CHAPTER 1

INTRODUCTION

Quantum computing threatens traditional cryptography used in TLS. This project implements a Pure Post-Quantum TLS 1.3 using Post-Quantum Cryptography algorithms: ML-KEM and ML-DSA, ensuring future-proof secure communication and evaluating its performance.

1.1 CLASSICAL CRYPTOGRAPHY

Classical Cryptography refers to traditional encryption methods used to secure digital communication and data. These methods rely on mathematical problems that are hard to solve with Classical computers.

Classical Cryptography is primarily based on problems from Number Theory, Abstract Algebra and Finite Fields. Common Classical Cryptographic Algorithms that have been used so far include RSA (Rivest-Shamir-Adleman), ECC (Elliptical Curve Cryptography), and Diffie-Hellman.

These algorithms are widely used in internet security protocols like TLS, HTTP, and SSH.

1.1.1. Rivest-Shamir-Adleman Algorithm

Rivest-Shamir-Adleman Algorithm (RSA) is a type of Asymmetric Cryptographic algorithm that uses Encryption and Digital Signature. It is based on the Integer Factorization Problem. Its mathematical Background includes Prime Number Theory, Modular Arithmetic, and Euler's Theorem.

The Hardness of RSA lies in finding large prime factors is computationally hard, no polynomial-time algorithm exists for integer factorization on classical computers. This can be broken by Shor's Algorithm on a quantum computer.

The Public key (n,e) where $n = p \times q$, which is a product of large primes. To encrypt the message m, the ciphertext c is computed as $c = m^e \bmod n$. To decrypt the ciphertext c, $m = c^d \bmod n$ is computed where d is the modular inverse of e mode $\phi(n)$.

1.1.2. Elliptic Curve Diffie-Hellman Algorithm

Elliptic Curve Diffie-Hellman Algorithm (ECDH) is a type of Asymmetric Cryptographic algorithm that uses Secure Key Exchange. It is based on the Elliptic Curve Discrete Logarithm Problem (ECDLP). Its mathematical Background includes Group theory over elliptical curves, and Modular arithmetic.

The secureness of ECDH is because of hardness of ECDLP, no efficient classical algorithm to solve ECDLP. This can be broken by Shor's Algorithm on a quantum computer.

In this algorithm, there will be Alice key $\boldsymbol{k_A}$ and Bob's key $\boldsymbol{k_B}$. The shared secret key is calculated as $\boldsymbol{k_{AB}} = \boldsymbol{k_A} \cdot \boldsymbol{k_B}.\mathbf{G}$, where $\mathbf{G}$ is the generator.

1.1.3. Quantum Threat to Classical Cryptography

Quantum computers pose a major threat to classical cryptographic systems like RSA, Diffie-Hellman, and ECC. These systems rely on problems that are hard for classical computers, such as factoring large numbers or solving discrete logarithms. However, Shor's algorithm, a powerful quantum algorithm, can solve these problems in polynomial time, making it possible to break these encryption methods quickly once a large enough quantum computer is available.

Even symmetric cryptography isn't completely safe. Grover's algorithm allows quantum computers to perform brute-force attacks much faster—reducing the time needed from $\mathcal{O}(2^n)$ to $\mathcal{O}(2^{n/2})$. This weakens algorithms like AES and SHA

(Secure Hash Algorithm), though increasing key sizes can help. These developments highlight the urgent need for Post-Quantum Cryptography to secure data against future quantum threats.

1.2. TRANSPORT LAYER SECURITY

Transport Layer Security (TLS) 1.3 Handshake establishes a secure communication session between a client and server.

The typical TLS 1.3 Handshake follows the given steps:

1. *ClientHello* - Client initiates handshake by sending a ClientHello message (TLS version, ClientRandom/*KeyShare,* CipherSuites…)
2. *ServerHello* - Server responds with ServerHello message (TLS version selected, ServerRandom/*KeyShare*, Selected CipherSuite…)
3. Server and Client derive the same *SharedSecret* from their private and other party's public key.
4. *ServerCertificate* - Server sends its X.509 certificate (ServerPublicKey, CA…) and *ServerFinished*. Client verifies ServerCertificate and sends *ClientFinished*.
5. The *SharedSecret* is fed to Key Derivation Function (KDF) along with handshake transcript which includes random nonces from both Client and Server to derive a set of symmetric keys for *SessionKeys*.
6. TLS Handshake completed and encrypted communication is valid using symmetric session key.

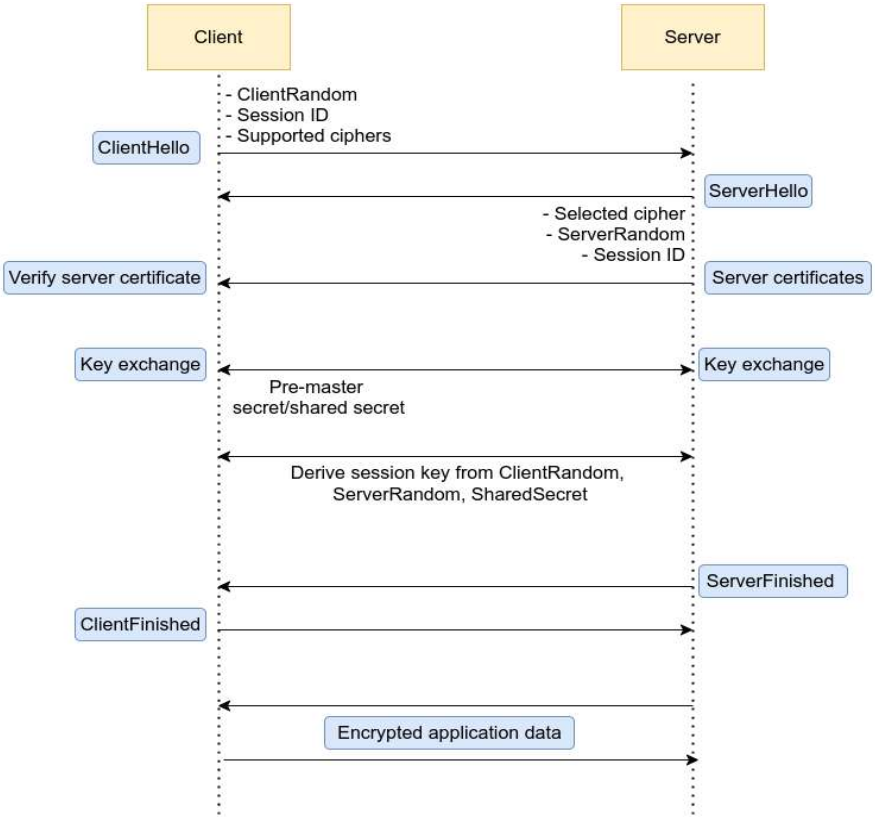


Figure 1.1 TLS Handshake

1.3. POST-QUANTUM CRYPTOGRAPHY

Post-Quantum Cryptography (PQC) refers to cryptographic methods designed to secure digital communications against the potential threats posed by quantum computers. While current encryption algorithms like RSA, and ECC are secure against classical computers, they could efficiently solve problems that underpin these classical systems.

PQC aims to develop new cryptographic algorithms that remain secure even in the presence of quantum computing capabilities. These algorithms are based on mathematical problems believed to be resistant to quantum attacks, such as lattice-based, hash-based, code-based, and multivariate polynomial problems.

1.3.1. Lattice-Based Cryptography

This Lattice-based cryptography is based on the hardness of finding the short or close vectors in a high-dimensional lattice. The problem type includes Shortest Vector Problem (SVP), Closest Vector Problem (CVP), and Learning with Error problem (LWE).

The hardness of this type of cryptography is because, lattice problems grow exponentially in complexity with the lattice dimension, and it is best known classical and quantum algorithms are inefficient for solving in high dimensions.

The example schemes included in this are ML-KEM (key encapsulation) and ML-DSA (digital signature).

1.3.1.1. Module-lattice key encapsulation mechanism

Module-Lattice Key Encapsulation Mechanism (ML-KEM) is a post-quantum cryptographic scheme standardized by NIST in FIPS 203, derived from the CRYSTALS-Kyber algorithm. It facilitates the establishment of a shared secret key between two parties over an insecure channel, which can then be used for symmetric encryption.

