve25519 as an approved elliptic curve for key agreement in SP 800-186 [16]. The security of X25519 relies on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP) over Curve25519, which provides approximately 128-bit security level [33, 8].

ECDSA-P256

The Elliptic Curve Digital Signature Algorithm (ECDSA) is a digital signature algorithm based on Elliptic Curve Cryptography (ECC) and is standardized by NIST in FIPS 186 [45]. It allows a signer to produce a digital signature using a private key, which can be verified by any party holding the corresponding public key. The P-256 curve is included among the approved elliptic curve in SP 800-186 [16]. Its security relies on the hardness of the ECDLP over P-256, providing approximately 128-bit security level [8]. ECDSA with P-256 is widely adopted in protocols such as TLS 1.3 [73].

4.1.2 Symmetric-Key Cryptography

Symmetric-key cryptography covers primitives in which the same secret key is used for both the encryption and decryption processes.

Encrypting a message is performed as follows:

$$c = E(m, k) \tag{4.1}$$

where m is the message, k is a secret key, E is the encryption algorithm, and c is the ciphertext.

The decryption of a ciphertext c is carried out as follows:

$$m = D(c, k) \quad (4.2)$$

where c is the ciphertext, k is a secret key, D is the decryption algorithm, and m is the recovered message.

In both the encryption and decryption phases, the algorithms D and E are publicly known, and therefore the secret key k must be kept confidential.

The parties first agree on a shared secret key k , which is typically obtained by applying asymmetric cryptography or through a secure channel such as certified mail or private delivery carriers [18, 19]. Examples of symmetric-key cryptographic algorithms are the Advanced Encryption Standard (AES) [43] and Snow 3G [1].

In the experiments of this thesis, no symmetric-key cryptographic primitives are utilized.

4.1.3 Hash Functions

The cryptographic hash function is a keyless primitive. Such function takes an arbitrarily long bit string as an input and outputs a bit string of a fixed length deterministically [18]. The function is expressed mathematically as:

$$h : \{0, 1\}^* \rightarrow \{0, 1\}^n, \quad m \mapsto h(m). \quad (4.3)$$

where m denotes the message and $h(m)$ signifies the hash value, digest, whose length typically ranges from 128 to 512 bits [18]. The most important requirement for a cryptographic hash function is that it must be a one-way function, i.e., given a digest y , it is infeasible in practice to find m such that $h(m) = y$.

Hash functions are typically used in digital signature schemes where a message m is hashed into digest $h(m)$, which is signed instead of the message itself [18]. This approach is usually recommended because signing large messages directly is computationally inefficient and impractical.

The hash functions are also used with a secret key in Message Authentication Codes (MACs) [18]. Since the hash function is deterministic and publicly computable, it does not inherently protect data integrity. By incorporating a secret key into MACs, a hash function can ensure integrity and provide authentication for parties who share that secret key.

Furthermore, hash functions are used in Key Derivation Functions (KDFs) to derive one or more cryptographic keys such as session keys, from a master secret or password. KDFs are often used in maintenance tasks such as key rotation, to ensure that updated keys remain mathematically independent and secure throughout the system's lifecycle.

In this thesis, the SHA3-256 hash function is used within the KDF and for authentication purposes, i.e., hashing the message before signing.

SHA3-256

Secure Hash Algorithm 3 (SHA3) is a hash function family standardized by NIST under FIPS 202 in 2015 [55]. It is constructed based on the sponge framework and the KECCAK [11] algorithm, the winner of the SHA-3 competition. SHA3 serves as an alternative to SHA-2 [54] due to its distinct underlying mathematical permutation. SHA3-256 produces a 256-bit digest and ensures 128-bit collision resistance.

4.2 Post-Quantum Cryptography

This section describes the PQC algorithms standardized by the National Institute of Standards and Technology (NIST), which is a governmental agency within the U.S. Department of Commerce. NIST is the preeminent cryptographic standards defining organization and spearheads the standardizations of PQC algorithms that withstand quantum-enabled cryptanalysis. The implementations of this thesis are restricted to PQC algorithms standardized by NIST, although other standards organizations such as the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) standardize their own algorithms. No quantum algorithms are known that could efficiently solve the mathematical problems that the PQC algorithms are based on. NIST underpins two cryptographic schemes for PQC development: Key Encapsulation Mechanisms (KEMs) and Digital Signature Algorithms (DSAs).

Figure 4.3 depicts a typical adoption pathway of the NIST selected PQC algorithms across various standards. Initially, these algorithms are adopted by the Internet Engineering Task Force (IETF), primarily in Transport Layer Security (TLS) and Internet Protocol Security (IPsec). Consequently, these structural choices influence and are integrated in ISO/IEC standards, or wireless standards such as ones defined by the 3rd Generation Partnership Project (3GPP), Global System for Mobile Communications Association (GSMA) and

European Telecommunications Standards Institute (ETSI). Nonetheless, ISO/IEC and wireless standards may need to integrate PQC algorithms also directly into their security architectures.

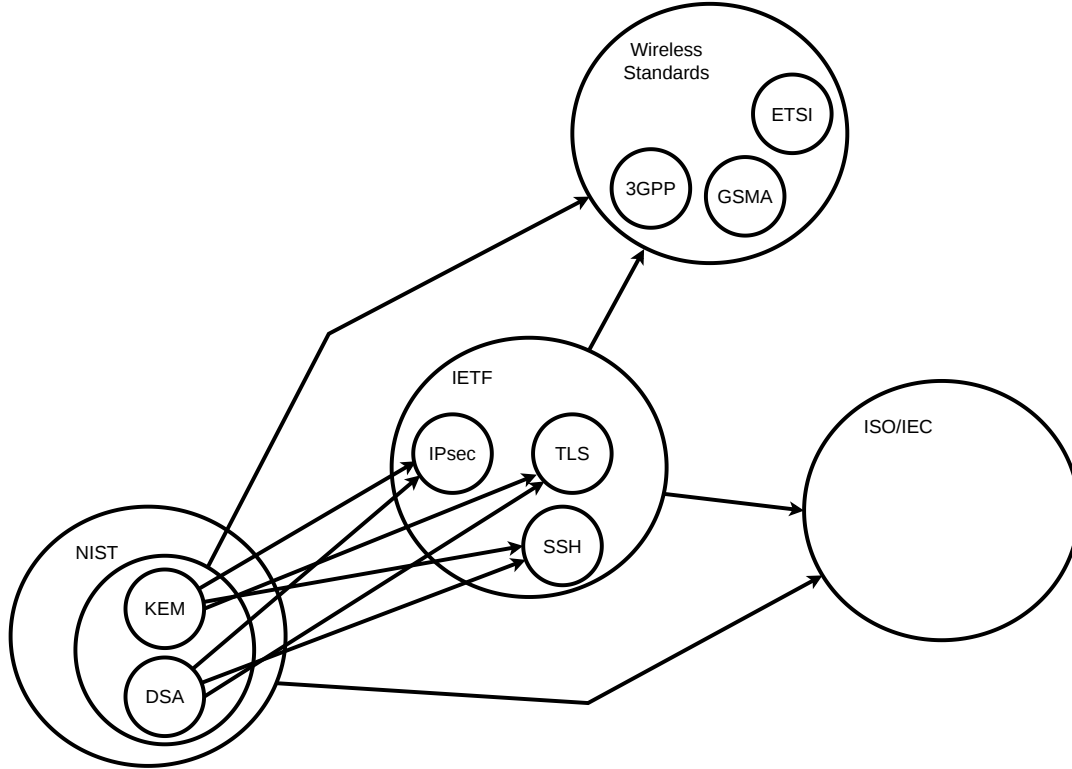


Figure 4.3: Adoption of NIST Selected PQC

Figure 4.4 illustrates the primary mathematical families underpinning the PQC landscape. PQC is mainly categorized into lattice-, code-, hash-, multivariate-, symmetric and isogeny-based cryptography, alongside MPC*-in-the-head signature schemes [52, 53].

Algorithms highlighted with green background represent NIST standardized algorithms, although FN-DSA and HQC have not yet, at the time of writing, been ratified as Federal Information Processing Standards (FIPS) [53]. The rest of the algorithms with blue background correspond to candidate signature algorithms that are undergoing evaluation within a NIST standardization track for additional digital signature algorithms [52]. As of July 2026, NIST has standardized only two KEMs and no candidate KEM is under active evaluation by NIST [53].

*Multi-Party Computation (MPC) is a cryptographic technique that enables multiple participants to jointly compute a function over their private inputs without revealing those inputs to one another beyond what can be inferred from the output.

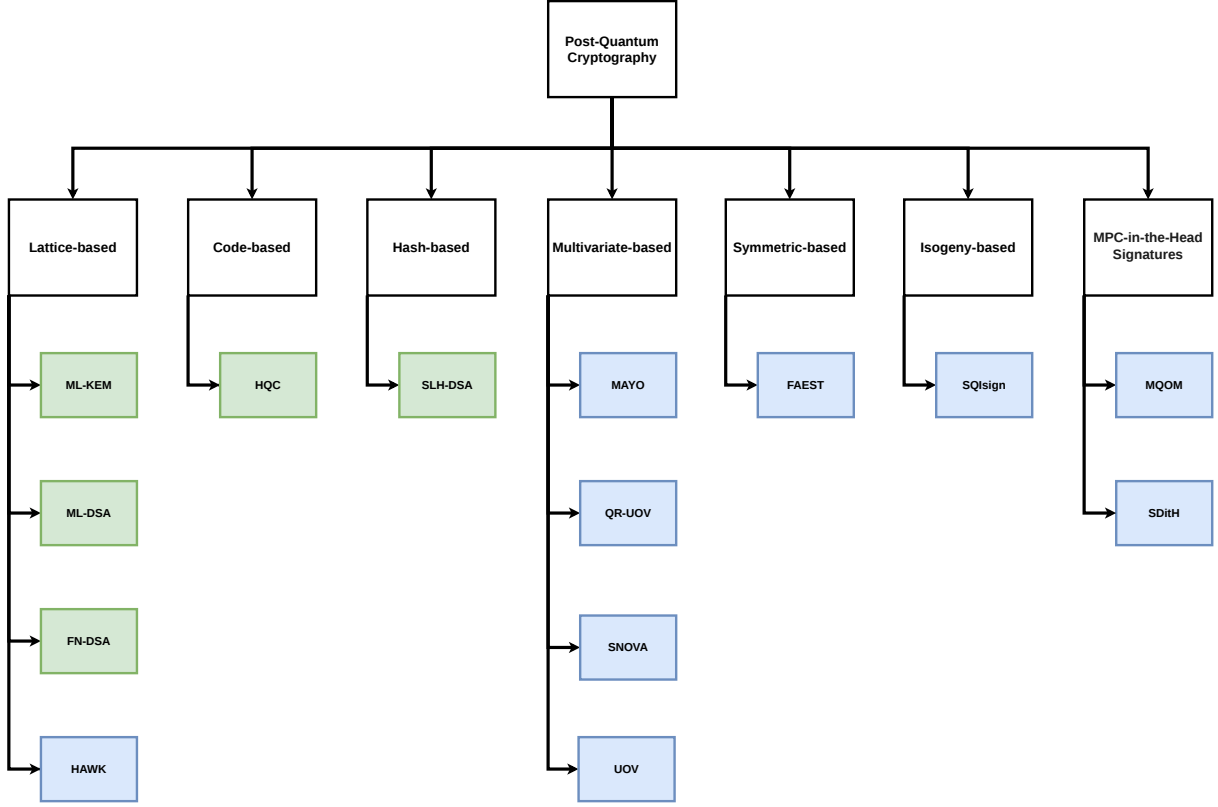


Figure 4.4: Structural Families and Corresponding Schemes of PQC under NIST [52, 53]

Table 4.2 highlights the NIST security levels, security benchmarks, and the corresponding PQC algorithms used in this thesis. These security levels are based on the computational resources required to break key search in block cipher and collision search in hash function [51]. In the table, only the algorithms implemented in the thesis are listed and other NIST-standardized PQC algorithms are excluded.

Table 4.2: NIST PQC Security Levels, Security Benchmarks and Algorithms Mapping [51]

Level	Security Benchmark	Associated PQC Algorithms
1	AES-128 key search	ML-KEM-512
2	SHA-256/SHA3-256 collision search	ML-DSA-44
3	AES-192 key search	ML-KEM-768, ML-DSA-65
4	SHA-384/SHA3-384 collision search	N/A
5	AES-256 key search	ML-KEM-1024, ML-DSA-87

4.2.1 Key Encapsulation Mechanisms

The Key Encapsulation Mechanism (KEM) is a cryptographic scheme for securely establishing a shared secret key between two communication parties over an insecure channel. Conceptually, a KEM uses an asymmetric cryptography property for key encapsulation and the resulting key is subsequently used in symmetric cryptography [3]. The primary goals of KEM are to ensure the confidentiality and data integrity.

