THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

Performance Analysis of Post-Quantum and Variant Algorithm on TLS 1.3

by
Peidong Liu

School of Information Technology and Electrical Engineering,
The University of Queensland.

Submitted for the degree of
Master of Computer Science

November 2024.

Originality Statement

The Dean
School of Information Technology and Electrical Engineering
University of Queensland
St Lucia, 4072

In accordance with the requirements of the degree of Master of Computer Science, I present the following thesis entitled Performance Analysis of Post-Quantum and Variant Algorithm on TLS 1.3. This work was performed under the supervision of Dr. Naipeng Dong.

I declare that the work submitted in this thesis is my own, except as acknowledged in the text and footnotes, and has not been previously submitted for a degree at the University of Queensland or any other institution.

Authors Signature

Peidong Liu

Contents

1 Introduction

1.1 Motivation

With the advancement of quantum computers, our digital security is facing new challenges. Traditional encryption techniques such as RSA (RivestShamirAdleman) and ECC (Elliptic Curve Cryptography) [1] are cryptographic algorithms that are commonly used to secure our information. However, quantum computers have the potential to break the difficult mathematical problems used in these encryption methods. Shors algorithm [2] and Grover [3]algorithm significantly reduces the time required for cracking. This greatly threatens many protocols that use traditional encryption algorithms, such as TLS 1.3. To counter this threat, post-quantum algorithms are created, which aims to resist cracking by quantum computers. But the security of many post-quantum algorithms has not yet been fully validated, and complete replacement of traditional algorithms is possible with unknown risks. In this context, it has become an important research task to merge post-quantum cryptography and traditional cryptography algorithms into a hybrid framework in TLS 1.3. The choice to focus on hybrid frameworks within TLS 1.3 is twofold. First, TLS 1.3 represents the latest standard for secure communication on the Internet, thus ensuring our research has immediate applicability and relevance. Second, the hybrid framework offers a pragmatic bridge between current cryptographic practices and future-proof solutions against quantum threats. This allows for both quantum security and the proven security of traditional algorithms. The hybrid strategy combines both traditional and post-quantum algorithms in Key Agreements (KAs) and Signature Algorithms (SAs) to protect encrypted data. This approach requires both components to be compromised for a security breach to occur, effectively creating a two-layer defense mechanism. This hybrid approach guarantees both traditional and post-quantum security while meeting the performance and interoperability requirements of existing Internet protocols. Research is focused on assessing the efficacy and performance of these post-quantum algorithms and their variants. Performance analysis is crucial for identifying potential impacts on user experience and operational efficiency. It also plays a critical role in the comparative evaluation between hybrid encryption frameworks and purely post-quantum solutions, offering insights into the trade-offs between enhanced security and system performance. As the digital infrastructure gradually adopts quantum-resistant technologies, understanding the implications of these transitions on performance becomes increasingly important.

2 Background

2.1 Basic Principles of Encryption Algorithms

Encryption algorithms are crucial for digital security, ensuring safe communication, data protection, and privacy. They convert readable data into coded text, decipherable only with a specific key. There are two primary types: symmetric, using one key for both encryption and decryption, exemplified by AES (Advanced Encryption Standard) [4] for its efficiency with large data volumes; and asymmetric, or public-key cryptography, such as RSA (Rivest Shamir Adleman) and ECC (Elliptic Curve Cryptography) [1] which uses a key pair, addressing the key distribution issue but at a slower pace, making it less suitable for large data sets. In this encrypted environment, Transport Layer Security (TLS) 1.3 has become the current standard for establishing a secure connection between networked computers, which is the most popular option for establishing a secure connection between two endpoints.The encryption algorithm in TLS is used for initial KA, handshake signature, and signature that constitutes the x.509 public key infrastructure (PKI). These three parts are the parts that are likely to face the threat of quantum computers in the future. Because of this, quantum computing advancements have also prompted the creation of post-quantum encryption to guard against future threats.