The mechanism includes three core algorithms:

i. KeyGen: Generates a public encapsulation key and a private decapsulation key.

ii. Encaps: Uses the public key to produce a ciphertext to recover the shared secret.

iii. Decaps: Utilizes the private key and receives ciphertext to recover the shared secret.

ML-KEM's security is grounded in the Module Learning with Error (MLWE) problem, a complex mathematical challenge believed to be resistant to quantum

attacks. Additionally, it employs the Number-Theoretic Transform (NTT) to optimize polynomial operations, enhancing computational efficiency.

1.3.1.2. Module-lattice digital signature algorithm

Module-Lattice Digital Signature Algorithm (ML-DSA) is a post-quantum digital signature scheme standardized by NIST in FIPS 204. It is based on the CRYSTAL-Dilithium algorithm and is designed to provide strong security against quantum computer attacks. ML-DSA ensures strong existential unforgeability under chosen message attacks, making it suitable for securing digital communications in a post-quantum world.

The algorithm operates through three primary functions:

i. KeyGen: Generates a pair of keys - public and private for signing and verification.

ii. Sign: Creates a digital signature for a given message using the private key.

iii. Verify: Validates the authenticity of the signature using the public key.

ML-DSA's security is rooted in hard lattice problems, especially the Module Learning with Error (MLWE) and Module Short Integer Solution (MSIS) problem. The signing process involves an interactive protocol with three steps: Commitment, Challenge, and Response. This approach ensures that the signature is both secure and efficient, making ML-DSA a robust choice for digital signatures in the era of quantum computing.

1.3.2. Benefits

Quantum-Resistant Security: PQC algorithms are designed to withstand attacks from both classical and quantum computers, ensuring the long-term confidentiality, integrity, and authenticity of digital communications.

Future-Proofing Data Protection: By adopting PQC, organizations can safeguard sensitive information against potential future quantum-enabled breaches, addressing concerns like "harvest now, decrypt later" scenarios.

Compatibility with Existing Systems: PQC algorithms can be integrated into current communication protocols and infrastructures, facilitating a smoother transition without the need for entirely new systems.

Standardization and Global Adoption: Initiatives by organizations like NIST are leading to the development and standardization of PQC algorithms, promoting widespread adoption and interoperability across various platforms.

1.4. OBJECTIVE

The primary goal of this project is to implement a pure Post-Quantum Cryptography (PQC) approach within the TLS 1.3 protocol, replacing the classical cryptographic mechanisms with quantum-resistant alternatives for both key encapsulation and authentication. This involves integrating PQC primitives such as ML-KEM and ML-DSA, which are lattice-based cryptographic algorithms designed to resist attacks from quantum computers.

To evaluate the deployment of these algorithms and develop a custom test environment with custom build Root CA Certificate signed using PQC algorithm. This setup allows comprehensive benchmarking of the performance of PQC-enabled TLS, including aspects such as handshake time, certificate size, and round-trip delays.

Furthermore, the project aims to assess the impact of adopting Pure PQC in TLS by analysing security benefits, computational overhead, and compatibility challenges. This evaluation is crucial for understanding the trade-offs and feasibility of migrating to post-quantum secure communication protocols in real-world applications.

CHAPTER 2

LITERATURE SURVEY

Rios, R., Montenegro, J. A., Muñoz, A., and Ferraris, D. in [1] benchmarked Post-Quantum TLS using a framework built on OpenSSL, libOQS, and oqs-provider within a dockerized environment. They evaluated handshake efficiency and transmission overhead, reporting ML-KEM as comparable to traditional algorithms, FALCON as the most efficient signature scheme, and SPHINCS+ as slower due to its hash-based structure. While their results provide valuable insights under controlled lab settings, the study did not account for real-world web traffic conditions or hybrid TLS configurations, limiting its practical applicability.

Kupcova, E., Simko, J., Pleva, M., and Drutarovsky, M. in [2] proposed an experimental framework for Post-Quantum TLS using a public Open Quantum Safe server. They found that PQC integration requires changes in identifiers and CA support. While certificate verification was efficient, large key sizes and system performance variations highlighted the need for further optimization.

Chen, A. C. H. in [3] conducted an empirical analysis of Post-Quantum TLS handshakes, comparing PQC and classical schemes. Saber and CRYSTALS-KYBER performed well for key exchange, with FALCON as the top signature scheme. While performance was comparable to classical methods, the study lacked setup details and did not test FALCON at level 3.

Chen, A. C. H. in [4] examined Post-Quantum Cryptography integration into X.509 certificates, analyzing RSA and ECC vulnerabilities. Case studies on Falcon, Dilithium, and SPHINCS assessed certificate size and efficiency.

However, TLS integration was not tested, and simulations lacked real-world applicability.

Fan, J., Willems, F., Zahed, J., Gray, J., Mister, S., Ounsworth, M., and Adams, C. in [5] analyzed the impact of post-quantum hybrid certificates on PKI and TLS. Using SPHINCS+ in hybrid X.509 certificates, they observed large increases in certificate size and handshake time, with OpenSSL failing above 100KB. Lattice-based algorithms were not evaluated.

Giron, A., Custódio, R., and Rodríguez-Henríquez, F. in [6] conducted a systematic mapping study on post-quantum hybrid key exchange. They categorized hybrid KEX methods, discussed key derivation and data transport strategies, and noted that rekeying limits data per key. Efficiency was found to vary by protocol and environment.

Alnahawi, N., Müller, J., Oupický, J., and Wiesmaier, A. in [7] surveyed Post-Quantum TLS, comparing classical and PQC schemes in terms of security, efficiency, and overhead. They found large signatures like SPHINCS+ reduced performance and emphasized that hybrid methods lack formal security proofs and require system-wide upgrades for deployment.

Sowa, J. et al. in [8] introduced a PQC network instrument deployed at NCSA and the Fabric testbed to assess adoption trends and support migration planning. The study focused on PQC integration in SSH, TLS, and scientific applications, identifying challenges in achieving full quantum resistance. It also revealed that most network protocols beyond SSH and TLS are not yet ready for PQC adoption.

Kampanakis, P. and Kallitsis, M. in [9] proposed a method to accelerate (D)TLS handshakes by suppressing intermediate CA certificates after caching. Their analysis of various caching strategies showed reduced handshake latency,

but also raised concerns about passive attackers identifying suppression patterns and potential failures in certificate retrieval due to cache misses.

Raavi, M., Chandramouli, P., Wuthier, S., Zhou, X., and Chang, S.-Y. in [10] analyzed the performance of NIST PQC algorithms in PKI using OpenSSL and libOQS. They compared classical, post-quantum, and hybrid schemes in terms of memory, security, and processing overhead. The results showed increased resource usage for PQC and hybrids, with Falcon offering fast verification, while Rainbow lagged significantly.

CHAPTER 3

PROPOSED WORK

3.1 INTRODUCTION

With the rise of quantum computing, classical cryptographic algorithms like RSA and ECC are at risk, as quantum algorithms can efficiently break them. Post-Quantum Cryptography (PQC) offers alternatives that remain secure against both classical and quantum attacks. This work focuses on integrating PQC algorithms into the TLS 1.3 protocol, replacing classical key exchange and digital signatures. It uses: ML-KEM for key exchange and ML-DSA for digital signatures.

Two TLS configurations are proposed:

- **Pure PQC-based TLS** - using post-quantum algorithms for both certificates and key exchange.
- **Hybrid PQC-based TLS** - combining classical and post-quantum algorithms for gradual transition.

The implementation is evaluated through metrics like handshake time, certificate size, key exchange size, and round-trip time (RTT).

3.2 ALGORITHMS

The following are the algorithms that are used as part of Post-Quantum Cryptography algorithms to replace the classical counterparts in the TLS protocol.

These Post-Quantum cryptography algorithms are based on Lattice-based hard problems like Learning with errors (LWE) and Shortest Integer Solution (SIS). These mathematical problems are secure against attacks based on quantum algorithms.

3.2.1 Module-Lattice Key Encapsulation Mechanism

ML-KEM (Module-Lattice Key Encapsulation Mechanism) is a post-quantum key exchange algorithm designed to establish a shared secret between two parties securely. The algorithm is built around problems that remain computationally difficult, even for quantum computers.

It uses concepts such as the Learning with Errors (LWE) problem, but in a more efficient and structured form called Module-LWE.