In a standard key exchange between two parties, Alice and Bob, the protocol is executed as follows:

1. **Preparation:** Alice and Bob fix a set of parameters prior to the key establishment. This set is either selected by Alice, by Bob, or by negotiation between Alice and Bob.
2. **Key generation:** Alice executes the probabilistic key-generation algorithm in her cryptographic module to create a pair of encapsulation ek_A and decapsulation dk_A keys.
3. Alice keeps dk_A private and sends ek_A to Bob.
4. **Encapsulation:** Bob receives ek_A from Alice and runs a probabilistic encapsulation algorithm in his cryptographic module to output a shared secret key K_B and the ciphertext of the shared secret key c_B .
5. Bob keeps K_B private and sends c_B to Alice.
6. **Decapsulation:** Alice receives c_B from Bob and runs a deterministic decapsulation algorithm in her cryptographic module to obtain a shared secret key K_A , which is kept private by Alice.

At the end of the KEM, Alice and Bob derive identical shared keys K_A and K_B , respectively, where $K_A = K_B$. This shared secret key can be used for subsequent symmetric-key cryptography.

KEM is a unidirectional, i.e., sender sends the ciphertext of the shared secret key c to receiver, which is decapsulated to obtain shared secret key K by receiver. Running KEM alone does not provide mutual authentication or key confirmation step. Thus, such verification must be operated in higher level protocols. NIST has documented recommendations

for KEM usage in wider applications such as for key confirmation, for authenticated key establishment or how KEM should be integrated in the protocols [3].

As of July 2026, NIST has selected two KEMs: ML-KEM [49] and HQC-KEM [47]. In the implementation of this thesis, ML-KEM is used for key establishment.

ML-KEM

Module-Lattice-Based Key Encapsulation Mechanism (ML-KEM) [49] is a KEM based on the CRYSTALS-Kyber algorithm. NIST selected ML-KEM in July 2022 and standardized under the Federal Information Processing Standard (FIPS) 203 in August 2024 [50]. ML-KEM achieves Indistinguishability under Adaptive Chosen Ciphertext Attack (IND-CCA2)* security. It is based on the computational difficulty of the Module Learning with Errors (MLWE) problem [34], which is a structured generalization of the Learning with Errors (LWE) problem [66]. The LWE problem deals with a system of noised linear equations with small errors that make the equations intractable with methods such as Gaussian elimination. MLWE extends this property of LWE problem by utilizing module lattices, where equations are formed with polynomial rings instead of integers. The polynomial multiplications are computed efficiently employing a Number Theoretic Transform (NTT). NIST provides three parameter sets for ML-KEM [49, 44].

4.2.2 Digital Signature Algorithms

The Digital Signature Algorithm (DSA) is a cryptographic scheme that allows to prove the authenticity of digital messages and ensures data integrity. The valid digital signature provides a solid proof to verifier that the message was generated by a known signer.

In a standard communication between signer (Alice) and verifier (Bob), the protocol is executed as follows:

Key Generation: Alice executes a probabilistic key-generation algorithm to produce a key pair consisting of a private signing key sk and a public verification key pk . The private key sk is used to sign messages and must be kept secret, while the public key pk is shared openly and used by Bob to verify the digital signatures generated by Alice. The same key pair may be used to sign and verify multiple messages.

*A property where an adversary, who can access a decryption oracle, cannot distinguish a shared secret from randomly sampled value

Signer Side (Alice):

1. **Hashing:** Before signing a message m , Alice processes m through a deterministic hash function to obtain a fixed-length message digest $h(m)$.
2. **Signing:** Alice runs a probabilistic signing algorithm using her sk and $h(m)$ to produce a digital signature σ .
3. Alice sends m and the digital signature σ to Bob.

Verifier Side (Bob):

1. **Message Processing:** After receiving a message m' and a digital signature σ' , Bob independently runs the same deterministic hash function on m' to obtain $h(m')$.
2. **Verification:** Bob runs a deterministic verification algorithm using Alice's public key pk , signature σ' , and the computed digest $h(m')$.
3. **Validation:** The verification algorithm returns a boolean result: if the result is True, Bob accepts the signature as authentic; otherwise, the message is rejected.

A prerequisite for the verification step is that Bob must possess Alice's authentic public key pk . In practice, this is achieved through a Public Key Infrastructure (PKI), where public keys are bound to identities via digitally signed certificates issued by a trusted Certificate Authority (CA) [12].

As of July 2026, NIST has selected three PQ DSAs: ML-DSA [48], SLH-DSA [56] and FN-DSA [46]. In the implementation of this thesis, ML-DSA is used for authentication purposes.

ML-DSA

The Module-Lattice-Based Digital Signature Algorithm (ML-DSA) [48] is based on the CRYSTALS-Dilithium scheme [38]. NIST selected ML-DSA in July 2022 and standardized under FIPS 204 in August 2024. ML-DSA achieves Strong Existential Unforgeability under Chosen Message Attack (SUF-CMA)*. ML-DSA inherits the security based on MLWE problem [34] and the unique version of the Module Short Integer Solution (MSIS) problem

*A property wherein an adversary, having seen a valid signature for a given message, cannot produce a different valid signature for that same message

called SelfTargetMSIS [30]. By applying “Fiat-Shamir with Aborts” framework, ML-DSA implements a rejection sampling to effectively mask a private key [36, 37]. ML-DSA employs the Number Theoretic Transform (NTT) to efficiently compute polynomial multiplications. NIST provides three parameter sets for ML-DSA [48].

4.3 Hybrid Approach

The hybrid approach refers to the combinations of PQC algorithms with classical algorithms. As most of the cryptographic components built upon RSA and ECC remain secure until Shor’s algorithm can feasibly compromise them, the hybrid approach provides a dual layer of security, i.e., one based on classical algorithms and another on PQC algorithms. Although this approach enhances security, the drawback of this approach is the increased computational complexity and performance cost because two distinct types of cryptographic algorithms must be executed instead of just one.

For key establishment, parties execute two separate protocols to generate two independent shared keys, which are then concatenated to derive the unified key [71, 67]. Typically, a KDF is applied to this concatenated value to generate a master secret for symmetric encryption algorithms, such as AES. Consequently, breaking the master secret key requires an adversary to solve both the classical and the PQ computational hardness problems to recover the constituent keys. Current internet-drafts [32, 71] propose hybrid key exchange mechanisms for TLS 1.3 by combining keys obtained from ECDH and ML-KEM algorithms. However, these implementations may require substantial updates to existing cryptographic libraries and may encounter challenges regarding message size [65]. The resulting hybrid key may exceed the Path Maximum Transmission Unit (Path MTU), necessitating packet fragmentation, which increases the likelihood of retransmissions and protocol latency [65, 71].

In a digital signature scheme, the hybrid approach involves signing the message to generate “dual signatures” [23]. To verify authenticity, the verifier must validate both signatures using their respective verification algorithms. The pitfall of this approach is the doubling of both computational cost and total signature length. These larger artifacts often exceed the MTU, the largest packet size of a data packet that can be sent on a network, which implies the need of packet fragmentation and complicates authentication protocols [64].

4.4 Cryptographic Agility

Cryptographic agility, commonly known as crypto-agility, is a property of systems, protocols, and applications to smoothly update or replace cryptographic mechanisms. Modern cryptographic products, protocols, and services rely on public-key cryptography. These entities require a proactive migration path toward PQC algorithms [17].

When adopting algorithms, practitioners should design components to be flexible so that algorithms can be swapped or parameters adjusted seamlessly. Cryptographic components must be resistant to downgrade attacks where an adversary attempts to force the selection of an algorithm that is weaker than what is intended [17]. The Mandatory-To-Implement (MTI) algorithm sets should be revised to identify and phase out weakened algorithms. The new MTIs must be introduced before the previous MTIs face obsolescence [9]. As a best practice, to ensure interoperability, protocols should maintain support for multiple algorithms concurrently during the migration process [26].

Protocol agility enables security protocols to incorporate new cryptographic algorithms through modular design and authenticated negotiation mechanisms [9]. By decoupling base protocol specifications from specific algorithm profiles, protocols can independently update MTI sets and accommodate evolving parameter requirements without structural failure [9, 26].

5 Protocol Design of MP-KEX

This chapter describes the Multi-Party Key Exchange (MP-KEX) protocol for establishing an initial conference key, including its assumptions and the respective execution rounds. The protocol is inspired by the conference key agreement protocol proposed by Burmester and Desmedt [14]. MP-KEX distributes the computational workload uniformly across all participants, ensuring that no single participant bears a disproportionate burden. In the implementation of this thesis, the MP-KEX protocol described in this chapter is run on a single machine, for the sake of simplicity.

Figure 5.1 illustrates the pipeline of the MP-KEX protocol. In Round 0 (see Section 5.2), participants undergo the setup phase, involving local key generation and establishing logical topology alignment. In Round 1 (see Section 5.3), each participant signs their public key, which is verified by their two neighbors. Then, an intermediate shared secret key is created with these two immediate neighbors. Following this, in Round 2 (see Section 5.4), each participant computes a value called a bridge and broadcasts it to the group, which value is then verified by the intended participants. Finally, in Round 3 (see Section 5.5), each participant derives and confirms the conference key.

The communication steps described in this chapter are conceptual and represent the protocol behavior in a distributed deployment. In the implementation presented in this thesis, neither KEX/KEM public keys nor bridges are transmitted over the network. Instead, these protocol messages are represented by local data structures and verified using the corresponding public signature verification keys.

5.1 Assumptions of the MP-KEX

The MP-KEX protocol assumes synchronous setting where participants are active and the group agrees to derive a conference key. Before executing the MP-KEX protocol, all participants are required to know the total number of participants, each others' signature verification keys, and the specific algorithms used, including ECDH, KEM, DSA and hashing.

Table 5.1 outlines the cryptographic algorithms deployed across the different execution

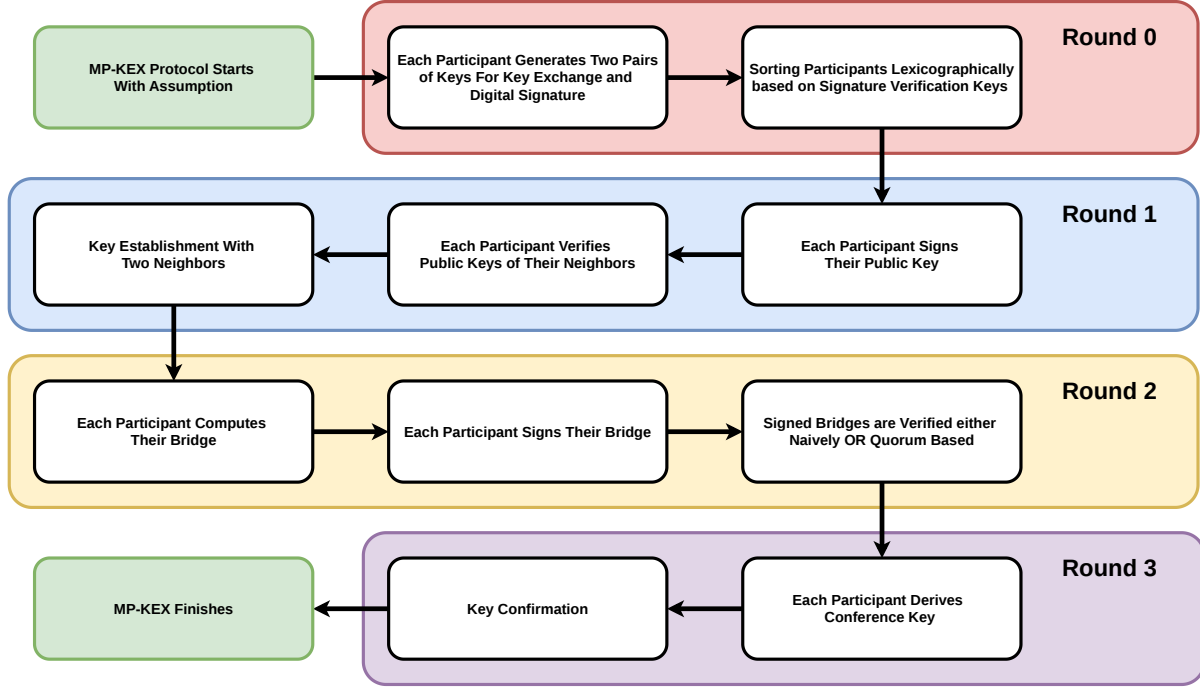


Figure 5.1: Pipeline of MP-KEX

rounds of the MP-KEX protocol. Algorithms highlighted in orange signify classical primitives, whereas those highlighted in blue indicate PQC algorithms. The hash function SHA3-256 is used exclusively in Round 3 as the basis for a Key Derivation Function (KDF) to find the final conference key.