The rise of **Post-Quantum Cryptography (PQC)** is a response to the quantum computing threat, particularly against asymmetric algorithms. For all the algorithms used in the experiment, a detailed list is given later in the Selection of Parameters and Metrics section. Examples of post-quantum cryptographic approaches include:

Lattice-based cryptography is built on the mathematical structure of lattices in n-dimensional space. Its security primarily relies on two fundamental hard problems: The Shortest Vector Problem (SVP), which involves finding the shortest non-zero vector in a lattice, and The Learning With Errors (LWE) [5] problem, which involves solving a system of linear equations with small errors. Both problems have proven resistant to quantum computing attacks. Its versatility allows for various applications, from encryption and digital signatures to fully homomorphic encryption (FHE), enabling operations on encrypted data. Its resilience against quantum attacks, with no known quantum algorithms to break SVP and LWE, positions it as a strong option for future digital security. Current implementations demonstrate

the versatility of lattice-based approaches: FrodoKEM [5] , based on standard LWE, offers conservative security by avoiding additional algebraic structure ML-KEM [6] (formerly Kyber [7]), using Ring-LWE and Module-LWE for improved efficiency ML-DSA [8] (formerly Dilithium [9]) and Falcon [10] for digital signatures, leveraging structured lattices. These approaches exemplify robust theoretical security for encryption, key exchange, and digital signatures against quantum threats.

Hash-based cryptography relies on the one-way and collision-resistant properties of cryptographic hash functions for security. These fundamental properties have shown resilience against quantum attacks, with only Grover's algorithm providing a quantum speedup that merely reduces the security level by half, which can be compensated for by increasing hash output sizes.

Hash-based signatures are particularly noteworthy in the post-quantum landscape. SPHINCS+ [11] , a stateless hash-based signature scheme, represents a significant advancement in this field. Unlike traditional signature schemes, SPHINCS+ derives its security solely from the properties of hash functions, making it an attractive option for long-term post-quantum security. While it produces larger signatures and has slower signing operations compared to some alternatives, its well-understood security properties and minimal assumptions make it a reliable choice for applications requiring strong security guarantees.

The scheme's key advantages include:

- Minimal security assumptions based on well-studied hash function properties.
- Quantum resistance based on proven theoretical foundations.
- No requirement for additional mathematical structures.
- Stateless operation, avoiding the complexities of state management.

These characteristics make hash-based cryptography, particularly SPHINCS+, a conservative yet robust choice for post-quantum digital signatures in security-critical applications.

Coding-based cryptography, based on error-correcting code theory, derives its security from the NP-hard problem of decoding general linear codes. This approach has stood the test of time since McEliece's original proposal in 1978, and remains secure against known quantum attacks, including Shor's algorithm. Recent advancements in this field have produced more efficient schemes while maintaining robust security properties:

- BIKE [13] (Bit Flipping Key Encapsulation) represents a modern approach using quasi-cyclic codes, significantly reducing key sizes compared to traditional code-based systems
- HQC [14] (Hamming Quasi-Cyclic) offers another promising variant, balancing security and efficiency
- Both schemes are KEM (Key Encapsulation Mechanism) candidates, specifically designed for post-quantum secure key exchange

While historically challenged by large key sizes, modern code-based cryptographic schemes have made substantial progress in addressing this limitation through:

- Advanced algebraic structures like quasi-cyclic codes
- Improved encoding and decoding algorithms
- Optimized parameter selections balancing security and efficiency

These advancements have made code-based cryptography increasingly practical for real-world applications.

Multivariate polynomial cryptography derives its security from the proven NP-hard problem of solving systems of multivariate polynomial equations over finite fields. This mathematical foundation makes it inherently resistant to quantum attacks, as even quantum computers cannot efficiently solve NP-complete problems. Modern implementations in this field have evolved significantly:

- MAYO [15] represents a new generation of multivariate signature schemes, offering improvements over previous systems like Rainbow
- The scheme achieves a better balance between security and efficiency through optimized design choices
- It maintains the field's characteristic advantage of fast signature generation and verification

Key characteristics of multivariate cryptography include:

- Strong quantum resistance based on NP-complete mathematical problems.

- Extremely fast signature operations compared to other post-quantum approaches.
- Particularly suitable for resource-constrained environments where signing speed is crucial.
- Ongoing research to optimize key sizes while maintaining security guarantees.