The three main algorithms of KEM:

- KeyGen - generates encapsulation key (public) and decapsulation key (private)
- Encaps - encapsulates a shared secret key and produces a ciphertext (shared_secretkey, ciphertext)
- Decaps - recovers shared secret key from the ciphertext (shared_secretkey)

Domain Parameters:

For concreteness, we'll use the ML-KEM-768 domain parameters:

- $q = 3329$
- $n = 256$
- $k = 3$
- $\eta_1 = 2$
- $\eta_2 = 2$

Key Generation:

Alice does:

1. Select $A \in_R R_q^{k \times k}$, $s \in_R S_{\eta_1}^k$, and $e \in_R S_{\eta_2}^k$.
2. Compute:
$$t = As + e$$
3. Alice's encryption (public) key is (A, t); her decryption (private) key is s.

Note: Computing s from (A, t) is an instance of MLWE.

Encryption:

To encrypt a message $m \in \{0,1\}^n$ for Alice, Bob does:

1. Obtain an authentic copy of Alice's encryption key (A, t).
2. Select $r \in_R S^k_{\eta_1}$, $e_1 \in_R S^k_{\eta_2}$, and $e_2 \in_R S_{\eta_2}$.
3. Compute:

$$u = A^T r + e_1$$

$$v = t^T r + e_2 + \lceil \tfrac{q}{2} \rceil m$$

4. Output the ciphertext:

$$c = (u, v)$$

Note: $c \in R_q^k \times R_q$.

Decryption:

To decrypt $c = (u, v)$, Alice does:

1. Compute:

$$m = Round_q\left(v - s^T u\right)$$

Note: Alice uses her decryption key s.

Figure 3.1. ML-KEM Algorithm

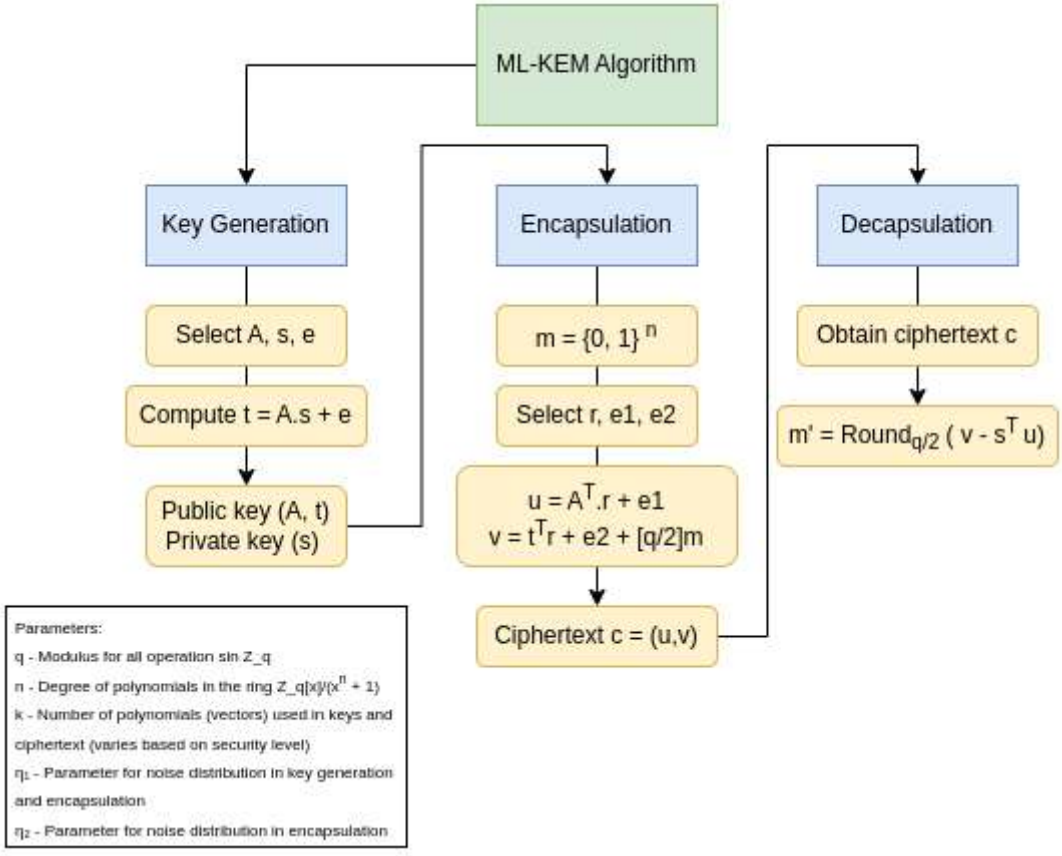


Figure 3.2 ML-KEM Algorithm Flow Diagram

3.2.2 Module-Lattice Digital Signature Algorithm

ML-DSA (Module Lattice Digital Signature Algorithm) is a post-quantum digital signature scheme designed to provide secure authentication and message integrity. Like ML-KEM, it's part of the family of lattice-based cryptography, and it is one of the candidates considered in the NIST Post-Quantum Cryptography standardization process.

ML-DSA is built upon the foundations of lattice-based problems, specifically based on the concept of Shortest Integer Solution (SIS) problem, structured in an efficient form called Module-SIS. It relies on the difficulty of finding certain specific patterns (short vectors) within large, complex, grid-like structures (lattices).

It introduces controlled randomness and clever construction techniques to make forging a valid signature computationally impractical, even for quantum attackers.

Domain Parameters:

- Field modulus: $q = 2^{23} - 2^{13} + 1$
- Dimension parameter: $n = 256$
- Matrix size: $(k, \ell) = (8, 7)$
- Secret key bounds: $\eta = 2$
- Commitment bound: $\gamma_1 = 2^{19}$
- Challenge length: $\tau = 60$
- Response bound: $\beta = \tau\eta = 120$
- Hash function: $H : \{0, 1\}^* \rightarrow B_\tau$

Key Generation (Alice):

Alice generates her keys as follows:

- Select:
  - A uniformly random public matrix $A \in R_q^{k \times \ell}$.
  - A small secret vector $s_1 \in R_\eta^\ell$.
  - A small secret vector $s_2 \in R_\eta^k$.
- Compute:

$$t = As_1 + s_2$$

- Alice's verification (public) key is (A, t), and her signing (private) key is (s_1, s_2).

Computing s_1 from (A, t) is an instance of the Module Learning With Errors (MLWE) problem, which is computationally hard.

Signature Generation:

To sign a message M, Alice performs:

- Select a random masking vector:

$$y \in R_{\gamma_1}^\ell$$

- Compute commitment:

$$w = Ay$$

- Compute challenge:

$$w_1 = HighBits(w)$$
$$c = H(M \parallel w_1)$$

- Compute response:

$$z = y + cs_1$$

- Output the signature:

$$\sigma = (c, z)$$

Signature Verification:

Bob verifies Alice's signature $\sigma = (c, z)$ on message M as follows:

- Obtain an authentic copy of Alice's public key (A, t).
- Compute the commitment reconstruction:

$$w' = Az - ct$$

- Since $z = y + cs_1$, we have:

$$Az = Ay + cAs_1 = w + c(t - s_2)$$
$$Az - ct = w - cs_2$$

- Since s_2 has small coefficients, the LowBits of $w - cs_2$ remain small:

$$HighBits(w - cs_2) = HighBits(w)$$

- Therefore, Bob computes:

$$w_1' = HighBits(Az - ct) = HighBits(w - cs_2) = HighBits(w) = w_1$$

- Verify that:

$$c = H(M \parallel w_1')$$

Figure 3.3 ML-DSA Algorithm

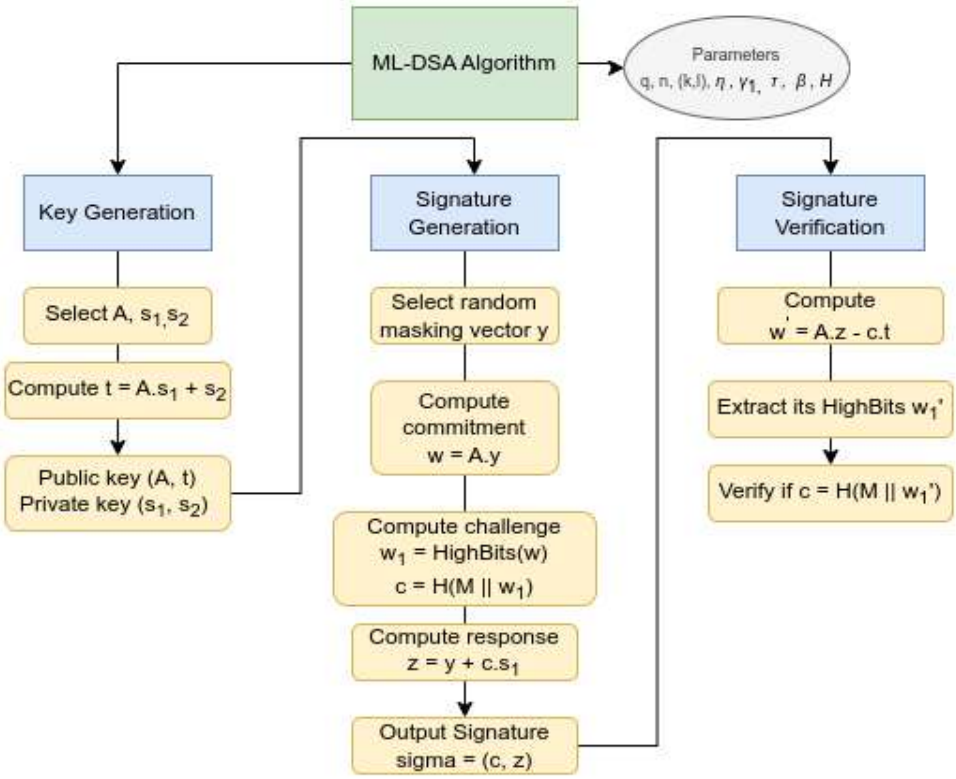


Figure 3.4 ML-DSA Algorithm Flow Diagram

3.3 POST-QUANTUM CRYPTOGRAPHY IN TLS

The proposed work aims to implement Post-Quantum Cryptography algorithms such as ML-KEM and ML-DSA in TLS 1.3 protocol, replacing their classical counterparts. The integration of PQC into TLS can be classified into two distinct approaches based on the type of cryptographic algorithms used for digital certificates and key exchange:

- Pure PQC-based TLS
- Hybrid PQC-based TLS

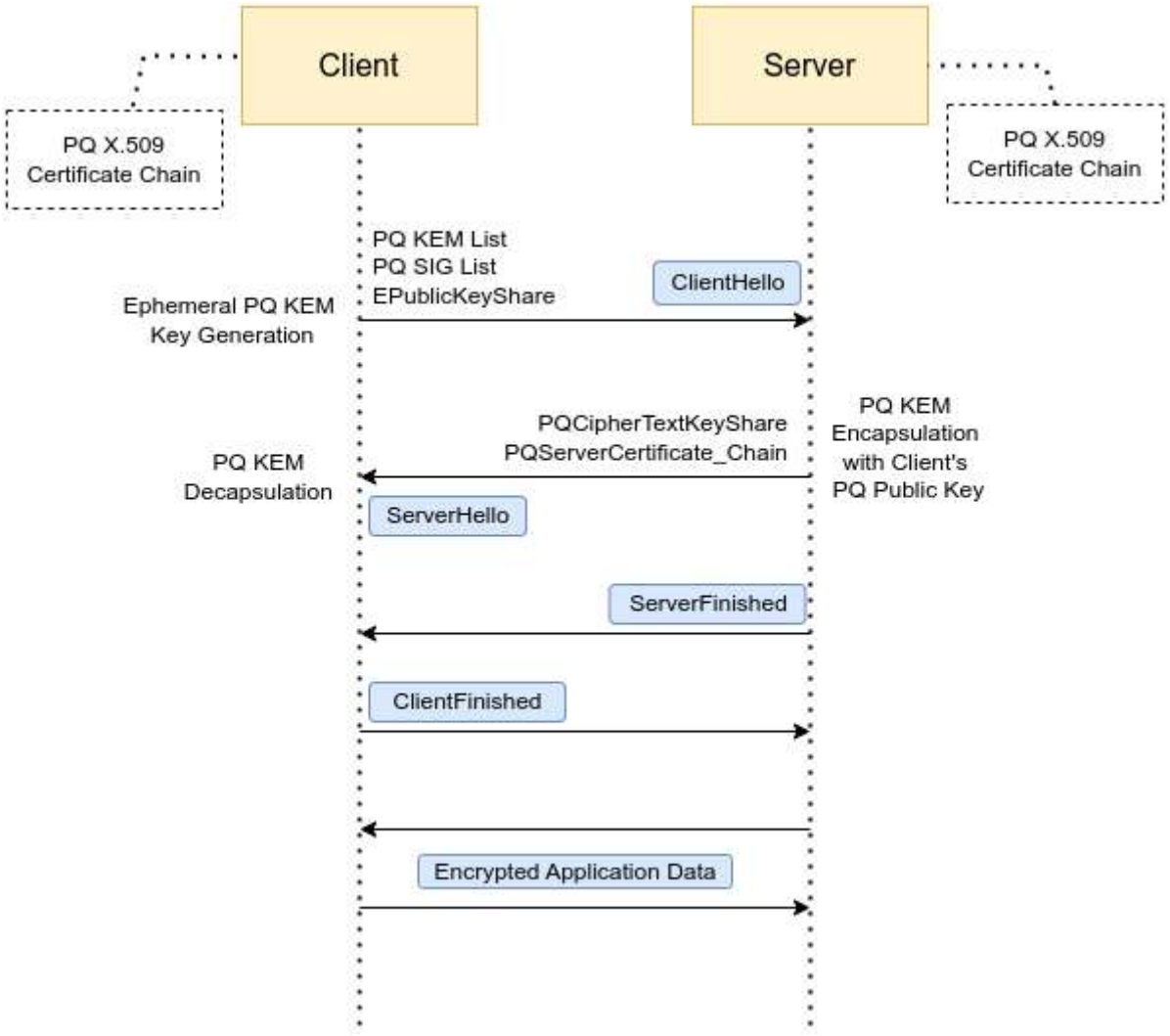


Figure 3.5 Post-Quantum based TLS

3.3.1 Pure PQC - TLS

In the Pure PQC-based TLS configuration, only Post-Quantum algorithms are used throughout the entire TLS trust chain and handshake process. This includes:

- Digital Certificates: Both the Root Certificate Authority (CA) and the Server certificates are generated using the ML-DSA algorithm for digital signatures. This ensures that the entire certificate chain, from the root to the end-entity (server), is secured using quantum-resistant signatures.
- Key Exchange: The key exchange mechanism within the TLS 1.3 handshake is performed using ML-KEM. This provides quantum-safe

protection for establishing the shared session keys used to encrypt communication between the client and server.

3.3.2 Hybrid PQC - TLS

In the Hybrid PQC-based TLS configuration, a combination of classical and post-quantum algorithms is used to gradually transition to quantum-safe security. This includes:

- Digital Certificates: The Root Certificate Authority (CA) continues to use a classical cryptographic algorithm such as RSA for signing its certificate. This maintains compatibility with existing infrastructure and trust anchors that still rely on classical cryptography.

  The Server certificate, however, is generated using the ML-DSA algorithm. This ensures that the server's identity is authenticated using a quantum-resistant signature while maintaining trust by chaining back to a classical root certificate.
- Key Exchange: As in the pure approach, the key exchange mechanism in the TLS 1.3 handshake uses ML-KEM. This secures the session keys against potential quantum attacks, ensuring the confidentiality of the encrypted data.

3.4 PERFORMANCE METRICS

The various performance metrics that are used to evaluate the performance of TLS when using Pure PQC based and Hybrid PQC based schemes are listed below. These metrics help measure the efficiency, overhead, and practical impact of integrating post-quantum algorithms into the TLS protocol.

3.4.1. Handshake Time

Time taken from initiating the TLS handshake (Client Hello) to its successful completion (Finished message), including key exchange, certificate verification, etc.

3.4.2. Certificate Size

Tle file size of the X.509 Root CA certificate during TLS handshake, typically in bytes or Kilobytes.

3.4.3. Key Exchange Size

Size of the key exchange message sent during the handshake. In PQC, these keys can be large depending on the algorithm.

3.4.4. Round Trip Time

The time it takes for a signal to travel from the client to the server and back. It reflects network latency and affects how fast the handshake completes.