The inclusion of quantum-vulnerable algorithms provides a vital security safety net. Because PQC standards are relatively new and lack extensive real-world deployment history, merging with battle-tested classical primitives creates a robust hybrid architecture. This approach maintains high-level security by leveraging time-proven standards as a safety net, ensuring protection against potential vulnerabilities in newer PQC algorithms until the emergence of CRQCs that can compromise classical primitives.

Table 5.1: Algorithms Used in the MP-KEX Protocol

Algorithms	Round 0	Round 1	Round 2	Round 3
KEX/KEMs	ECDH / ML-KEM	ECDH / ML-KEM		
DSAs	ECDSA / ML-DSA	ECDSA / ML-DSA	ECDSA / ML-DSA	ECDSA / ML-DSA
Hash Function				SHA3-256

5.2 Setup and Topology Alignment (Round 0)

Algorithm 1 presents the pseudocode for the MP-KEX setup phase. First, the algorithm requires the input parameters: the number of participants n , the key exchange/encapsulation mode M_{ke} , and the digital signature scheme M_{sig} . These modes determine whether classical or PQC algorithms are used. For modes M_{ke} and M_{sig} , a value of 0 specifies the use of a quantum-vulnerable algorithm, while a value of 1 indicates a PQC algorithm. For each participant, the algorithm generates the appropriate KEX/KEM and signature key pairs based on the selected modes. During each iteration, the resulting key tuple is appended to the registry $\mathbf{P}$.

Throughout the entire MP-KEX protocol, the operating modes M_{ke} and M_{sig} remain unchanged. For instance, if participants agree to utilize KEX for establishing key and PQC-based DSA for authentication, i.e., $M_{ke} = 0$ and $M_{sig} = 1$, respectively, then these parameters persist until the key derivation phase (see Section 5.5).

Figure 5.2 illustrates the ordering of participants (represented by green boxes) and Algorithm 2 is the pseudocode for a sorting mechanism which is run on a single machine. Initially, there are participants who want to create a conference key among themselves. All participants are lexicographically sorted based on their public signature verification keys. This sorting works regardless of which digital signature algorithm is used. After ordering, every participant determines their sequential index based on the sorted public key registry $\mathbf{P}$, representing their position in the ring topology. While the registry $\mathbf{P}$ is created and stored centrally during this simulation, this is just an architectural abstraction for the purpose of the analysis. In real-world deployments, private keys remain exclusively in the possession of their respective participants and are never accessible to outsiders.

Likewise, although the sorting computation is presented as a centralized process for clarity, this is also an architectural abstraction. In real-world deployments, each participant independently executes the sorting algorithm using public signature keys. Since the sorting is deterministic and based on public signature verification keys, every participant arrives at the same global ordering locally, thereby eliminating the need for a persistent leader or central coordinator to manage the topology.

Algorithm 1 Local Key Generation (Round 0 - Step 1)

Input :Number of participants n Key Exchange/Encapsulation Mode $M_{ke} \in \{0 : \text{KEX}, 1 : \text{KEM}\}$ Signature Scheme Mode $M_{sig} \in \{0 : \text{DSA}, 1 : \text{PQDSA}\}$ **Output:**Registry $\mathbf{P} = \{T_0, \dots, T_{n-1}\}$ where key tuple $T_i = (pk_{sig}, sk_{sig}, pk_{ke}, sk_{ke})$ for participant P_i and $n = |\mathbf{P}|$

```

1  $\mathbf{P} \leftarrow \emptyset$ 
2 for  $i = 0$  to  $n - 1$  do
3   if  $M_{sig} == 1$  then
4      $(pk_{sig}, sk_{sig}) \xleftarrow{\$} \text{PQDSA.KeyGen}()$ 
5   else
6      $(pk_{sig}, sk_{sig}) \xleftarrow{\$} \text{DSA.KeyGen}()$ 
7   if  $M_{ke} == 1$  then
8      $(pk_{ke}, sk_{ke}) \xleftarrow{\$} \text{KEM.KeyGen}()$ 
9   else
10     $(pk_{ke}, sk_{ke}) \xleftarrow{\$} \text{KEX.KeyGen}()$ 
11     $T_i \leftarrow (pk_{sig}, sk_{sig}, pk_{ke}, sk_{ke})$ 
12     $\mathbf{P} \leftarrow \mathbf{P} \cup \{T_i\}$ 
13 return  $\mathbf{P}$ 

```

Algorithm 2 Global Topology Registry Assignment (Round 0 - Step 2)

Input :Registry $\mathbf{P} = \{T_0, \dots, T_{n-1}\}$ where key tuple $T_i = (pk_{sig}, sk_{sig}, pk_{ke}, sk_{ke})$ for participant P_i and $n = |\mathbf{P}|$ **Output:**Sorted global participant topology registry $\mathbf{P}$

```

1  $\mathbf{P} \leftarrow \text{Sort}(\mathbf{P} \text{ lexicographically by } pk_{sig})$ 
2 return  $\mathbf{P}$ 

```

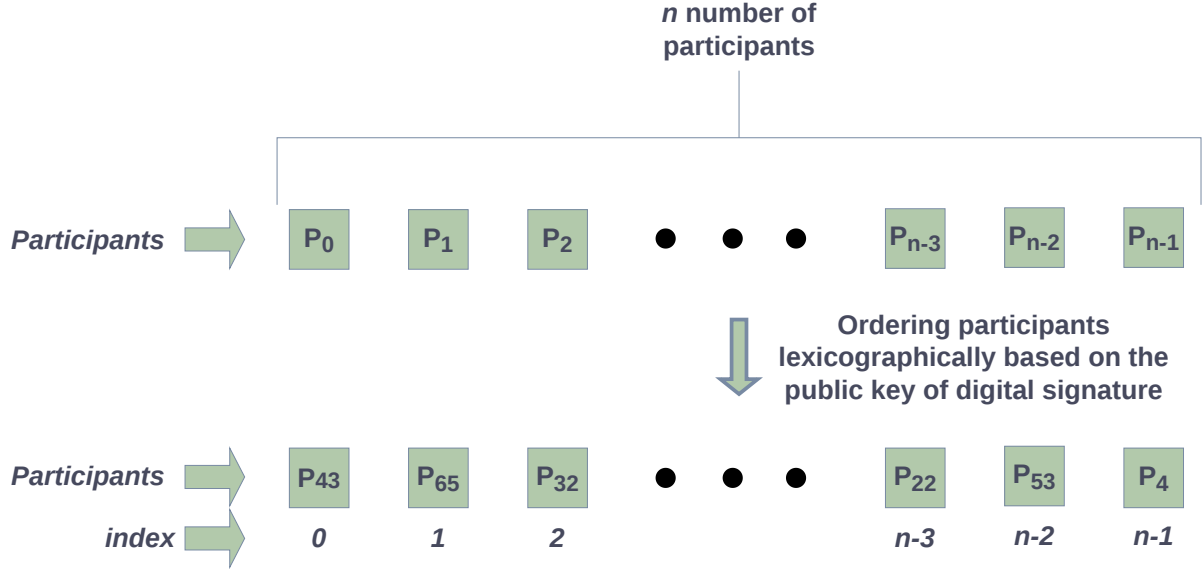


Figure 5.2: Ordering of Participants

5.3 Key Establishment With Neighbors (Round 1)

Once the participants are ordered based on the public signature keys, every participant can determine their immediate neighbors, as described in Algorithm 2.

Figure 5.3 shows the ordered topological arrangement of the participants. For visualization purposes, the ring illustrates the immediate neighbors of any given participant, shown by bidirectional arrows. For example, participant P_0 has two immediate neighbors P_1 (on the right-hand side) and P_{n-1} (on the left-hand side). The three black dots positioned at the bottom left serve as a shorthand notation to represent an arbitrary number of intermediate participants.

Algorithm 3 presents the pseudocode for signing and verifying public keys in Round 1. Initially, participant P_i signs their respective KEX/KEM public keys to generate a digital signature and sends the resulting signature together with the public key to their immediate neighbors (both left and right neighbors). Then, these signatures are verified by the receiving neighbors. If any verification fails, the protocol aborts ($\perp$).

Algorithm 4 is the pseudocode showing how each participant executes cryptographic primitives according to the values of M_{ke} . If M_{ke} equals 1, then the protocol proceeds via KEM. In this case, each participant encapsulates a symmetric seed using the right neighbor's public key $\mathbf{P}[i+1].pk_{ke}$, outputting a ciphertext $C_{i,i+1}$ and a shared secret key $K_{i,i+1}$. The ciphertext $C_{i,i+1}$ is signed with the participant's private key $\mathbf{P}[i].sk_{sig}$ to yield σ_i . The

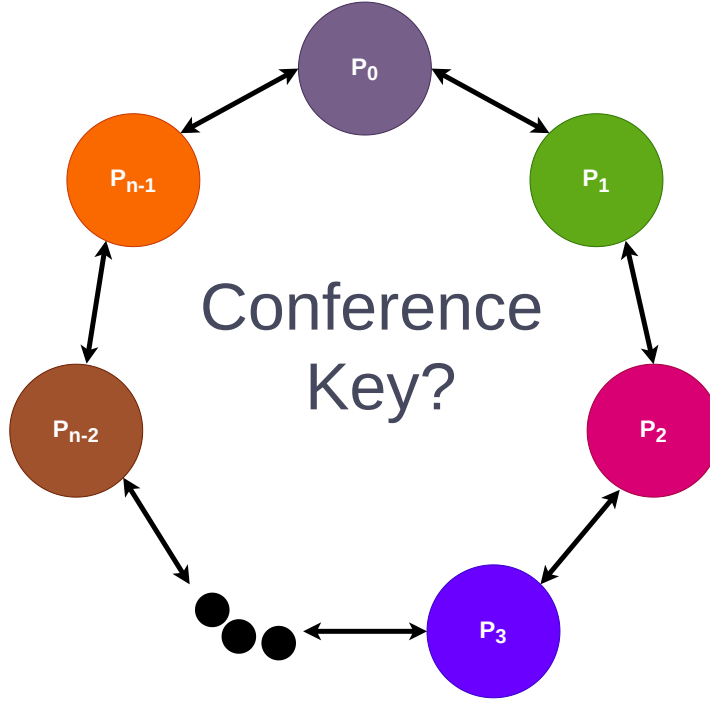


Figure 5.3: Topological Arrangement of Participants

Algorithm 3 Signing and Verifying Public Key (Round 1 - Step 1)

Input :Key Exchange/Encapsulation Mode $M_{ke} \in \{0 : \text{KEX}, 1 : \text{KEM}\}$ Sorted registry $\mathbf{P} = \{T_0, \dots, T_{n-1}\}$ where key tuple $T_i = (pk_{sig}, sk_{sig}, pk_{ke}, sk_{ke})$ for participant P_i and $n = |\mathbf{P}|$

```

1  $n = |\mathbf{P}|$ 
2 for  $i = 0$  to  $n - 1$  do
3   Set  $L = (i - 1 + n) \pmod n$ 
4   Set  $R = (i + 1) \pmod n$ 
5    $\sigma_i \leftarrow \text{Sign}_{\mathbf{P}[i].sk_{sig}}(\mathbf{P}[i].pk_{ke})$ 
6   Send  $(\mathbf{P}[i].pk_{ke}, \sigma_i)$  to left neighbor  $P_L$  and right neighbor  $P_R$ 
7 for  $i = 0$  to  $n - 1$  do
8   Set  $L = (i - 1 + n) \pmod n$ 
9   Set  $R = (i + 1) \pmod n$ 
10  Receive  $(\mathbf{P}[L].pk_{ke}, \sigma_L)$  from neighbor  $P_L$  and  $(\mathbf{P}[R].pk_{ke}, \sigma_R)$  from neighbor  $P_R$ 
11  if  $(\text{Verify}_{\mathbf{P}[L].pk_{sig}}(\mathbf{P}[L].pk_{ke}, \sigma_L) \neq 1) \vee (\text{Verify}_{\mathbf{P}[R].pk_{sig}}(\mathbf{P}[R].pk_{ke}, \sigma_R) \neq 1)$  then
12    return  $\perp$ 

```

signature σ_i and ciphertext $C_{i,i+1}$ are sent to the right neighbor P_{i+1} . Next, the right neighbor P_{i+1} verifies the incoming signature σ_i using the sender's public key $\mathbf{P}[i].pk_{sig}$. If the signature is valid, then the right neighbor P_{i+1} decapsulates the ciphertext $C_{i,i+1}$ to recover the intermediate shared secret key $K_{i,i+1}$; otherwise, the protocol aborts.

If a value of M_{ke} is 0, participant P_i computes two intermediate shared secret keys, $K_{i-1,i}$ and $K_{i,i+1}$, directing using the ECDH function with their immediate neighbor's public keys, $\mathbf{P}[i-1].pk_{ke}$ and $\mathbf{P}[i+1].pk_{ke}$. Finally, regardless of the mode, every participant successfully establishes a pair of unique intermediate secret keys, $K_{i-1,i}$ and $K_{i,i+1}$, shared with their left P_{i-1} and right P_{i+1} neighbors.