2.2 Overview of TLS 1.3

TLS (Transport Layer Security) 1.3. Key features of TLS 1.3 include a simplified handshake process.The TLS handshake in TLS 1.3 requires only one round-trip (or back-and-forth communication) instead of two, which shortens the process by a few milliseconds.TLS 1.3 is faster and more secure than TLS 1.2.TLS 1.3 does not use weak cryptographic algorithms and insists on stronger cipher suites that support Perfect Forward Secrecy (PFS), thereby improved security. It masks session details by encrypting more of the handshake process. The protocol ensures forward secrecy through brief Diffie-Hellman [16] key exchanges.TLS 1.3 introduces efficient session recovery through "Pre-Shared Keys" (PSK), which improves speed and network efficiency. Despite substantial changes, it maintains backward compatibility, allowing a smooth transition and widespread adoption.

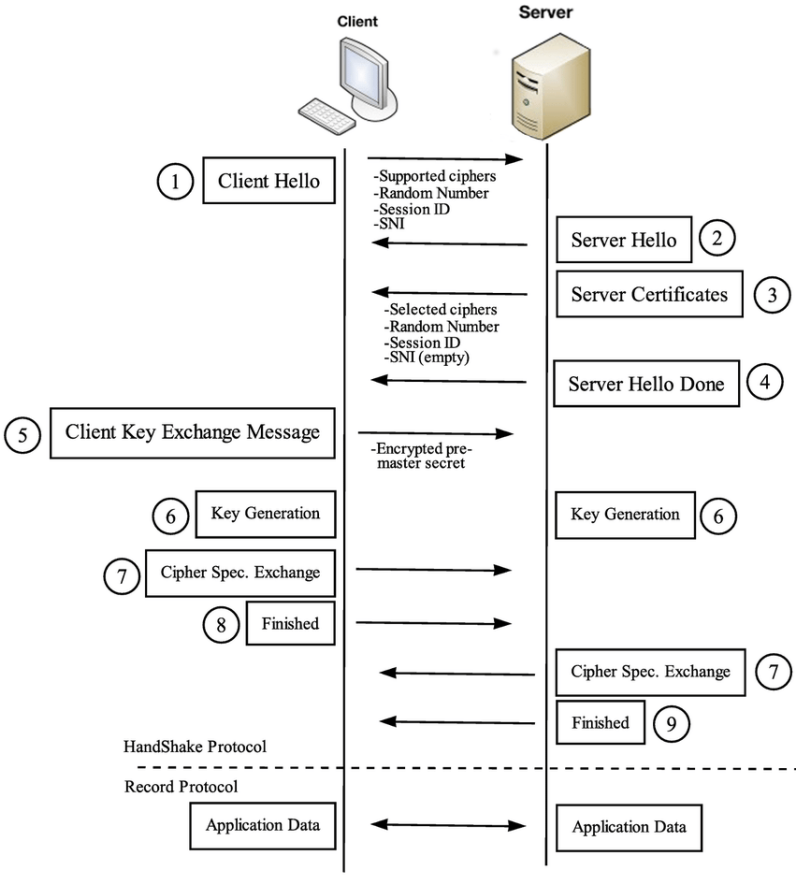


Figure 1: TLS hndshake process.

TLS 1.3 operates independently of particular Key Agreements (KAs) or Signature Algorithms (SAs), treating these as parameters that are negotiated during the handshake process. The key distinction of a Post-Quantum safe TLS from its traditional counterpart lies in its capability to negotiate Post-Quantum Cryptography (PQC) options during the handshake. This allows clients to prepare a key-share in anticipation of the KA that servers are likely to choose, thereby enabling handshakes to be completed in a single Round Trip Time (RTT). This sequence initiates with the client sending a Client Hello (CH) message, which is reciprocated by the server with a Server Hello (SH) message. Both of these messages play a role in the KA process and are thus affected by the post-quantum context. Upon finalizing the KA, the handshake continues over a channel encrypted symmetrically. At this stage, the "Certificate" message becomes crucial for the servers authentication by the client, with the certificates potentially bearing PQ signatures, influencing their size. The "Certificate Verify" message that follows carries the handshakes signature, created