CHAPTER 4

IMPLEMENTATION

The implementation of the proposed work involves the creation of a controlled test environment to evaluate the performance and integration of post-quantum cryptographic algorithms in the TLS protocol. This is achieved by deploying a custom-built NGINX server configured to use OpenSSL integrated with the libOQS (Open Quantum Safe) library, enabling support for both hybrid and pure PQC-based TLS configurations.

4.1 TOOLS USED

1. OpenSSL

OpenSSL is an open-source library that provides support for TLS/SSL protocols and a wide range of cryptographic functions. It is a widely used toolkit for implementing secure communications over networks. In this project, OpenSSL is extended with support for Post-Quantum Cryptography (PQC) algorithms through the integration of libOQS and OQS-Provider, enabling the use of quantum-resistant key exchange and digital signature schemes within the TLS protocol.

2. libOQS

libOQS (Open Quantum Safe) is a C library that offers implementations of post-quantum cryptographic algorithms, specifically focusing on key exchange and digital signature schemes such as ML-KEM and ML-DSA, among others. It includes algorithms selected and recommended by the NIST Post-Quantum Cryptography standardization process. libOQS provides a consistent API, making it easier to integrate these algorithms into cryptographic libraries like OpenSSL and other applications that require quantum-safe security.

3. OQS-Provider

OQS-Provider (Open Quantum Safe Provider) is a plugin designed for OpenSSL 3.x that connects libOQS with OpenSSL, enabling the use of post-quantum algorithms in TLS communications. It registers PQC algorithms from libOQS within OpenSSL and handles the necessary key exchange and digital signature operations during the TLS handshake, effectively bridging the gap between traditional cryptography frameworks and quantum-safe implementations.

4. NGINX

NGINX is a high-performance, open-source web server and reverse proxy that is widely used for hosting secure web services. It supports OpenSSL-based TLS configurations, making it suitable for integrating with the OQS-extended version of OpenSSL. By doing so, NGINX can facilitate secure, quantum-resistant HTTPS connections using post-quantum cryptographic algorithms.

5. cURL

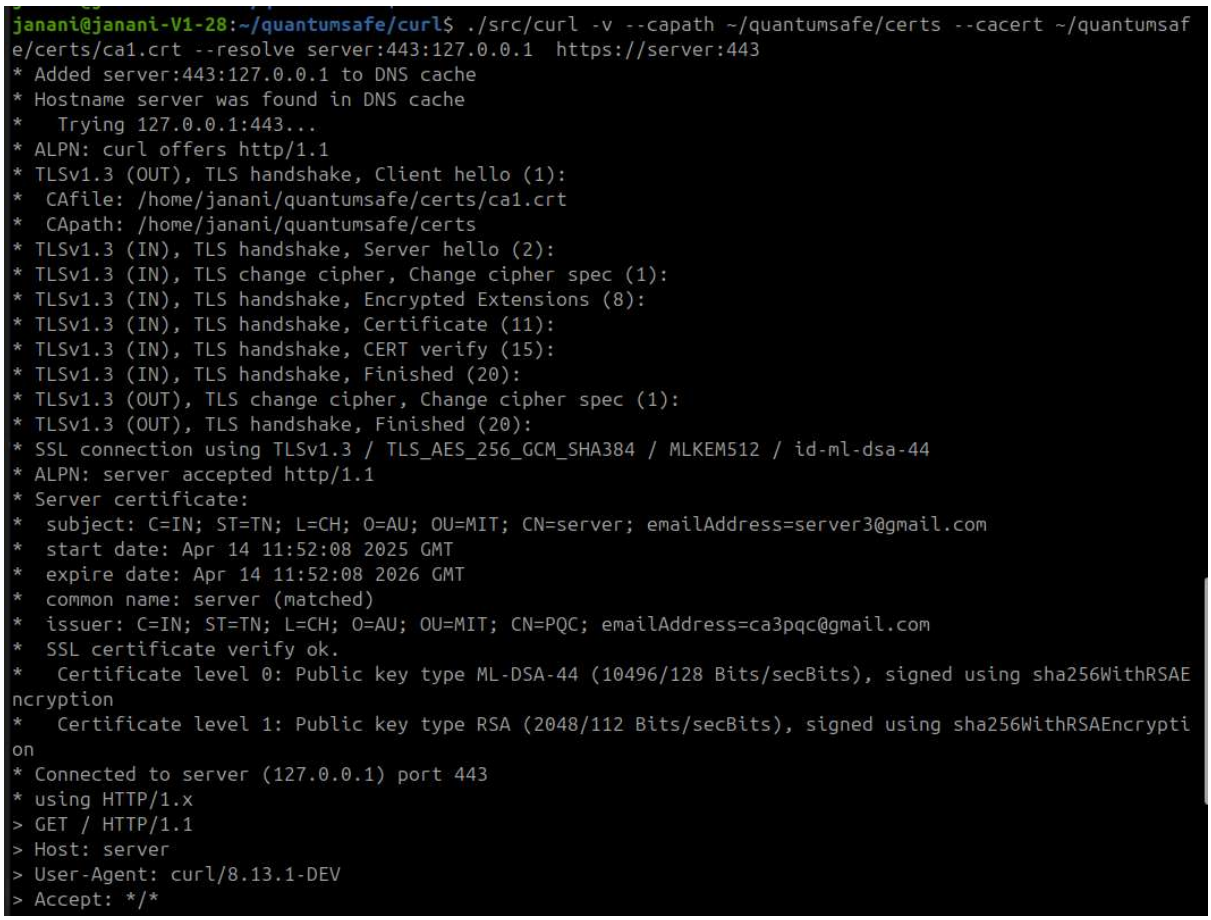
cURL (Client URL) is a command-line tool used for transferring data using various protocols such as HTTP and HTTPS. It is particularly useful for testing TLS connections, including those secured with post-quantum cryptography. In this project, cURL is employed to initiate and verify PQC-secured TLS handshakes with the NGINX server, ensuring the proper functioning of ML-KEM and ML-DSA within the TLS protocol.

4.2 HYBRID PQC - TLS

In the Hybrid PQC-based TLS implementation:

- The Root CA certificate is generated using a classical cryptographic algorithm (RSA), maintaining compatibility with existing trust infrastructures.
- The Server certificate is created using the ML-DSA post-quantum digital signature algorithm, offering quantum-safe authentication at the server level.
- During the TLS handshake, the key exchange mechanism uses ML-KEM, providing a quantum-resistant method for securely sharing session keys.

Algorithms used: RSA, MLDSA-44, MLKEM-512



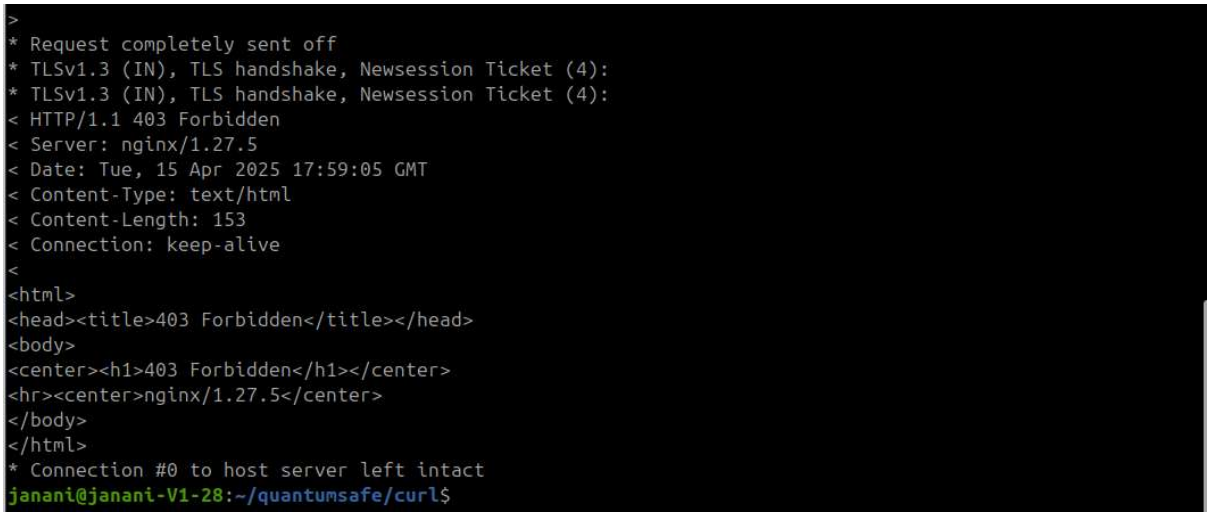


Figure 4.1 Hybrid PQC – TLS Implementation

4.3 PURE PQC - TLS

In the Pure PQC-based TLS implementation:

- Both the Root CA certificate and the Server certificate are generated using the ML-DSA algorithm. This ensures that the entire certificate chain is post-quantum secure, removing reliance on classical cryptography.
- The key exchange process is performed using ML-KEM, securing the handshake against quantum attacks.
- This setup provides end-to-end post-quantum security, from certificate validation to key exchange, ensuring future-proof protection for TLS communications.