Algorithm 4 Key Exchange/Encapsulation Mechanism (Round 1 - Step 2)

Input :

Key Exchange/Encapsulation Mode $M_{ke} \in \{0 : \text{KEX}, 1 : \text{KEM}\}$

Sorted registry $\mathbf{P} = \{T_0, \dots, T_{n-1}\}$ where key tuple $T_i = (pk_{sig}, sk_{sig}, pk_{ke}, sk_{ke})$ for participant P_i and $n = |\mathbf{P}|$

```

1   $n = |\mathbf{P}|$ 
2  if  $M_{ke} == 1$  then
3      for  $i = 0$  to  $n - 1$  do
4          Set  $R = (i + 1) \pmod{n}$ 
          /* Encapsulate key for right neighbor */
5           $(C_{i,R}, K_{i,R}) \xleftarrow{\$} \text{Encaps}_{\mathbf{P}[R].pk_{ke}}()$ 
6           $\sigma_i \leftarrow \text{Sign}_{\mathbf{P}[i].sk_{sig}}(C_{i,R})$ 
7          Send  $(C_{i,R}, \sigma_i)$  to right neighbor  $P_R$ 
8      for  $i = 0$  to  $n - 1$  do
9          Set  $L = (i - 1 + n) \pmod{n}$ 
          /* Process inbound ciphertext C from left neighbor */
10         Receive  $(C_{L,i}, \sigma_L)$  from left neighbor  $P_L$ 
11         if  $\text{Verify}_{\mathbf{P}[L].pk_{sig}}(C_{L,i}, \sigma_L) \neq 1$  then
12             return  $\perp$ 
13          $K_{L,i} \leftarrow \text{Decaps}_{\mathbf{P}[i].sk_{ke}}(C_{L,i})$ 
14 else
15     for  $i = 0$  to  $n - 1$  do
16         Set  $L = (i - 1 + n) \pmod{n}$ 
17         Set  $R = (i + 1) \pmod{n}$ 
18          $K_{L,i} \leftarrow \text{ECDH}(\mathbf{P}[i].sk_{ke}, \mathbf{P}[L].pk_{ke})$ 
19          $K_{i,R} \leftarrow \text{ECDH}(\mathbf{P}[i].sk_{ke}, \mathbf{P}[R].pk_{ke})$ 

```

5.4 Bridges (Round 2)

After completing Round 1, each participant P_i possesses two intermediate shared secret keys, $K_{i-1,i}$ and $K_{i,i+1}$. Using these intermediate shared secret keys, each participant P_i computes a bridge value B_i based on the steps described in Algorithm 5. This bridge acts as a cryptographic link that binds the secrets of adjacent neighbors together, allowing the independent links to eventually telescope into a single global key (see Section 5.5). There are two ways of computing a bridge depending on the key exchange/encapsulation mode M_{ke} .

1. **If $M_{ke} = 0$:** In the additive group over an elliptic curve, the bridge is defined as the group subtraction of the left-hand point from the right-hand point:

$$B_i = K_{i,i+1} - K_{i-1,i} = K_{i,i+1} + (-K_{i-1,i}) \quad (5.1)$$

2. **If $M_{ke} = 1$:** In the bitwise space, the bridge is computed as the bitwise XOR of the two neighboring shared secret keys:

$$B_i = K_{i,i+1} \oplus K_{i-1,i} \quad (5.2)$$

Algorithm 5 Bridge Computation (Round 2 - Step 1)

Input :

Number of participants n

Vector of intermediate shared secret keys $\mathbf{K} = (K_{0,1}, K_{1,2}, \dots, K_{n-1,0})$

Key Exchange/Encapsulation Mode $M_{ke} \in \{0 : \text{KEX}, 1 : \text{KEM}\}$

Output:

Vector of bridge values $\mathbf{B} = (B_0, \dots, B_{n-1})$

1 Initialize $\mathbf{B}$ as a vector of size n

2 **for** $i = 0$ **to** $n - 1$ **do**

3 **if** $M_{ke} == 0$ **then**
4 $\mathbf{B}[i] \leftarrow K_{i,i+1} + (-K_{i-1,i})$
5 **else**
6 $\mathbf{B}[i] \leftarrow K_{i,i+1} \oplus K_{i-1,i}$

7 **return** $\mathbf{B}$

Once the bridge is computed, each participant signs the bridge B_i with their private-key $\mathbf{P}[i].sk_{sig}$ and broadcasts the bridge value together with its digital signature, as described in Algorithm 6.

Algorithm 6 Signing and Broadcasting Bridge (Round 2 - Step 2)

Input :

Sorted registry $\mathbf{P} = \{T_0, \dots, T_{n-1}\}$ where key tuple $T_i = (pk_{sig}, sk_{sig}, pk_{ke}, sk_{ke})$
for participant P_i and $n = |\mathbf{P}|$
Vector of bridge values $\mathbf{B} = (B_0, \dots, B_{n-1})$

Output:

Signed bridge value $(\mathbf{B}[i], \sigma_i)$, broadcast by participant P_i to all other participants

```

1  $n = |\mathbf{P}|$ 
  /* Each participant signs their respective bridge */
2 for  $i = 0$  to  $n - 1$  do
3    $\sigma_i \leftarrow \text{Sign}_{\mathbf{P}[i].sk_{sig}}(\mathbf{B}[i])$ 
  /* Broadcast the signed bridge to all participants in the network */
4   Broadcast  $(\mathbf{B}[i], \sigma_i)$ 

```

Algorithm 7 presents the pseudocode for the signature verification phase. All participants follow one of the two architectural branches to validate the received signatures. The first branch represents the naive verification model where every participant verifies every other participant's signature. The second branch is triggered under the quorum configuration where each participant verifies a sliding window of the next $\lfloor n/2 \rfloor + 1$ neighbors' signatures. While the naive model provides a higher level of security, the quorum-based architecture operates on the assumption that a majority of participants are honest. The rationale is that if a majority were dishonest, establishing a conference key would be futile from the honest minority's point of view. This reasoning renders the security overhead of exhaustive verification unnecessary.

5.5 Deriving Conference Key (Round 3)

Following the authentication phase executed in Algorithm 7, every participant, and possibly an attacker as well, possesses the complete set of valid bridges $\{B_0, B_1, \dots, B_{n-1}\}$, which are required to derive the final shared secret key SK_{final} , i.e., the conference key.

Algorithm 7 Verifying Signed Bridges (Round 2 - Step 3)**Input :**Broadcasted vector of bridge values $\mathbf{B} = (B_0, \dots, B_{n-1})$ Verification Option $V \in \{0 : \text{Naive}, 1 : \text{Quorum}\}$ Broadcasted vector of signatures $(\sigma_0, \dots, \sigma_{n-1})$ Sorted registry $\mathbf{P} = \{T_0, \dots, T_{n-1}\}$ where key tuple $T_i = (pk_{sig}, sk_{sig}, pk_{ke}, sk_{ke})$ for participant P_i and $n = |\mathbf{P}|$ **Output:****Success**, if all required bridge signatures verify for participant P_i ; or $\perp$ on failure

```

1  $n = |\mathbf{P}|$ 
2 if  $V == 0$  then
3   for  $i = 0$  to  $n - 1$  do
4     for  $j = 0$  to  $n - 1$  do
5       if  $j \neq i$  then
6         if  $\text{Verify}_{\mathbf{P}[j].pk_{sig}}(\mathbf{B}[j], \sigma_j) \neq 1$  then
7           return  $\perp$ 
8 else if  $V == 1$  then
9   Set  $\text{quorum\_size} \leftarrow \lfloor n/2 \rfloor + 1$ 
10  for  $i = 0$  to  $n - 1$  do
11    for  $j = 1$  to  $\text{quorum\_size}$  do
12      Set  $\text{target} \leftarrow (i + j) \pmod n$ 
13      if  $\text{Verify}_{\mathbf{P}[\text{target}].pk_{sig}}(\mathbf{B}[\text{target}], \sigma_{\text{target}}) \neq 1$  then
14        return  $\perp$ 
15 return Success

```

Algorithm 8 presents the pseudocode to derive SK_{final} while the underlying mathematical derivation based on the key exchange mode M_{ke} is detailed below:

1. **If $M_{ke} = 0$:** Each participant derives the conference key by performing scalar multiplication on their local left-hand point by n and adding the series of bridges multiplied by descending scalars:

$$SK = (n \cdot K_{i-1,i}) + \sum_{j=0}^{n-2} (n-1-j) \cdot B_{i+j} \quad (5.3)$$

Equation 5.3 is equivalent to Equation 5.4 due to a mathematical telescoping property; when expanding Equation 5.3, all repetitive intermediate pairwise secrets cancel out, forming a summation of all unique cyclic pairwise secrets:

$$SK = \sum_{k=0}^{n-1} K_{k,k+1} \quad (5.4)$$

The core idea for deriving a conference key is to sum all the elliptic curve points based on Equation 5.4. However, the intermediate shared secrets are not needed. Therefore, the protocol follows Equation 5.3. This implies that even though participants could derive all intermediate shared secret keys, participants can directly utilize the obtained bridges and compute a conference key using Equation 5.3.

Once the conference key SK is obtained as an elliptic curve point, the conference key SK_{final} is derived by executing a SHA3-256 key derivation function over its coordinates:

$$SK_{final} = \text{KDF}(SK_x, SK_y) \quad (5.5)$$

2. **If $M_{ke} = 1$:** Each participant derives the group key through a sequential recovery of raw pairwise secrets followed by a final hash-fusion. Starting with the locally known $K_{i-1,i}$, each member iteratively unwraps the broadcasted bridges to recover the full set of secrets:

$$K_{j,j+1} = B_j \oplus K_{j-1,j}, \quad \text{where } j = i + 1, \dots, i + n - 2 \pmod{n} \quad (5.6)$$

Unlike the key exchange mode $M_{ke} = 0$, the key encapsulation method ($M_{ke} = 1$) requires an additional step to derive all intermediate shared secrets from the bridges. Since these bridges are computed using an XOR operation, they possess an algebraic structure that allows an attacker to compromise the conference key if it would be computed as the XOR of all intermediate secrets, and the number of participants is even.

Once a participant derives all intermediate shared secret keys ($K_{0,1}, K_{1,2}, \dots, K_{n-1,0}$), a conference key SK_{final} is derived by executing a SHA3-256 key derivation function over the concatenated byte stream:

$$SK_{final} = \text{KDF}(K_{0,1} \parallel K_{1,2} \parallel \dots \parallel K_{n-1,0}) \quad (5.7)$$

Algorithm 8 Conference Key Derivation (Round 3 - Step 1)

Input :Number of participants n Key Exchange/Encapsulation Mode $M_{ke} \in \{0 : \text{KEX}, 1 : \text{KEM}\}$ Vector of bridge values $\mathbf{B} = (B_0, \dots, B_{n-1})$ Pairwise secret $K_{i-1,i}$ **Output:**Final Shared Secret Key SK_{final}

```

1 if  $M_{ke} == 0$  then
    | /* Derive using telescoping property (Eq. 5.3) */
2      $SK \leftarrow (n \cdot K_{i-1,i}) + \sum_{j=0}^{n-2} (n-1-j) \cdot B_{i+j}$ 
3      $SK_{final} \leftarrow \text{KDF}(SK)$ 
4 else if  $M_{ke} == 1$  then
    | /* Iterative unwrap to recover all pairwise secrets (Eq. 5.6) */
5      $K_{i,i+1} \leftarrow K_{i-1,i} \oplus B_i$ 
6     for  $j = i+1$  to  $i+n-2$  do
7         |  $K_{j,j+1} \leftarrow B_j \oplus K_{j-1,j} \pmod{n}$ 
            | /* Final KDF over concatenated secrets (Eq. 5.7) */
8         |  $SK_{final} \leftarrow \text{KDF}(K_{0,1} \parallel K_{1,2} \parallel \dots \parallel K_{n-1,0})$ 
9 return  $SK_{final}$ 

```

Algorithm 9 presents the pseudocode for the key confirmation phase of the MP-KEX protocol. Key confirmation is initiated by the first participant in the lexicographically sorted sequence, who hashes the conference key SK_{final} and signs the digest, subsequently broadcasting the resulting signature to the remaining participants. The protocol aborts if any participant detects a digest mismatch or signature verification failure; otherwise, the protocol succeeds.