with the server certificates private key and altered by PQ considerations. The handshake is concluded by the server with a "Handshake Finished" (HF) message that includes a hash sum of all prior messages to confirm the handshake's integrity. Depending on the Maximum Transmission Unit (MTU), it's possible for messages from SH to HF to be sent in one IP packet, with their efficient amalgamation depending on the specific implementation. Utilizing the certificates received, the client authenticates the server's legitimacy via the PKI. The handshake is finalized when the client dispatches a "Change Cipher Spec" and an HF message, both containing a hash of preceding communications and typically sent in the same IP packet in our observations. An external observer has the capability to analyze the first (CH to SH) and second (SH to Client Finished) phases of the handshake, as they comprise data not encrypted.

2.2.1 Quantum-Safe Integration Points

KEM Algorithm (e.g., Kyber) Usage

1. KeyGen (Key Generation):
    - Timing: Before ClientHello
    - Client generates Kyber key pair (public key and private key)
2. Encap (Encapsulation):
    - Timing: At ServerHello
    - Server uses client's Kyber public key for encapsulation to generate shared key and ciphertext
3. Decap (Decapsulation):
    - Timing: After client receives ServerHello
    - Client uses its own Kyber private key to decapsulate the received ciphertext to obtain the shared key

SIG Algorithm (e.g., Falcon) Usage

1. Signing:
    - Timing: At CertificateVerify

- Server uses Falcon private key to sign all previous handshake messages
- If client authentication is required, the client uses Sign in its CertificateVerify message

2. Verification (Authentication):

- Timing: After client processes server's CertificateVerify
- Client uses Falcon public key from server's certificate to verify server's signature
- If client authentication is required, server verifies client's signature

2.3 Introduction to Open Quantum Safe

The Open Quantum Safe (OQS) project is an open-source project that aims to support the transition to quantum-resistant cryptography. OQS is part of the Linux Foundation's Post-Quantum Cryptography Alliance. Within the OQS ecosystem, there are two significant projects that serve as essential tools in our experiments: liboqs [17] and oqs-provider [18] . The liboqs is a C library that provides implementations of quantum-resistant cryptographic algorithms, including various key encapsulation mechanisms (KEMs) and digital signature algorithms. It features highly optimized implementations and a clean API designed for easy integration into applications and cryptographic protocols.

The oqs-provider complements liboqs by serving as a provider for OpenSSL 3.0, enabling the integration of post-quantum algorithms into the OpenSSL framework. It allows applications to use quantum-resistant cryptography through OpenSSL's standard interfaces, supporting both pure post-quantum and hybrid schemes in protocols such as TLS 1.3. Together, these components create a comprehensive framework that enables developers and organizations to experiment with and deploy post-quantum cryptographic solutions in real-world applications.

3 Literature Review

3.1 Related Literature

3.1.1 PQ Authentication in TLS 1.3: A Performance Study

The paper [19] was presented by Dimitrios Sikeridis et al. The research into post-quantum cryptography (PQC) within TLS 1.3 offers critical insights into transitioning to quantum-resistant algorithms.

This study evaluates round 2 submissions NISTs post-quantum cryptographic signature algorithm candidates, focusing on their TLS 1.3 integration, effects on connection latency, and server performance in realistic network conditions. Figure 2 shows the features and parameter selection of the post-quantum algorithm used in this paper. It investigates the trade-offs between post-quantum signatures' size and the computational load of PQC operations.

A key outcome is that two post-quantum signature algorithms, Dilithium and Falcon, show feasible performance with minimal overhead, indicating that selecting appropriate post-quantum algorithms could enable a seamless transition to quantum-resistant security in TLS protocols without significantly impacting performance.

Additionally, the study addresses the challenges of integrating post-quantum authentication in TLS 1.3, suggesting solutions like combining different post-quantum signature algorithms in the same certificate chain for optimized performance.