Algorithms used: MLDSA-44, MLKEM-512

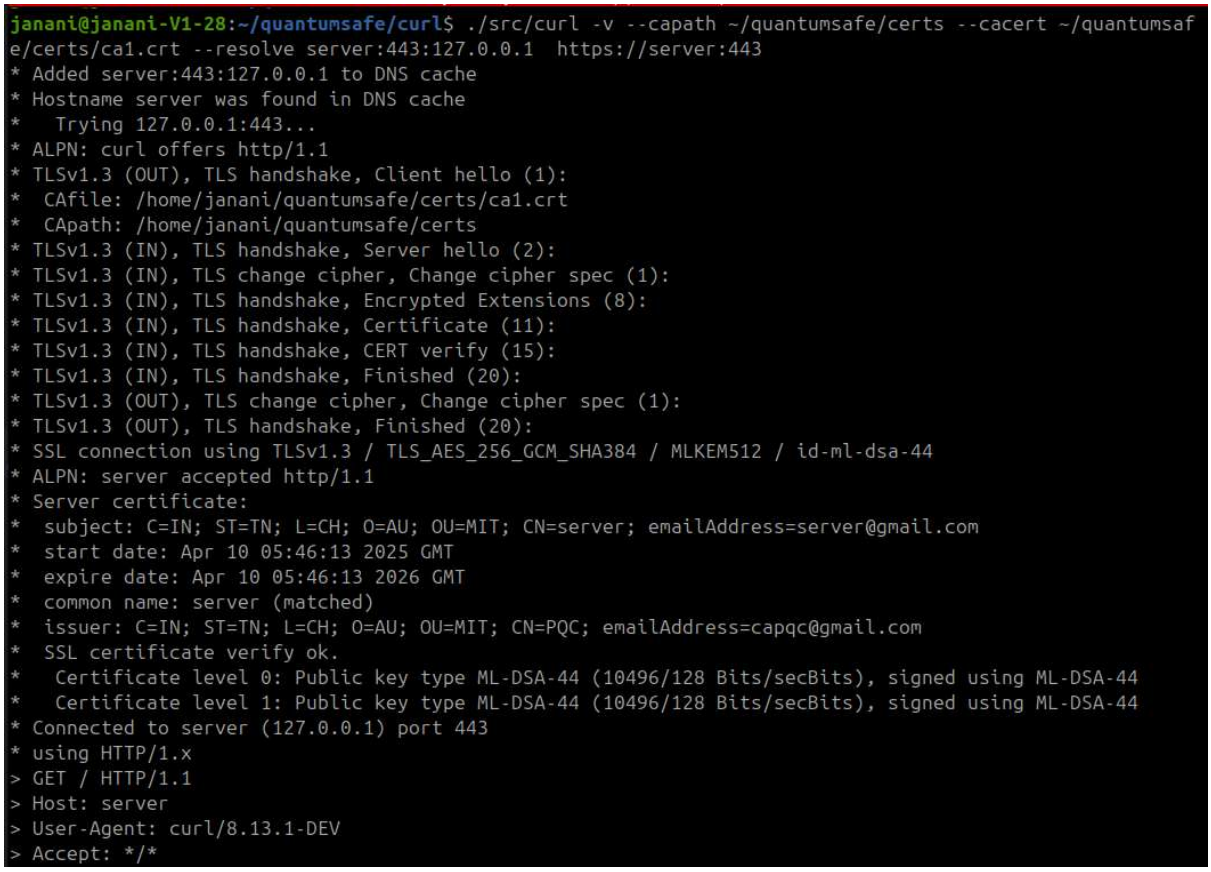


Figure 4.2. Pure PQC – TLS Implementation

CHAPTER 5

RESULT AND ANALYSIS

5.1 EVALUATION METRICS

5.1.1. Handshake Time

Handshake Time is the total time taken to complete the TLS handshake, starting from the client's initial ClientHello message to the successful completion of the handshake with the Finished message.

TLS Handshake Time = Time ($TLS_{Finished\_message}$) - Time ($TLS_{ClientHello}$)

tls

No.	Time	Source	Destination	Protocol	Length	Info
4	0.017114388	127.0.0.1	127.0.0.1	TLSv1.3	1230	Client Hello (SNI=server)
6	0.017656452	127.0.0.1	127.0.0.1	TLSv1.3	4162	Server Hello, Change Cipher Spec, Application Data
8	0.018131726	127.0.0.1	127.0.0.1	TLSv1.3	3597	Application Data, Application Data, Application Data
10	0.020218745	127.0.0.1	127.0.0.1	TLSv1.3	146	Change Cipher Spec, Application Data
11	0.020350560	127.0.0.1	127.0.0.1	TLSv1.3	162	Application Data
12	0.020437706	127.0.0.1	127.0.0.1	TLSv1.3	353	Application Data
13	0.020487369	127.0.0.1	127.0.0.1	TLSv1.3	353	Application Data
15	0.020556655	127.0.0.1	127.0.0.1	TLSv1.3	396	Application Data
16	0.020638137	127.0.0.1	127.0.0.1	TLSv1.3	90	Application Data

Figure 5.1. Handshake Time

5.1.2. Certificate Size

Certificate Size refers to the file size of the X.509 Root CA certificate exchanged during the TLS handshake. It is typically measured in bytes or kilobytes.

```
janani@janani-V1-28:~/quantumsafe/pqc-certs/ca$ ls -l ca.crt
-rw-rw-r-- 1 janani janani 5697 Apr 10 11:06 ca.crt
janani@janani-V1-28:~/quantumsafe/pqc-certs/server$ ls -l server.crt
-rw-rw-r-- 1 janani janani 5677 Apr 10 11:16 server.crt
```

Figure 5.2. Certificate Size

5.1.3. Key Exchange Size

Key Exchange Size represents the size of the key exchange message transmitted during the TLS handshake.

Key Exchange Size = Size of Key share field in TLS Handshake

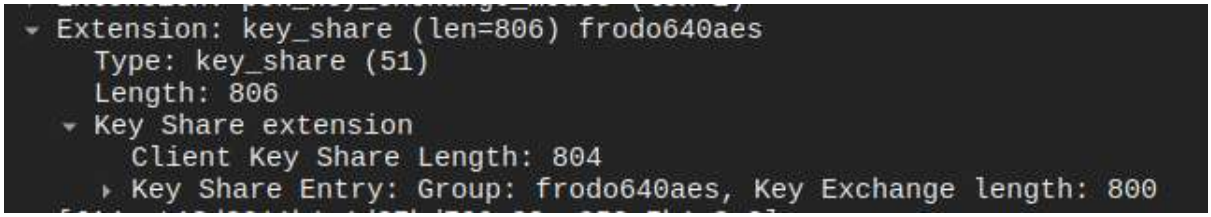


Figure 5.3. Key Exchange Size

5.1.4. ROUND TRIP TIME

Round Trip Time (RTT) is the time taken for a message to travel from the client to the server and back. It reflects network latency and affects how fast the handshake completes.