Algorithm 9 Key Confirmation (Round 3 - Step 2)

Input :

Sorted registry $\mathbf{P} = \{T_0, \dots, T_{n-1}\}$ where key tuple $T_i = (pk_{sig}, sk_{sig}, pk_{ke}, sk_{ke})$
 for participant P_i and $n = |\mathbf{P}|$

Final Shared Secret Key SK_{final}

Output:

Success or $\perp$, returned independently by each participant P_i , $i \neq 0$

```

1  $n = |\mathbf{P}|$ 
  /*  $P_0$  computes the confirmation hash, signs it, and broadcasts */
2  $H \leftarrow \text{SHA3-256}(SK_{final})$ 
3  $\sigma \leftarrow \text{Sign}_{\mathbf{P}[0].sk_{sig}}(H)$ 
4  $P_0$  Broadcasts  $(H, \sigma)$ 
  /* Every other participant independently verifies the confirmation */
5 for  $i = 1$  to  $n - 1$  do
6   Receive  $(H', \sigma')$  from  $P_0$ 
7    $H_{local} \leftarrow \text{SHA3-256}(SK_{final})$ 
8   if  $(\text{Verify}_{\mathbf{P}[0].pk_{sig}}(H', \sigma') \neq 1) \vee (H_{local} \neq H')$  then
9     | return  $\perp$ 
10  else
11  | return Success
```

6 Performance Analysis

This chapter is organized into two sections. Section 6.1 describes the computing platform used for the experiments, while Section 6.2 presents the empirical results obtained from implementing the protocol described in Chapter 5.

6.1 Experimental Setup

The experiments were run on an Apple MacBook Air (M2, 2022). The hardware configuration and software environment used for these evaluations are detailed in Table 6.1.

Table 6.1: Experimental Computing Platform Specifications

Component	Specification
Hardware	Apple MacBook Air (M2, 2022)
Processor	Apple M2
Memory	16 GB
Operating System	macOS 26.5.1 (25F80)
Compiler	Apple clang version 21.0.0 (clang-2100.1.1.101)
Cryptographic Library	OpenSSL 3.6.0

The implementation employs OpenSSL [61] and the Library for Open Quantum Safe (liboqs) [60] as its cryptographic foundations for providing a reproducible and modular testing environment. OpenSSL supplies well-established implementations of classical cryptographic primitives, serving as a performance baseline for conventional security operations. The library liboqs complements this by providing open-source implementations of the NIST-standardised PQ algorithms such as ML-KEM (FIPS 203) and ML-DSA (FIPS 204), enabling evaluation of the protocol under post-quantum threat assumptions. Both libraries are widely adopted in cryptographic research and derive their algorithm implementations from submissions to the NIST PQC Standardization process.

For the implementation purpose, algorithms described in Chapter 5 were programmed in C language. The versions of the compiler and cryptographic library are mentioned in

Table 6.1. The size of the source code is from 5 KB to 16 KB, depending on the chosen algorithms, while the size of the executable code is from 248 KB to 4 MB.

All pre-processing actions related to group set-up, defined in Section 5.1 to happen before Round 0, were omitted. As this pre-processing typically occurs during the handshake phase of real-world deployments, and our focus is specifically on the computational performance of the MP-KEX algorithm combinations, this omission is justified. All other phases described in Section 5.2 through Section 5.5 were executed, and their results (excluding Round 0) are reported in Section 6.2.2.

The group size is ranged from 5 to 100 participants because this seems to be sufficient to capture the scaling trends of the protocol. To account for CPU noise and ensure meaningful results, each measurement reported in this chapter represents the average of ten experimental runs. While this approach helps quantify performance, it also reflects real-world conditions, as the external factors influencing CPU cycles are often present in the deployed systems and environments where such protocols operate.

6.2 MP-KEX Evaluation

This section is organized into three subsections. Section 6.2.1 explains the parameter settings of the algorithms. Section 6.2.2 presents the empirical findings, while Section 6.2.3 addresses the research questions based on the outcomes.

6.2.1 Experiments

The MP-KEX protocol was evaluated across the following four cryptographic combinations:

- ECDH+ECDSA : X25519 for key establishment and ECDSA-P256 for authentication.
- ECDH+ML-DSA : X25519 for key establishment and ML-DSA (parameters: 44, 65, 78) for authentication.
- ML-KEM+ECDSA: ML-KEM (parameters: 512, 768, 1024) for key establishment and ECDSA-P256 for authentication.

- ML-KEM+ML-DSA : ML-KEM (parameters: 512, 768, 1024) for key establishment and ML-DSA (parameters: 44, 65, 78) for authentication.

The rationale behind choosing X25519 is that it is relatively fast and offers a 128-bit security level. ECDSA-P256 also offers 128-bit security and is a widely adopted authentication algorithm. For simplicity, we used the lowest NIST security level for ML-KEM and ML-DSA when plotting graphs. This gives baseline performance figures for these PQC algorithms. The results for the other NIST security levels are also reported.

With these classical and PQ algorithms as building blocks, we designed experiments whose primary objective is to assess the performance of each algorithm combination and determine its viability for hybrid or fully PQC migrations.

6.2.2 Results

Figure 6.1 illustrates the performance scalability of the MP-KEX protocol by plotting the total latency in microseconds against an increasing number of participants, ranging from 5 to 100. This figure is plotted to observe the baseline performance of the MP-KEX protocol. The figure provides a comparative view of four distinct combinations of cryptographic algorithms, showcasing how the total execution time changes as the group size increases. The consistent, upward trend across all four lines confirms that the protocol exhibits linear complexity, providing a clear visual assessment of which algorithm combinations offer the most favorable scaling characteristics as the number of participants increases.

In Figure 6.1, the green line (X25519 and ECDSA-P256) and the blue line (ML-KEM 512 and ECDSA-P256) overlap closely. Although the ML-KEM based protocol includes more steps in the Round 1 phase, where the public key pk_{ke} and ciphertext C are signed and verified, this does not significantly impact performance when ECDSA-P256 is used. In contrast, there is a noticeable difference between the red (X25519 and ML-DSA-44) and orange (ML-KEM-512 and ML-DSA-44) lines. While the protocol structure remains consistent across these comparisons, the time difference arises due to the computationally expensive ML-DSA-44 signature algorithm in signing and verifying the ciphertext C leading to the intermediate shared key in Round 1. Thus, the shift from the blue and green lines to the red line is likely caused by the change to ML-DSA-44 from ECDSA-P256, and the consistent vertical gap between the red and orange lines is attributed to the additional signing and verification steps for the ciphertext of shared key C when using ML-KEM-512 in Round 1.

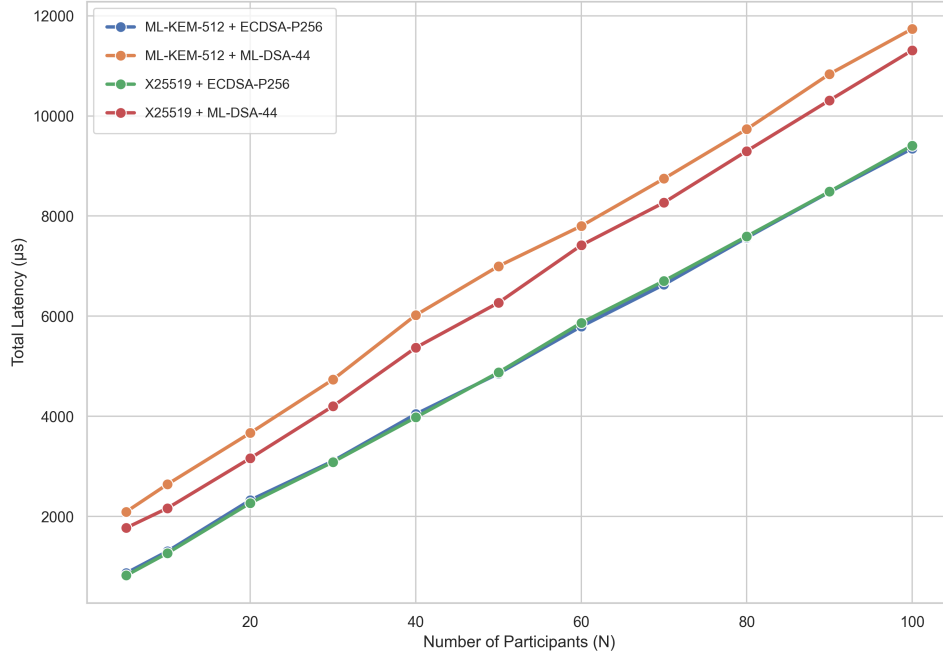


Figure 6.1: Total Latency Comparison across different combinations of algorithms

Figure 6.2 provides a computational breakdown per participant for both small-scale (5 participants) and large-scale (100 participants) MP-KEX protocol deployments. Each stacked bar represents the total execution time of the MP-KEX protocol, segmented into three distinct rounds: “Setup and Auth” (Round 1), “Bridge and Verify All” (Round 2), and “Finalization” (Round 3). In the upper chart of Figure 6.2, i.e., when the number of participants is 5, as regards Round 1, the bars using ML-DSA-44 (second and fourth bar graphs from the left) appear larger than the bars using ECDSA-P256 (first and third bar graphs from left). Within the ML-DSA-44 bars, moving from X25519 (second graph) to ML-KEM-512 (fourth graph) adds two extra steps: signing and verifying the ciphertext C . This accounts for the increase in computation time from roughly 675 to 1150, which is about 70 % increase. On the other hand, with ECDSA-P256 used for authentication, there is roughly 36 % increase in time when moving from X25519 to ML-KEM-512.

Again when looking in the upper chart of Figure 6.2, as regards Round 3, first and third bars are almost the same, while second and fourth bars appear to be of roughly the same size. This ratio of the first and third bars to the second and fourth bars is explained by the type of authentication algorithm used. To confirm that all participants have derived the same final shared secret key, the first participant in lexicographical order hashes and signs this key. Since key derivation remains the same for all participants, only the type of

authentication algorithm used for the key confirmation accounts for this small difference in computation time between ECDSA-P256 (first and third bars) and ML-DSA-44 (second and fourth bars).

In the lower chart of Figure 6.2, i.e., when the number of participants is 100, the runtime for Round 1 does not change at all compared to the upper chart of Figure 6.2. This is because all the operations performed during Round 1 are independent of group size. The computational time for Round 3 is relatively shorter, although the group size slightly influences the key derivation. As regards Round 2, the computational time becomes extremely large, which highlights the primary performance bottleneck of the MP-KEX protocol. The “Bridge and Verify All” phase becomes the dominant factor in total latency because each participant must verify all of the other $n - 1$ bridges, turning a manageable workload for small groups into a significant computational burden as the group size increases.

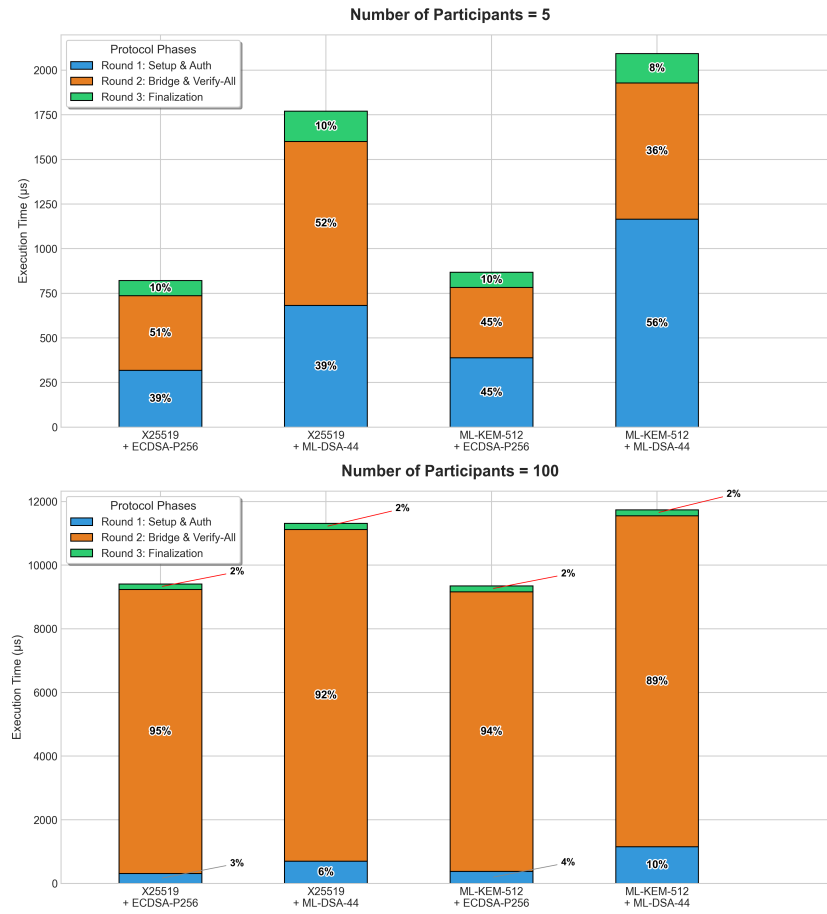


Figure 6.2: Computational Load Breakdown per Participant (Naive Verification Model in Round 2)

Figure 6.3 presents the total time taken for verifying the bridges in Round 2. The execution time scales linearly with the number of participants and closely mirrors the total latency trends observed in Figure 6.1. The sharp, consistent increase in microseconds across all four cryptographic algorithm combinations underscores that the computational demands of verifying participant data dominate the protocol’s workload. This forms a critical performance constraint that limits efficiency as the group size increases. Because bridge verification depends on the authentication algorithm used, the runtime difference between ECDSA-P256 and ML-DSA-44 grows to as large as 1100 μs when the group size is 100.