Signature Algorithm	Hard Problem	Public Key Size (Bytes)	Private Key Size (Bytes)	Signature Size (Bytes)	Claimed Classical Security Level	Claimed PQ Security Level
RSA 3072	Integer Factorization	387	384	384	128 bits	∼0 bits
ECDSA 384	EC Discrete Logarithm	48	48	48	192 bits	∼0 bits
Dilithium II	Module Learning with Errors	1184	2800	2044	100 bits	91 bits
Falcon 512	NTRU	897	1281	690	114 bits	103 bits
MQDSS 48	Multivariate	46	13	20854	160 bits	99 bits
Picnic $L1FS$	Zero-Knowledge Proofs	33	49	34036	128 bits	64 bits
SPHINCS$^+$ SHA256-128f-simple	Hash-Based	32	64	16976	128 bits	64 bits
Rainbow Ia - $Cyclic$	Multivariate	58144	92960	64	143 bits	106 bits
Dilithium IV	Module Learning with Errors	1760	3856	3366	174 bits	158 bits
Falcon 1024	NTRU	1793	2305	1330	263 bits	230 bits

Figure 2: Algorithms and Parameter Sets used in this study.

3.1.2 The Performance of Post-Quantum TLS 1.3

The paper [20] was presented by Markus Sosnowski et al. This study explores how different signature algorithms and key agreements impact TLS 1.3's handshake latencies and computational costs.

The researchers use both black-box and white-box measurements to assess handshake latencies and computational costs across a variety of scenarios, including constrained environments and potential attack scenarios. Additionally, the author conducted grouped comparisons for algorithms at each security level.

Key findings include:

- Some post-quantum algorithms, like HQC and Kyber, perform comparably to or even better than current cryptographic standards, with no significant performance drawbacks even when using hybrid algorithms.
- Algorithms such as Dilithium and Falcon offer improved performance over traditional algorithms like RSA, suggesting that post-quantum TLS can be suitably adopted in today's systems.
- The study also examines the independence of Key Agreements (KA) and Signature Algorithms (SA), revealing that certain combinations can influence overall performance due to factors like message buffering in TLS implementations.
- In constrained environments (emulated with loss, bandwidth limitations, and delay), the size and computational requirements of post-quantum cryptographic algorithms can significantly affect performance.

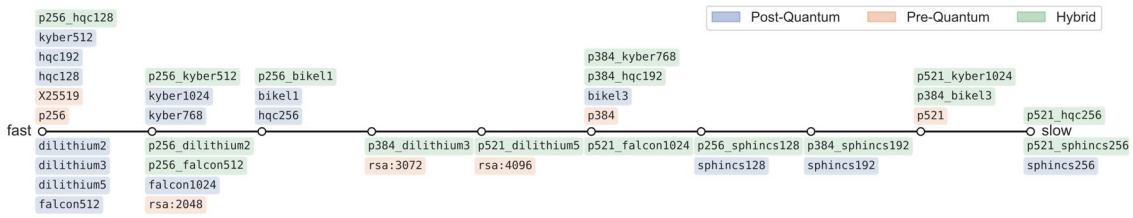


Figure 3: HANDSHAKE LATENCY: On the left being the fastest.

3.1.3 Security Comparisons and Performance Analyses

This paper [21] was presented by Manohar Raavi et al. The study focuses on the finalist algorithms Dilithium, Falcon, Rainbow and GeMSS offering a comprehensive analysis of their security strengths and computational performance, especially when integrated into the Transport Layer Security (TLS) protocol and Transmission Control Protocol/Internet Protocol (TCP/IP).

The authors navigate through the complexities of comparing these algorithms by employing a unified measure of security strength analysis, the depth-width cost for quantum circuits (DW cost), which accommodates the different computational hardness assumptions underlying each algorithm (lattice-based for Dilithium and Falcon, and multivariate polynomial-based for Rainbow). This novel approach enables a direct comparison across different cryptographic families, enhancing the understanding of each algorithm's resilience against potential quantum attacks.

Raavi et al. find that, in terms of security strength measured through DW cost, Rainbow presents a higher computational challenge for potential quantum attackers compared to Dilithium and Falcon. However, in performance evaluations, Dilithium exhibits superior efficiency, with the fastest execution times for key generation, signing, and verification processes across varying message lengths. The study also reveals that while Dilithium and Falcon are viable for integration with current TLS protocols, Rainbow's large key sizes pose significant challenges.