RTT = Time (ServerHello) – Time (ClientHello)

Figure 5.4. Round Trip Time (RTT)

5.2 PERFORMANCE ANALYSIS

5.2.1. Pure PQC - TLS

Table 5.1 Performance analysis across Pure PQC security levels

Root CA Algorithm	Security Level	Key Exchange Algorithm	Signature Algorithm	Handshake Time (ms)	Certificate size (bytes)		Key Exchange Length (bytes)	RTT (Round Trip Time) (ms)
					Root CA	Server		
MLDSA44	1	MLKEM512	MLDSA44	2.5415523	5697	5677	800	0.3458945
MLDSA65	3	MLKEM768	MLDSA65	2.620661	7769	7651	1184	0.357551
MLDSA87	5	MLKEM1024	MLDSA87	2.8391154	10422	10304	1568	0.3620452

5.2.2. Hybrid PQC - TLS

Table 5.2 Performance analysis across Hybrid PQC security levels

Root CA Algorithm	Security Level	Key Exchange Algorithm	Signature Algorithm	Handshake Time (ms)	Certificate size (bytes)		Key Exchange Length (bytes)	RTT (Round Trip Time) (ms)
					Root CA	Server		
RSA	1	MLKEM512	MLDSA44	3.5840696	1367	2659	800	1.1983031
RSA	3	MLKEM768	MLDSA65	3.6592075	1367	3524	1184	1.2344027
RSA	5	MLKEM1024	MLDSA87	4.1899918	1367	4389	1568	1.3166472

5.2.2.1. Classical cryptography algorithm

Table 5.3 Performance analysis across Classical Cryptography Algorithm

Root CA Algorithm	Key Exchange Algorithm	Signature Algorithm	Handshake Time (ms)	Certificate size (bytes)		Key Exchange Length (bytes)	RTT (Round Trip Time) (ms)
				Root CA	Server		
RSA	X25519	RSA	1.8071568	1712	1424	32	0.6668389