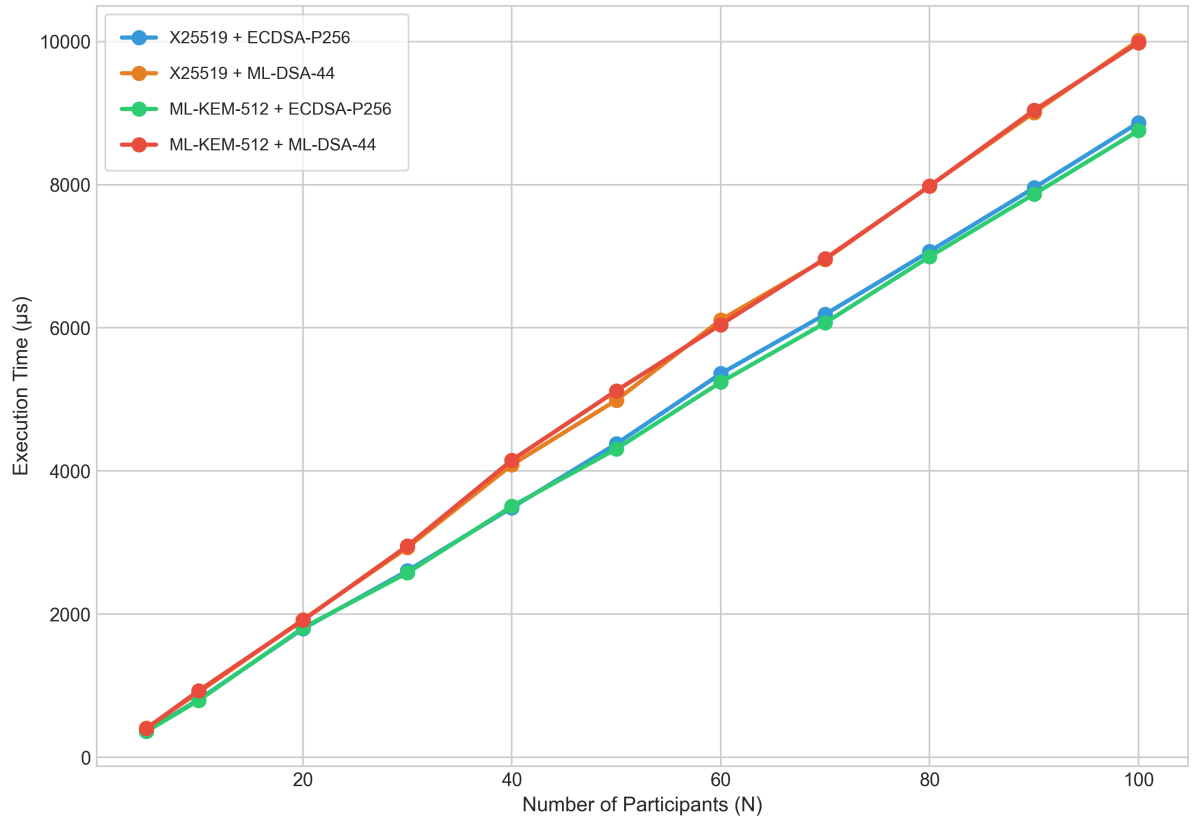


Figure 6.3: Verifying $n - 1$ bridges in Round 2 (Naive Verification Model)

Figure 6.4 compares the bridge verification time of the naive and quorum-based models across four algorithm combinations when the number of participants (N) is 10, 50, 70 or 100. The quorum-based model consistently reduces verification time across all group sizes, with the proportional decrease growing from approximately reduction by one third at $N=10$ to nearly reduction by half at $N=100$. This trend indicates that the quorum-based model becomes increasingly effective as group size grows, stabilizing at a nearly 50 % reduction in verification time for larger number of participants. This is because as the

group size increases, each participant verifies $\lfloor n/2 \rfloor + 1$ bridges, instead of $n - 1$ bridges.

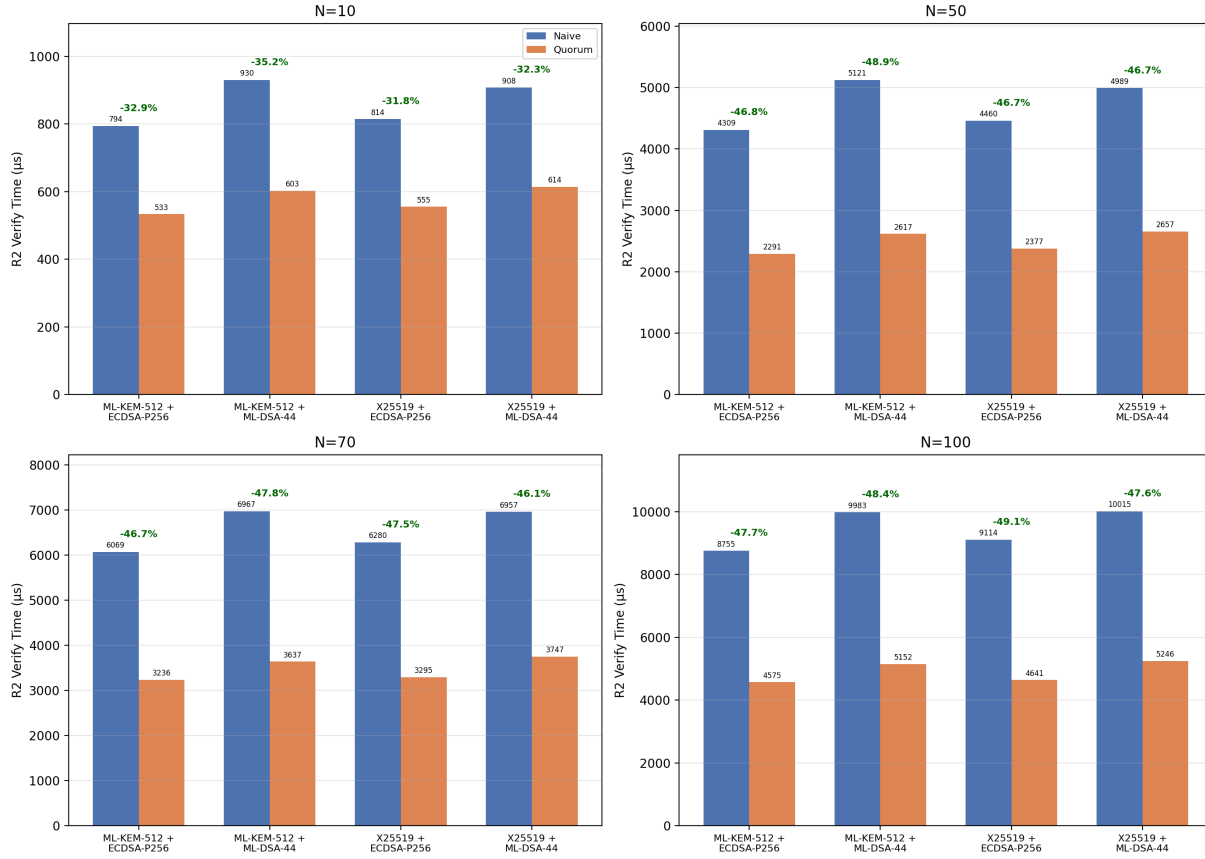


Figure 6.4: Comparison of Round 2 Verification with Naive and Quorum-Based

Table 6.2 presents the stability of the computation time per operation per participant as group size scales. The majority of operations remain computationally stable regardless of group size, since each participant only interacts with a fixed number of neighbors or performs operations on their own data. Notably, bridge verification (operation 6) and final secret derivation (operation 7) are the two exceptions, as both depend on a variable number of components that grows with group size. Consequently, total latency (listed as operation 9) is primarily influenced by bridge verification (operation 6), with final secret derivation (operation 7) contributing to a lesser extent, as the number of participants increases.

Table 6.3 presents the average computational time per operation per participant. The blue highlighted cells show that bridge computation time remains nearly constant across all ML-KEM parameter sets (512, 768, and 1024), approximately 0.21–0.22 μ s. This is because the operation relies solely on bitwise XOR over fixed-length bit strings, independent of the KEM parameter size. The green highlighted cells indicate that signing, verification,

Table 6.2: Execution Time Stability Relative to Group Size

No.	Operation	Stable?	Reason
1	Signing Public Key (+ Ciphertext for KEM)	✓	Signs own public key (also ciphertext for KEM)
2	Verifying Public Key (+ Ciphertext for KEM)	✓	Verifies two neighbors' keys (also ciphertext for KEM)
3	Key Establishment	✓	Performs with two neighbors
4	Bridge Computation	✓	Uses two intermediate keys
5	Signing Bridge	✓	Signs own bridge
6	Verifying Bridge	✗	Verifies variable number of bridges
7	Final Secret Derivation	✗	Requires variable number of components
8	Confirming Final Secret	✓	One-to-one confirmation
9	Total Latency	✗	Resulting from operations 6 and 7

and key establishment times increase monotonically with the security parameter. This monotonic increase reflects the higher computational cost of larger parameter settings in both ML-KEM and ML-DSA variants.

Table 6.3: Average Time Taken (μ s) to Execute Various Operations. ‘-’ sign indicates that value is not available since operation does not rely on corresponding algorithm

Operation	X25519	ECDSA-P256	ML-KEM-512	ML-KEM-768	ML-KEM-1024	ML-DSA-44	ML-DSA-65	ML-DSA-87
Signing Public Key (+ Ciphertext for KEM)	-	35.09 (66.67)	-	-	-	403.46 (835.75)	685.18 (1392.29)	846.83 (1663.42)
Verifying Public Key (+ Ciphertext for KEM)	-	181.69 (274.06)	-	-	-	204.08 (317.82)	335.29 (517.37)	562.73 (838.69)
Key Establishment	98.19	-	44.27	65.07	98.18	-	-	-
Bridge Computation	17.29	-	0.21	0.22	0.22	-	-	-
Signing Bridge	-	33.02	-	-	-	411.17	658.30	839.95

Table 6.4 presents the security and design properties achieved by the MP-KEX protocol. MP-KEX achieves basic secrecy and perfect forward secrecy through the use of fresh ephemeral key pairs generated in Round 0, alongside the hybrid approach and crypto-agility through its modular algorithm selection. However, the unachieved properties, marked with red crosses, are not fundamental weaknesses of the protocol design itself, but rather stem from the absence of dynamic group management and key rotation mechanisms. These unachieved properties are identified as directions for future work.

6.2.3 Addressing Research Questions

RQ1 “To what extent does the MP-KEX protocol satisfy the security requirements of a post-quantum contributory group key agreement protocol?” is addressed through the security analysis presented in Table 6.4, evaluating the protocol against established security properties. As described in Section 6.2.2, the MP-KEX protocol achieves the fundamental security properties. Table 6.4 suggests that incorporating dynamic group management

Table 6.4: Properties Achieved by MP-KEX Protocol

Property	Achieved?	Reason
Basic Secrecy	✓	Conference key is not revealed to outside adversaries
Perfect Forward Secrecy	✓	Fresh ephemeral key pairs are generated in Round 0
Forward Secrecy (Leave)	✗	Requires rekeying upon participant departure from the group
Backward Secrecy (Join)	✗	Requires rekeying upon participant arrival to the group
Post-Compromise Secrecy	✗	Requires Key Rotation and Dynamic Group Management
Hybrid Approach	✓	Demonstrated with four algorithm combinations
Crypto-Agility	✓	Algorithm selection is modular

and key rotation is required to achieve other security properties.

Regarding RQ2 “How do classical and post-quantum algorithm combinations compare in terms of computational efficiency within the MP-KEX protocol?”, ML-KEM demonstrates significantly lower key establishment latency compared to X25519, attributed to its reliance on structured lattice operations rather than elliptic curve scalar multiplication. Table 6.3 suggests that bridge computation with the ML-KEM variant is significantly faster than that of X25519. In contrast, ML-DSA signing is considerably more computationally expensive than ECDSA-P256. For verification, ML-DSA-44 takes slightly more runtime than ECDSA-P256. However, with larger parameter settings, ECDSA-P256 verification becomes comparatively faster. Consequently, the hybrid combination of ML-KEM for key exchange and ECDSA-P256 for authentication provides the optimal near-term performance balance.

Regarding RQ3 “How does the MP-KEX protocol scale in terms of computational complexity as the number of participants increases?”, as shown in Table 6.2, the majority of operations remain computationally consistent regardless of group size. However, bridge verification scales linearly with the number of participants, forming the primary performance bottleneck of the protocol. This bottleneck is partially addressed by introducing a quorum-based verification model, which reduces the number of required verifications per participant from $n - 1$ to $\lfloor n/2 \rfloor + 1$.