3.1.4 Benchmarking Post-Quantum Cryptography in TLS

The paper [22] was presented by Christian Paquin et al. The paper's objective is to assess how the slower computation or larger key sizes and signatures of post-quantum algorithms affect TLS's performance under realistic network conditions.

The study is distinguished by its comprehensive methodology, utilizing a network emulation framework that simulates various network conditions. This approach allows the authors to meticulously analyze the performance impact of post-quantum key exchange and authentication mechanisms within TLS 1.3, considering variables such as link latency and packet loss rates. Among the key findings, it was observed that packet loss rates above 3-5% significantly affect the performance of post-quantum algorithms, particularly those that involve larger messages or signatures, as these require fragmentation across multiple packets.

Paquin, Stebila, and Tamvada's work is instrumental in highlighting the practical considerations of deploying post-quantum cryptography in TLS. By showcasing that certain post-quantum algorithms can be integrated into TLS 1.3 with minimal overhead, their research paves the way for the gradual adoption of quantum-resistant cryptographic solutions. Additionally, the paper's insights into the nuanced trade-offs between security, performance, and network conditions contribute valuable guidance for both researchers and practitioners in the field of cryptography and network security.

3.2 Research Gaps

Building on the progressive strides marked by the submissions in the National Institute of Standards and Technologys (NIST) third and fourth rounds of evaluation for post-quantum cryptography algorithms, this study aims to undertake a more comprehensive comparison between TLS 1.3 implementations that exclusively incorporate post-quantum algorithms (both Key Encapsulation Mechanisms (KEMs) and Signature Algorithms) and those that adopt a hybrid approach, blending traditional cryptographic methods with post-quantum algorithms. The necessity for such a comparative analysis stems from the ongoing evolution of quantum computing and its potential to compromise existing cryptographic standards.

While previous research has laid the groundwork by exploring the integration of post-quantum algorithms within TLS protocols, there remains a gap in thoroughly evaluating the performance, and interoperability of purely post-quantum versus hybrid approaches in TLS 1.3. First, the existing research includes incomplete algorithms (Such as FrodoKEM and ML-KEM), and secondly, many previous studies have addressed performance in network packet-loss and bandwidth-constrained environments, but there has been insufficient evaluation of comprehensive, limited network environments. Thirdly, Current research primarily relies on the oqs-openssl framework. Limited evaluation of the newer oqs-provider framework's capabilities. Fourth, Some previous studies have modelled performance-constrained conditions, but the performance statistics are not complete for some normal-performing devices. Fifth, in addition to the overall handshake time, it is also important to have separate measurements for Verification, Signing, Decap, Encap, and KeyGen, for which previous research is not complete.

A comprehensive analysis, as proposed, would not only provide valuable insights into the readiness and efficacy of these cutting-edge algorithms but also offer guid-

ance on the strategic direction for transitioning to quantum-resistant cryptographic standards. This includes assessing the computational overhead and latency implications associated with different algorithmic configurations. Ultimately, such research could play a pivotal role in informing the development of robust, future-proof cryptographic protocols that ensure the continued security of digital communications in the advent of quantum computing.

4 Project Design

4.1 Methodology

4.1.1 Environment Setup

Hardware Configuration

- Processor: Intel Core i7 10700K (x86_64)
- OS: Ubuntu 20.04 LTS (Linux kernel 5.15.0-122-generic)

Cryptographic Libraries

- OpenSSL Default Provider (v3.3.2)
- OpenSSL OQS Provider (v0.7.1-dev)
- liboqs (v0.11)

Software Environment

Python 3.8.10 with libraries:

- Data Processing: NumPy (1.24.4), Pandas (2.0.3)
- Visualization: Matplotlib (3.7.5), Seaborn (0.13.2)
- System Tools: psutil (6.0.0), PyYAML (5.3.1), re (2.2.1)
- Python Built-ins: subprocess, multiprocessing, socket, ssl, threading

4.1.2 Network Configuration

- Network Namespaces:

  - Server namespace (server_ns): IP 10.200.1.1/24
  - Client namespace (client_ns): IP 10.200.1.2/24
  - Connection: Virtual Ethernet (veth) pair
- Traffic Control Parameters:
  - Configurable bandwidth limitation
  - Adjustable packet loss rate
  - Customizable network interface parameters
- Network Configuration Tool:
  - Linux Traffic Control (tc)
  - Hierarchical Token Bucket (htb)
  - Network Address Configuration (ip)

4.1.3 Selection of Parameters and Metrics

Selection of Algorithms

The post-quantum and hybrid algorithms chosen to be used are listed in Tables 4.2 and 4.3, and the traditional algorithms are not listed in the table. In controlling variables, Ed25519 was used as the signature algorithm when testing the KEM algorithm, while X25519 served as the KEM algorithm when evaluating the signature algorithm. This choice reflects the default preferences of TLS 1.3 in most scenarios. According to the description of TLS in the background, the KEM algorithm and the signature algorithm are separate from each other, and the TLS handshake process is executed sequentially, so we will test them separately. To assess algorithmic performance across varying security levels, the study evaluated diverse combinations at NIST security levels 1, 3, and 5, respectively. Although there are five levels of security in NIST, the mainstream classification is based on level 1, level 3, and level 5. ED448 and X448's security is AES224, which is closer to level 5 (Details of the classification are shown in Table 4.1) , so it is categorised as level 5, and the later analysis is based on this assumption. The security of all other algorithms is based on the security of the algorithm statement.

Level	Security Description
I	At least as hard to break as AES128 (exhaustive key search)
II	At least as hard to break as SHA256 (collision search)
III	At least as hard to break as AES192 (exhaustive key search)
IV	At least as hard to break as SHA384 (collision search)
V	At least as hard to break as AES256 (exhaustive key search)

Table 4.1: Security Level Descriptions

Algorithm	Variants
BIKE	bikel1, p256_bikel1, x25519_bikel1, bikel3, p384_bikel3, x448_bikel3, bikel5, p521_bikel5
CRYSTALS-Kyber	kyber512, p256_kyber512, x25519_kyber512, kyber768, p384_kyber768, x448_kyber768, x25519_kyber768, p256_kyber768, kyber1024, p521_kyber1024
FrodoKEM	frodo640aes, p256_frodo640aes, x25519_frodo640aes, frodo640shake, p256_frodo640shake, x25519_frodo640shake, frodo976aes, p384_frodo976aes, x448_frodo976aes, frodo976shake, p384_frodo976shake, x448_frodo976shake, frodo1344aes, p521_frodo1344aes, frodo1344shake, p521_frodo1344shake
HQC	hqc128, p256_hqc128, x25519_hqc128, hqc192, p384_hqc192, x448_hqc192, hqc256, p521_hqc256
ML-KEM	mlkem512, p256_mlkem512, x25519_mlkem512, mlkem768, p384_mlkem768, x448_mlkem768, x25519_mlkem768, p256_mlkem768, mlkem1024, p521_mlkem1024, p384_mlkem1024

Table 4.2: Key Encapsulation Mechanisms (KEM) Algorithms

Selection of Parameters and Metrics

For network parameters, we employ Linux Traffic Control (tc) with Hierarchical Token Bucket (htb) queuing discipline to simulate various network conditions. The configurable parameters include bandwidth limitation and packet loss rate, which are applied bidirectionally on both server (server_ns) and client (client_ns) network interfaces. Through netem queue discipline, we can precisely control these parameters on both client and server interfaces, ensuring realistic network condition simulation

for our experiments.

For performance evaluation, we measure three key metrics: handshake completion time, CPU usage, and memory consumption. Handshake time is measured using Python's time.perf_counter() with millisecond precision. System resource utilization, including CPU usage percentage and memory consumption in kilobytes, is monitored through psutil library. Each test is conducted with 1000 iterations per algorithm combination, recording both successful and failed connection attempts. Failed connections are logged with detailed error information. For each test iteration, we store detailed TLS handshake information including certificate verification status, cipher suites negotiation, and protocol version details. Additionally, we save the complete handshake output, key and certificate generation information (in PEM format).