5.2.3 Graphs

5.2.3.1 Handshake time variation between pure PQC, hybrid PQC and classical approaches:

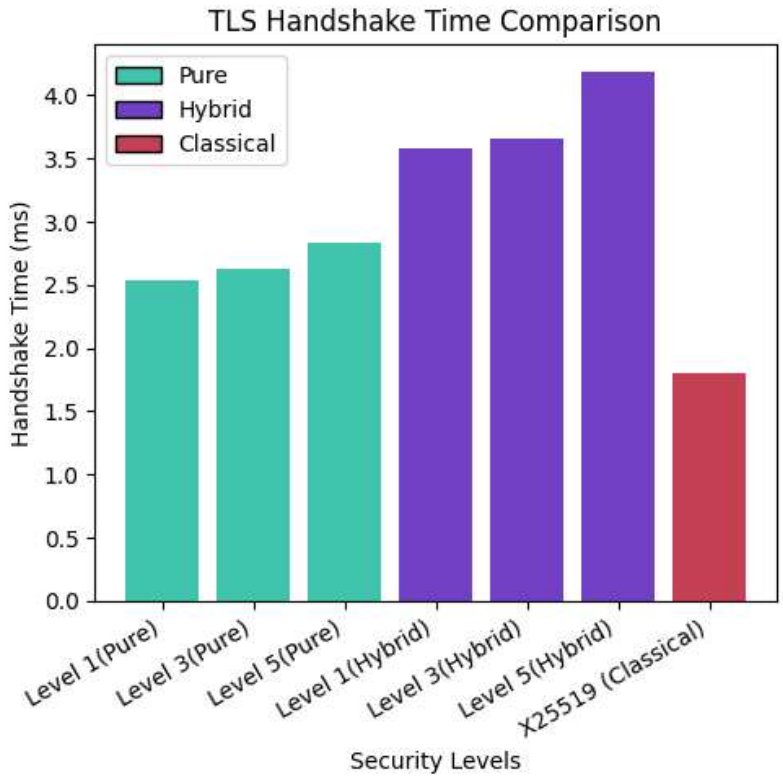


Figure 5.5 Handshake time comparison

5.2.3.2 Certificate size variation between pure PQC, hybrid PQC and classical approaches:

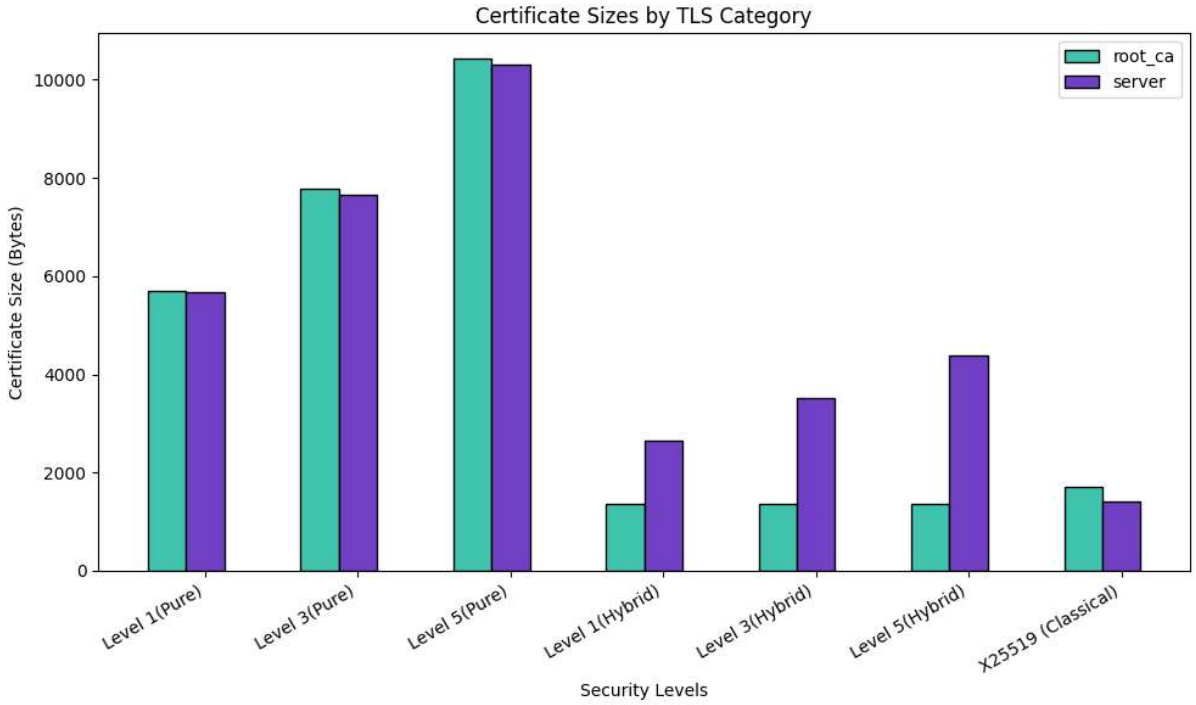


Figure 5.6 Certificate size variation

5.2.3.3 Key exchange length variation between pure PQC, hybrid PQC and classical approaches:

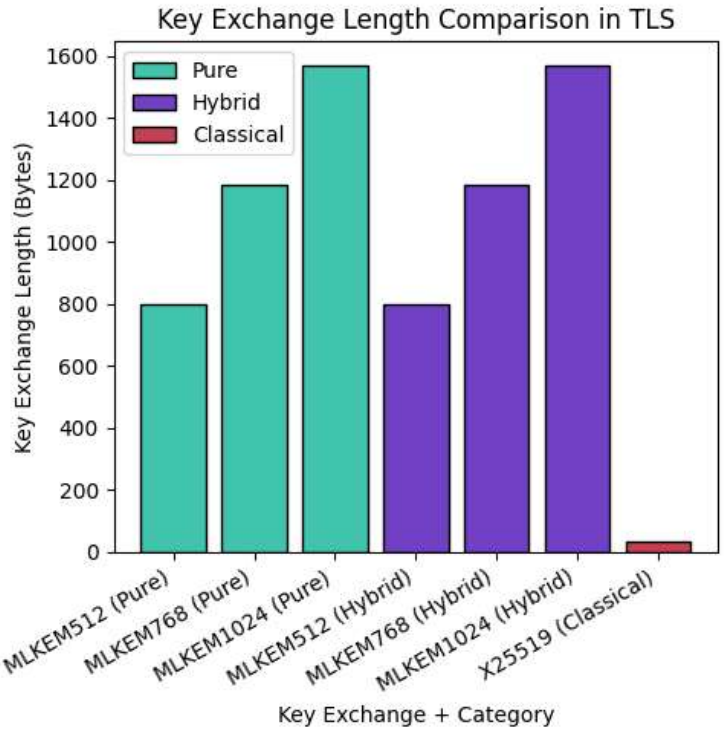


Figure 5.7 Key exchange length variation

5.2.3.4 Round trip time variation between pure PQC, hybrid PQC and classical approaches:

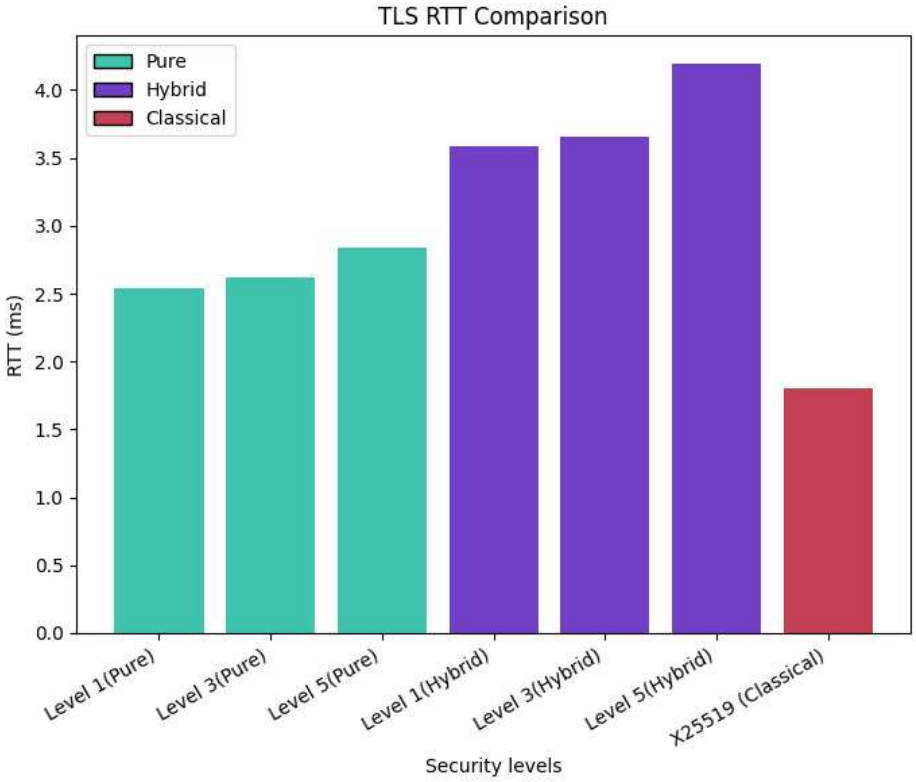


Figure 5.8 Round trip time variation

5.3 INFERENCES AND RESULTS

5.3.1. Handshake Time

Pure-PQC exhibits moderately increasing handshake times as the security level rises, ranging from 2.54 ms at Level 1 to 2.83 ms at Level 5.

In contrast, Hybrid-PQC incurs higher handshake times, starting at 3.58 ms and increasing to 4.19 ms due to the additional overhead of combining classical RSA operations with post-quantum key exchange and signatures.

Classical cryptography remains the fastest, with a handshake time of 1.80 ms.

Trend:

A clear positive correlation is observed between handshake time and security level in both Pure-PQC and Hybrid-PQC configurations, attributed to the increased size and complexity of key materials and cryptographic computations.

5.3.2. Certificate Size

In Pure-PQC, certificate sizes increase substantially as security levels rise - from 5697 bytes (Level 1) to 10422 bytes (Level 5).

In Hybrid-PQC, the Root CA certificate remains constant at 1367 bytes (RSA-based), while the server certificate size grows proportionally with the security level, from 2659 bytes to 4389 bytes.

Classical certificates are relatively small, with a fixed size of 1712 bytes.

Trend:

There is a significant growth in certificate size with increasing security levels in post-quantum schemes, driven by larger public keys and signatures embedded within the certificates.

5.3.3. Key Exchange Length

The key exchange payload size for both Pure-PQC and Hybrid-PQC increases with security level, ranging from 800 bytes (Level 1) to 1568 bytes (Level 5).

Classical X25519 key exchange is minimal, requiring only 32 bytes.

Trend:

Key exchange lengths scale consistently with security levels in post-quantum configurations, as stronger cryptographic assurances necessitate larger encapsulated key materials.

5.3.4. Round Trip Time

Interestingly, Pure-PQC achieves the lowest RTT(Round Trip Time) values, starting at 0.345 ms and reaching 0.362 ms at higher security levels.

Hybrid-PQC demonstrates higher RTT values, ranging from 1.198 ms to 1.316 ms, reflecting the additional processing involved in hybrid cryptographic operations.

Classical cryptography maintains a constant RTT of 0.667 ms.

Trend:

RTT increases slightly with security level in both Pure-PQC and Hybrid-PQC settings, influenced by larger key and certificate sizes. However, Pure-PQC consistently outperforms Hybrid-PQC in terms of RTT, primarily due to the elimination of RSA-based processing overhead.

CHAPTER 6

CONCLUSION AND FUTURE WORK

While Pure-PQC schemes in TLS do not outperform classical cryptography in terms of handshake time, certificate size, and key exchange length, they deliver something far more critical: quantum-resilient security. Classical cryptographic algorithms, though highly efficient, are fundamentally vulnerable to quantum attacks, and their security cannot be guaranteed in the post-quantum era.In essence, the slightly increased computational and communication costs of Pure-PQC in TLS are a necessary and worthwhile trade-off for achieving unconditional, long-term security against both classical and quantum adversaries - a level of protection that no classical system can offer.

As a future work, we aim to dockerize the Pure-PQC TLS setup with custom Root CA certificates, making it more adaptable for real-world deployment. This will ensure that the system not only provides quantum-safe security but also works efficiently in practical environments, where scalability and seamless integration are essential.

REFERENCES

[1] R. Rios, J. A. Montenegro, A. Muñoz and D. Ferraris (2025), "Towards the Quantum-Safe Web: Benchmarking Post-Quantum TLS." in IEEE Network

[2] E. Kupcova, J. Simko, M. Pleva and M. Drutarovsky (2024), "Experimental Framework for Secure Post-Quantum TLS Client-Server Communication", 2024 International Symposium ELMAR, Zadar, Croatia, 2024.

[3] R. Döring and M. Geitz (2022), "Post-Quantum Cryptography in Use: Empirical Analysis of the TLS Handshake Performance", NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium, Budapest, Hungary, 2022.

[4] A. C. H. Chen (2024), "Post-Quantum Cryptography X.509 Certificate," 2024 International Conference on Smart Systems for applications in Electrical Sciences (ICSSES), Tumakuru, India, 2024.

[5] Fan, Jinnan & Willems, Fabian & Zahed, Jafar & Gray, John & Mister, Serge & Ounsworth, Mike & Adams, Carlisle. (2021). “Impact of post-quantum hybrid certificates on PKI, common libraries, and protocols.” International Journal of Security and Networks.

[6] Giron, Alexandre & Custódio, Ricardo & Rodríguez-Henríquez, Francisco. (2022). “Post-quantum hybrid key exchange: a systematic mapping study.” Journal of Cryptographic Engineering.

[7] Nouri Alnahawi, Johannes Müller, Jan Oupický, and Alexander Wiesmaier (2024), “A Comprehensive Survey on Post-Quantum TLS” IACR Communications in Cryptology, vol. 1, no. 2, Jul 08, 2024.

[8] J. Sowa et al. (2024), "Post-Quantum Cryptography (PQC) Network Instrument: Measuring PQC Adoption Rates and Identifying Migration Pathways," 2024 IEEE International Conference on Quantum Computing and Engineering (QCE), Montreal, QC, Canada, 2024.

[9] Panos Kampanakis and Michael Kallitsis (2022), “Speeding Up Post-Quantum TLS handshakes by Suppressing Intermediate CA Certificates”, In Cyber Security, Cryptology, and Machine Learning: 6th International Symposium, CSCML 2022, Be'er Sheva, Israel, June 30 –

July 1, 2022, Proceedings. Springer-Verlag, Berlin, Heidelberg, 337–355.

[10] M. Raavi, P. Chandramouli, S. Wuthier, X. Zhou and S. -Y. Chang (2021), "Performance Characterization of Post-Quantum Digital Certificates," 2021 International Conference on Computer Communications and Networks (ICCCN), Athens, Greece, 2021.

[11] OpenSSL github: https://github.com/openssl/openssl.git.

[12] libOQS github: https://github.com/open-quantum-safe/liboqs.git.

[13] Integration and testing github: https://gist.github.com/ajbozarth/65aee2084f7bc089704b4e7afb54801e.

[14] Test Server by Open Quantum Safe Provider (OQS-Provider) reference: https://test.openquantumsafe.org/?utm_source=ibm_developer&utm_content=in_content_link&utm_id=tutorials_awb-quantum-safe-openssl