7 Future Directions

Although the experiment was conducted in a controlled environment, i.e., in a standard computer rather than a decentralized network, it yielded promising results, highlighting potential research directions for establishing conference keys using PQC algorithms. The potential future research directions are listed as follows:

- **Key Confirmation:** Key confirmation is currently performed in a one-to-many manner, where the lexicographically first-ordered participant hashes and signs the conference key while the remaining participants verify the signature. It would be worthwhile to explore alternative confirmation ways that distribute this workload more evenly, avoiding the single point of failure inherent in this design.
- **Formal Verification:** Although MP-KEX demonstrates promising results, its security properties have only been informally assessed in this thesis. Formal verification using tools such as Tamarin [72] and ProVerif [63] could provide security proofs under various threat models and authentication levels under Lowe’s hierarchy [35].
- **Dynamic Group Key Management:** Frequent key rotation strengthens secrecy by limiting how much an attacker can compromise from a single session key. Group membership changes pose a related challenge: a) when a participant joins, the group must derive a new key that excludes knowledge of prior secrets, ensuring backward secrecy, and b) when a participant leaves, forward secrecy is harder to guarantee, since the departing member retains intermediate secrets that underpin MP-KEX’s security model. A straight-forward solution for both leaving and joining is to begin from scratch every time the group composition changes but this is not a practical solution if such changes are frequent.
- **Algorithms and Parameters:** As MP-KEX is algorithm-agnostic, its inherent crypto-agility enables future work to evaluate the performance of additional PQC algorithms and parameter sets for key exchange and authentication other than those used in this thesis.
- **Real-world Deployment:** If all future works are successful, soft-launching the MP-KEX protocol in a controlled environment enables stakeholders to evaluate its practical viability and identify interoperability challenges absent from a single-machine

simulation. Such deployment trials could inform whether the MP-KEX protocol is suitable for practical settings such as IoT networks and telecommunications infrastructures.

8 Conclusions

This thesis presents the design, implementation, and evaluation of the MP-KEX protocol that addresses a research gap in post-quantum group key agreement. Due to the impending emergence of CRQCs, Shor’s algorithm compromises existing cryptographic primitives based on discrete logarithm and integer factorization problems, motivating the migration toward key encapsulation mechanisms built on novel mathematical hardness assumptions that remain robust against quantum attacks.

Beginning with the quantum threat landscape in Chapter 2, this thesis covered the urgency of post-quantum migration through Mosca’s Inequality and regulatory actions by NIST, NSA, and EU. Chapter 3 surveyed previous work related to multi-party key exchange and identified the corresponding research gap. Chapter 4 introduced the cryptographic foundations underpinning the MP-KEX protocol, including classical primitives, NIST-standardized PQC algorithms, hybrid approaches, and cryptographic agility. Chapter 5 presented the full design of the MP-KEX protocol across four execution rounds, from key generation and topology alignment through bridge computation and to final conference key derivation.

Chapter 6 evaluated the MP-KEX protocol across four cryptographic algorithm combinations. The results show a monotonic increase in computational time per participant as the group size increases. Notably, digital signature algorithm dictate much of the computational time of the MP-KEX protocol per participant, arising from the naive bridge verification model. This was addressed with a quorum-based verification model, which reduced this verification time to nearly half from that of the naive model as for the larger group size, say from number of participants is 50. ML-KEM showed comparable computation time to elliptic-curve-based operations for key establishment with two neighbors in Round 1, but significantly faster bridge computation in Round 2.

Although the MP-KEX protocol demonstrates promising results for post-quantum group key establishment, some core secrecy properties remain unachieved according to the informal assessment in Table 6.4, necessitating the inclusion of additional features such as key rotation and dynamic group key management in future iterations of the design. Chapter 7 outlines these future research directions, including the formal security assessment of the MP-KEX protocol and the tuning of algorithms and their parameters to suit varying

deployment contexts.

All in all, this thesis demonstrated an algorithm-agnostic post-quantum multi-party key exchange protocol for establishing a conference key. The empirical results enabled the identification of key performance patterns and bottleneck across algorithm combinations and group sizes, marking a crucial step toward practically feasible and quantum-resistant group communication as the post-quantum migration effort continues.

References

- [1] 3GPP. *Specification of the 3GPP Confidentiality and Integrity Algorithms UEA2 & UIA2; Document 2: SNOW 3G specification*. Tech. rep. TS 35.215. 3rd Generation Partnership Project, 2006. URL: <https://www.gsma.com/about-us/wp-content/uploads/2014/12/snow3gspec.pdf>.
- [2] A. Adamantiadis, S. Josefsson, and M. D. Baushke. *Secure Shell (SSH) Key Exchange Method Using Curve25519 and Curve448*. RFC 8731. Feb. 2020. DOI: [10.17487/RFC8731](https://doi.org/10.17487/RFC8731).
- [3] G. Alagic, E. Barker, L. Chen, D. Moody, A. Robinson, H. Silberg, and N. Waller. *Recommendations for Key-Encapsulation Mechanisms*. NIST Special Publication (SP) 800-227. Gaithersburg, MD: National Institute of Standards and Technology (NIST), Sept. 2025. DOI: [10.6028/NIST.SP.800-227](https://doi.org/10.6028/NIST.SP.800-227).
- [4] J. Alwen, M. Campagna, D. Hartmann, S. Katsumata, E. Kiltz, J. Massimo, M. Mu-larczyk, G. Pascual-Perez, T. Prest, and P. Schwabe. “How Multi-Recipient KEMs can help the Deployment of Post-Quantum Cryptography”. In: *Proceedings of the 5th NIST PQC Standardization Conference*. 2024. URL: <https://csrc.nist.gov/csrc/media/Events/2024/fifth-pqc-standardization-conference/documents/papers/how-multi-recipient-kems.pdf>.
- [5] D. Apon, D. Dachman-Soled, H. Gong, and J. Katz. “Constant-Round Group Key Exchange from the Ring-LWE Assumption”. In: *International Conference on Post-Quantum Cryptography*. Springer. 2019, pp. 189–205. DOI: https://doi.org/10.1007/978-3-030-25510-7_11.
- [6] R. Babbush, A. Zalcman, C. Gidney, M. Broughton, T. Khattar, H. Neven, T. Bergamaschi, J. Drake, and D. Boneh. *Securing Elliptic Curve Cryptocurrencies against Quantum Vulnerabilities: Resource Estimates and Mitigations*. 2026. arXiv: [2603.28846](https://arxiv.org/abs/2603.28846) [quant-ph].
- [7] S. Bantanidis, K. Master, R. Shah, P. Sharma, and R. Ghose. *Quantum Threat: The Trillion-Dollar Security Race Is On*. Tech. rep. Accessed: 2026-01-24. Citi Institute, Jan. 2026. URL: https://www.citigroup.com/rcs/citigpa/storage/public/Citi_Institute_Quantum_Threat.pdf.

- [8] E. Barker. *Recommendation for Key Management: Part 1 – General*. Tech. rep. NIST Special Publication 800-57 Part 1 Revision 5. National Institute of Standards and Technology (NIST), May 2020. DOI: [10.6028/NIST.SP.800-57pt1r5](https://doi.org/10.6028/NIST.SP.800-57pt1r5).
- [9] E. Barker, L. Chen, D. Cooper, D. Moody, A. Regenscheid, M. Souppaya, W. Newhouse, R. Housley, S. Turner, W. Barker, and K. Kent. “Considerations for Achieving Crypto Agility: Strategies and Practices”. In: (2025). DOI: [10.6028/NIST.CSWP.39.ipd](https://doi.org/10.6028/NIST.CSWP.39.ipd).
- [10] D. J. Bernstein. “Curve25519: New Diffie-Hellman Speed Records”. In: *Public Key Cryptography - PKC 2006*. Ed. by M. Yung, Y. Dodis, A. Kiayias, and T. Malkin. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 207–228. ISBN: 978-3-540-33852-9. DOI: https://doi.org/10.1007/11745853_14.
- [11] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. *The Keccak SHA-3 submission*. Submission to the NIST SHA-3 Cryptographic Hash Algorithm Competition Version 3. Keccak Team, 2011. URL: <https://keccak.team/files/Keccak-submission-3.pdf>.
- [12] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: [10.17487/RFC5280](https://doi.org/10.17487/RFC5280).
- [13] D. Boneh and M. Zhandry. “Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation”. In: *Algorithmica* 79.4 (2017), pp. 1233–1285. DOI: <https://doi.org/10.1007/s00453-016-0242-8>.
- [14] M. Burmester and Y. G. Desmedt. “Efficient and secure conference-key distribution”. In: *Security Protocols*. Ed. by M. Lomas. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 119–129. ISBN: 978-3-540-68047-5. DOI: https://doi.org/10.1007/3-540-62494-5_12.
- [15] L. Chen, S. Jordan, Y.-K. Liu, D. Moody, R. Peralta, R. Perlner, and D. Smith-Tone. *Report on Post-Quantum Cryptography*. Tech. rep. NIST IR 8105. National Institute of Standards and Technology, 2016. DOI: [10.6028/nist.ir.8105](https://doi.org/10.6028/nist.ir.8105).
- [16] L. Chen, D. Moody, A. Regenscheid, A. Robinson, and K. Randall. *Recommendations for Discrete Logarithm-based Cryptography: Elliptic Curve Domain Parameters*. Tech. rep. NIST Special Publication (SP) 800-186. National Institute of Standards and Technology, 2023. DOI: [10.6028/nist.sp.800-186](https://doi.org/10.6028/nist.sp.800-186).

- [17] CISA, NSA, and NIST. *Quantum-Readiness: Migration to Post-Quantum Cryptography*. Tech. rep. 2023. URL: <https://www.cisa.gov/resources-tools/resources/quantum-readiness-migration-post-quantum-cryptography>.
- [18] H. Delfs and H. Knebl. *Introduction to Cryptography: Principles and Applications*. 3rd ed. Information Security and Cryptography. Springer Berlin, Heidelberg, 2015. DOI: <https://doi.org/10.1007/978-3-662-47974-2>.
- [19] W. Diffie and M. Hellman. “New directions in cryptography”. In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: [10.1109/TIT.1976.1055638](https://doi.org/10.1109/TIT.1976.1055638).
- [20] R. Doring and M. Geitz. “Redundant multipath PQC key exchange using Shamir’s Secret Sharing”. In: *2024 13th International Conference on Communications, Circuits and Systems (ICCCAS)*. 2024, pp. 474–479. DOI: [10.1109/ICCCAS62034.2024.10652767](https://doi.org/10.1109/ICCCAS62034.2024.10652767).
- [21] J. I. Escribano Pablos and M. I. González Vasco. “Secure post-quantum group key exchange: Implementing a solution based on Kyber”. In: *IET Communications* 17.6 (2023), pp. 758–773. DOI: <https://doi.org/10.1049/cmu2.12561>.
- [22] European Commission. *A Coordinated Implementation Roadmap for the Transition To Post-Quantum Cryptography*. Accessed on 3 June 2026. 2024. URL: <https://digital-strategy.ec.europa.eu/en/library/coordinated-implementation-roadmap-transition-post-quantum-cryptography>.
- [23] D. Ghinea, F. Kaczmarczyk, J. Pullman, J. Cretin, S. Kölbl, R. Misoczki, J.-M. Picod, L. Invernizzi, and E. Bursztein. “Hybrid Post-quantum Signatures in Hardware Security Keys”. In: *Applied Cryptography and Network Security Workshops*. Cham: Springer Nature Switzerland, 2023, pp. 480–499. ISBN: 978-3-031-41181-6. DOI: https://doi.org/10.1007/978-3-031-41181-6_26.
- [24] C. Gidney. *How to factor 2048 bit RSA integers with less than a million noisy qubits*. 2025. arXiv: [2505.15917 \[quant-ph\]](https://arxiv.org/abs/2505.15917).
- [25] L. K. Grover. “A fast quantum mechanical algorithm for database search”. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC ’96. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996, pp. 212–219. ISBN: 0897917855. DOI: [10.1145/237814.237866](https://doi.org/10.1145/237814.237866).
- [26] R. Housley. *Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms*. IETF RFC: 7696. Nov. 2015. DOI: [10.17487/RFC7696](https://doi.org/10.17487/RFC7696).

- [27] M. Ivezic. *Trust Now, Forge Later (TNFL) – The Overlooked Quantum Threat*. <https://postquantum.com/post-quantum/trust-now-forge-later/>. Blog post. Accessed: June 4, 2026.
- [28] F. Kagai, P. Branch, J. But, and R. Allen. “Harvest-Now, Decrypt-Later: A Temporal Cybersecurity Risk in the Quantum Transition”. In: *Telecom* 6.4 (2025). ISSN: 2673-4001. DOI: [10.3390/telecom6040100](https://doi.org/10.3390/telecom6040100).
- [29] J. Katz and M. Yung. “Scalable Protocols for Authenticated Group Key Exchange”. In: *Journal of Cryptology* 20.1 (2007), pp. 85–113. DOI: https://doi.org/10.1007/978-3-540-45146-4_7.
- [30] E. Kiltz, V. Lyubashevsky, and C. Schaffner. “A Concrete Treatment of Fiat-Shamir Signatures in the Quantum Random-Oracle Model”. In: *Advances in Cryptology – EUROCRYPT 2018*. Ed. by J. B. Nielsen and V. Rijmen. Cham: Springer International Publishing, 2018, pp. 552–586. ISBN: 978-3-319-78372-7. DOI: https://doi.org/10.1007/978-3-319-78372-7_18.
- [31] N. Koblitz. “Elliptic Curve Cryptosystems”. In: *Mathematics of Computation* 48.177 (1987), pp. 203–209.
- [32] K. Kwiatkowski, P. Kampanakis, B. Westerbaan, and D. Stebila. *Post-quantum hybrid ECDHE-MLKEM Key Agreement for TLSv1.3*. Internet-Draft draft-ietf-tls-ecdhe-mlkem-05. Work in Progress. Internet Engineering Task Force, May 2026. 12 pp. URL: <https://datatracker.ietf.org/doc/draft-ietf-tls-ecdhe-mlkem/05/>.
- [33] A. Langley, M. Hamburg, and S. Turner. *Elliptic Curves for Security*. RFC 7748. Jan. 2016. DOI: [10.17487/RFC7748](https://doi.org/10.17487/RFC7748).
- [34] A. Langlois and D. Stehlé. “Worst-case to average-case reductions for module lattices”. In: *Designs, Codes and Cryptography* 75.3 (2015), pp. 565–599. DOI: [10.1007/s10623-014-9938-4](https://doi.org/10.1007/s10623-014-9938-4).
- [35] G. Lowe. “A Hierarchy of Authentication Specifications”. In: *Proceedings 10th computer security foundations workshop*. IEEE. 1997, pp. 31–43. DOI: [10.1109/CSFW.1997.596782](https://doi.org/10.1109/CSFW.1997.596782).
- [36] V. Lyubashevsky. “Fiat-Shamir With Aborts: Applications to Lattice and Discrete-Log Based Signatures”. In: *Advances in Cryptology – EUROCRYPT 2009*. Ed. by C. Antoine. Vol. 5479. Lecture Notes in Computer Science. Springer, 2009, pp. 598–616. DOI: [10.1007/978-3-642-10366-7_35](https://doi.org/10.1007/978-3-642-10366-7_35).

- [37] V. Lyubashevsky. “Lattice Signatures without Trapdoors”. In: *Advances in Cryptology – EUROCRYPT 2012*. Ed. by D. Pointcheval and T. Johansson. Vol. 7237. Lecture Notes in Computer Science. Springer, 2012, pp. 738–755. DOI: [10.1007/978-3-642-29011-4_43](https://doi.org/10.1007/978-3-642-29011-4_43).
- [38] V. Lyubashevsky, L. Ducas, E. Kiltz, T. Lepoint, P. Schwabe, G. Seiler, D. Stehlé, R. Avanzi, J. Bos, and J. M. Schanck. *CRYSTALS-Dilithium: A Post-Quantum Digital Signature Scheme*. <https://pq-crystals.org/dilithium/>. Last Accessed: 2026-02-20. 2024.
- [39] J. P. Mattsson, E. Thormarker, M. Asadi, and S. Ruohomaa. *Migrating telecom to quantum-resistant cryptography on a global scale*. <https://www.ericsson.com/en/reports-and-papers/ericsson-technology-review/articles/migrating-telecom-to-quantum-resistant-cryptography>. Accessed: June 17, 2026. May 2026.
- [40] M. Mosca. *Cybersecurity in a Quantum World: will we be ready?* Presentation slides, NIST Workshop on Cybersecurity in a Post-Quantum World. Accessed: 2026-06-19. Apr. 2015. URL: <https://csrc.nist.gov/csrc/media/events/workshop-on-cybersecurity-in-a-post-quantum-world/documents/presentations/session8-mosca-michele.pdf>.
- [41] M. Mosca. “Cybersecurity in an Era with Quantum Computers: Will We Be Ready?” In: *IEEE Security & Privacy* 16.5 (2018), pp. 38–41. DOI: [10.1109/MSP.2018.3761723](https://doi.org/10.1109/MSP.2018.3761723).
- [42] M. Mosca and M. Piani. *Quantum Threat Timeline Report 2025*. Tech. rep. Co-authored by evolutionQ Inc. Global Risk Institute, Mar. 2025. URL: <https://globalriskinstitute.org/publication/quantum-threat-timeline-report-2025b/>.
- [43] National Institute of Standards and Technology. *Advanced Encryption Standard (AES)*. Tech. rep. Federal Information Processing Standards (FIPS) 197. National Institute of Standards and Technology, 2001. URL: <https://doi.org/10.6028/NIST.FIPS.197-upd1>.
- [44] National Institute of Standards and Technology. *Announcing PQC Candidates to be Standardized, Plus Fourth Round Candidates*. <https://csrc.nist.gov/news/2022/pqc-candidates-to-be-standardized-and-round-4>. Last Accessed: 2026-02-20. July 2022.

- [45] National Institute of Standards and Technology. *Digital Signature Standard (DSS)*. Federal Information Processing Standards Publication (FIPS PUB) 186-5. Gaithersburg, MD: U.S. Department of Commerce, Feb. 2023. URL: <https://csrc.nist.gov/pubs/fips/186-5/final>.
- [46] National Institute of Standards and Technology. *FIPS 206 Status Update*. Tech. rep. Presentation slide. National Institute of Standards and Technology, 2025. URL: [https://csrc.nist.gov/csrc/media/presentations/2025/fips-206-fn-dsa-\(falcon\)/images-media/fips_206-perlner_2.1.pdf](https://csrc.nist.gov/csrc/media/presentations/2025/fips-206-fn-dsa-(falcon)/images-media/fips_206-perlner_2.1.pdf).
- [47] National Institute of Standards and Technology. *FIPS 207: HQC KEM*. Tech. rep. Presentation slide. National Institute of Standards and Technology, 2025. URL: https://csrc.nist.gov/csrc/media/presentations/2025/fips-207-hqc-kem/fips_207-robinson_2.4.pdf.
- [48] National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard*. Tech. rep. Federal Information Processing Standards (FIPS) 204. Gaithersburg, MD: U.S. Department of Commerce, Aug. 2024. DOI: [10.6028/NIST.FIPS.204](https://doi.org/10.6028/NIST.FIPS.204).
- [49] National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*. Tech. rep. Federal Information Processing Standards (FIPS) 203. National Institute of Standards and Technology, 2024. DOI: [10.6028/NIST.FIPS.203](https://doi.org/10.6028/NIST.FIPS.203).
- [50] National Institute of Standards and Technology. *Post-Quantum Cryptography Project*. <https://csrc.nist.gov/projects/post-quantum-cryptography>. Accessed: 2026-06-03.
- [51] National Institute of Standards and Technology. *Post-Quantum Cryptography Security (Evaluation Criteria)*. [https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-\(evaluation-criteria\)](https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria)). Accessed: 2026-06-20.
- [52] National Institute of Standards and Technology. *Post-Quantum Cryptography: Additional Digital Signature Schemes (Round 3)*. <https://csrc.nist.gov/Projects/pqc-dig-sig/round-3-additional-signatures>. Accessed: 2026-06-21.
- [53] National Institute of Standards and Technology. *Post-Quantum Cryptography: Selected Algorithms*. Accessed: 2026-06-22. URL: <https://csrc.nist.gov/Projects/>

[post-quantum-cryptography/post-quantum-cryptography-standardization/selected-algorithms](#).

- [54] National Institute of Standards and Technology. *Secure Hash Standard (SHS)*. Federal Information Processing Standards Publication FIPS PUB 180-4. U.S. Department of Commerce, 2015. DOI: [10.6028/NIST.FIPS.180-4](#).
- [55] National Institute of Standards and Technology. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*. Federal Information Processing Standards Publication (FIPS PUB) 202. National Institute of Standards and Technology, Aug. 2015. DOI: [10.6028/NIST.FIPS.202](#).
- [56] National Institute of Standards and Technology. *Stateless Hash-Based Digital Signature Standard*. Tech. rep. Federal Information Processing Standards (FIPS) 205. National Institute of Standards and Technology, 2024. DOI: [10.6028/NIST.FIPS.205](#).
- [57] National Institute of Standards and Technology. *Workshop on Cybersecurity in a Post-Quantum World*. <https://csrc.nist.gov/events/2015/workshop-on-cybersecurity-in-a-post-quantum-world>. Accessed: 2026-06-03.
- [58] National Security Agency. *Announcing Commercial National Security Algorithm Suite 2.0*. U.S. NSA: Fort Meade, MD, USA. Accessed on 5 October 2026. 2022. URL: https://media.defense.gov/2025/May/30/2003728741/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS.PDF.
- [59] National Security Agency. *The Commercial National Security Algorithm Suite 2.0 and Quantum Computing FAQ*. Tech. rep. Version 2.1. Dec. 2024. URL: https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ.PDF.
- [60] Open Quantum Safe. *liboqs*. <https://openquantumsafe.org/liboqs/>. Accessed: June 13, 2026.
- [61] OpenSSL Project. *OpenSSL*. <https://www.openssl.org/>. Accessed: June 13, 2026.
- [62] T. Perrin. *The XEdDSA and VXEdDSA Signature Schemes*. <https://signal.org/docs/specifications/xeddsa/>. Specification. Accessed: 2026-06-29. 2016.
- [63] *ProVerif User Manual*. Accessed: 2026-06-26. URL: <https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf>.

- [64] T. Reddy, V. Smyslov, and S. Fluhrer. *Signature Authentication in the Internet Key Exchange Version 2 (IKEv2) using PQC*. Internet-Draft draft-ietf-ipsecme-ikev2-pqc-auth-08. Work in Progress. Internet Engineering Task Force, Apr. 2026. 19 pp. URL: <https://datatracker.ietf.org/doc/draft-ietf-ipsecme-ikev2-pqc-auth/08/>.
- [65] T. Reddy and H. Tschofenig. *Post-Quantum Cryptography Recommendations for TLS-based Applications*. Internet-Draft draft-ietf-uta-pqc-app-01. Work in Progress. Internet Engineering Task Force, Feb. 2026. 23 pp. URL: <https://datatracker.ietf.org/doc/draft-ietf-uta-pqc-app/01/>.
- [66] O. Regev. “On lattices, learning with errors, random linear codes, and cryptography”. In: *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*. STOC ’05. Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 84–93. DOI: [10.1145/1060590.1060603](https://doi.org/10.1145/1060590.1060603).
- [67] S. Ricci, P. Dobias, L. Malina, J. Hajny, and P. Jedlicka. “Hybrid Keys in Practice: Combining Classical, Quantum and Post-Quantum Cryptography”. In: *IEEE Access* 12 (2024), pp. 23206–23219. DOI: [10.1109/ACCESS.2024.3364520](https://doi.org/10.1109/ACCESS.2024.3364520).
- [68] R. L. Rivest, A. Shamir, and L. Adleman. “A method for obtaining digital signatures and public-key cryptosystems”. In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: [10.1145/359340.359342](https://doi.org/10.1145/359340.359342).
- [69] P. W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1484–1509. DOI: [10.1137/S0097539795293172](https://doi.org/10.1137/S0097539795293172).
- [70] Standards for Efficient Cryptography Group. *SEC 2: Recommended Elliptic Curve Domain Parameters*. Tech. rep. Version 2.0. Certicom Research, Jan. 2010. URL: <https://www.secg.org/sec2-v2.pdf>.
- [71] D. Stebila, S. Fluhrer, and S. Gueron. *Hybrid key exchange in TLS 1.3*. Internet-Draft draft-ietf-tls-hybrid-design-16. Work in Progress. Internet Engineering Task Force, Sept. 2025. 23 pp. URL: <https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/16/>.
- [72] *The Tamarin Prover*. Manual. Accessed: 2026-06-26. URL: https://tamarin-prover.com/manual/master/book/001_introduction.html.
- [73] *The Transport Layer Security (TLS) Protocol Version 1.3*. RFC 9846. July 2026. DOI: [10.17487/RFC9846](https://doi.org/10.17487/RFC9846).

Use of AI

Generative AI tools (Claude Sonnet 5, by Anthropic) were utilized in preparing this thesis. The AI tools were used to refine linguistic precision and improve sentence flow and cohesion across chapters. Any AI-generated output was thoroughly reviewed, verified, and edited by the author. All research questions, experiments, artifacts, conclusions, and final content of the thesis are entirely the author's own work and responsibility. The AI tools did not originate any part of the research or its findings. Importantly, the abstract was written entirely by the author without the use of AI tools.