4.1.4 Performance Metrics and Measurement Methodology

For performance evaluation, we measure three key metrics: handshake completion time, CPU usage, and memory consumption. Handshake time is measured using Python's `time.perf_counter()` with millisecond precision, capturing the total elapsed time from connection initiation to handshake completion or failure. CPU usage is calculated as the percentage of CPU time consumed by the TLS handshake process relative to total available CPU time, normalized per logical CPU core, tracking both user and system CPU time through the formula: $CPU\ Usage = \frac{CPUTime}{ElapsedTime} \times \frac{100}{CPU\ \ Count}$. Memory consumption is measured as the incremental Resident Set Size (RSS) memory usage, where RSS represents the portion of process memory that is held in physical RAM rather than swap space, providing a more accurate representation of actual memory impact. The memory consumption is calculated by monitoring the differential RSS before and after the handshake process: $Memory\ Usage = \frac{Final\ \ RSS - \ Initial\ \ RSS}{1024}$ KB. System resource utilization is monitored through the `psutil` library, providing precise measurements of both CPU and memory metrics. Each test is conducted with 1000 iterations per algorithm combination, recording both successful and failed connection attempts. Success rate is determined by the percentage of successful handshakes, where success is defined by a verification return code of 0 and complete handshake confirmation. Failed connections are comprehensively logged with detailed error information. For each test iteration, we store detailed TLS handshake information including certificate verification status, cipher suites negotiation, and protocol version details. Additionally, we save the complete handshake output, key and certificate generation information

(in PEM format), ensuring thorough documentation of each test case for subsequent analysis.

Algorithm	Variants
CRYSTALS-Dilithium	dilithium2, p256_dilithium2, rsa3072_dilithium2, dilithium3, p384_dilithium3, dilithium5, p521_dilithium5
ML-DSA	mldsa44, p256_mldsa44, rsa3072_mldsa44, mldsa44_pss2048, mldsa44_rsa2048, mldsa44_ed25519, mldsa44_p256, mldsa44_bp256, mldsa65, p384_mldsa65, mldsa65_pss3072, mldsa65_rsa3072, mldsa65_p256, mldsa65_bp256, mldsa65_ed25519, mldsa87, p521_mldsa87, mldsa87_p384, mldsa87_bp384, mldsa87_ed448
FALCON	falcon512, p256_falcon512, rsa3072_falcon512, falconpadded512, p256_falconpadded512, rsa3072_falconpadded512, falcon1024, p521_falcon1024, falconpadded1024, p521_falconpadded1024
SPHINCS+	sphincssha2128fsimple, p256_sphincssha2128fsimple, rsa3072_sphincssha2128fsimple, sphincssha2128ssimple, p256_sphincssha2128ssimple, rsa3072_sphincssha2128ssimple, sphincssha2192fsimple, p384_sphincssha2192fsimple, sphincssha2192ssimple, p384_sphincssha2192ssimple, sphincssha2256fsimple, p521_sphincssha2256fsimple, sphincssha2256ssimple, p521_sphincssha2256ssimple, sphincsshake128fsimple, p256_sphincsshake128fsimple, rsa3072_sphincsshake128fsimple, sphincsshake128ssimple, p256_sphincsshake128ssimple, rsa3072_sphincsshake128ssimple, sphincsshake192fsimple, p384_sphincsshake192fsimple, sphincsshake192ssimple, p384_sphincsshake192ssimple, sphincsshake256fsimple, p521_sphincsshake256fsimple, sphincsshake256ssimple, p521_sphincsshake256ssimple
MAYO	mayo1, p256_mayo1, mayo2, p256_mayo2, mayo3, p384_mayo3, mayo5, p521_mayo5
CROSS	CROSSrsdp128balanced, CROSSrsdp128fast, CROSSrsdp128small, CROSSrsdp192balanced, CROSSrsdp192fast, CROSSrsdp192small, CROSSrsdp256small, CROSSrsdpg128balanced, CROSSrsdpg128fast, CROSSrsdpg128small, CROSSrsdpg192balanced, CROSSrsdpg192fast, CROSSrsdpg192small, CROSSrsdpg256balanced, CROSSrsdpg256fast, CROSSrsdpg256small

Table 4.3: Signature Algorithms and their Variants

4.2 Implementation Details

4.2.1 Framework Architecture

The framework is designed with a modular architecture to evaluate post-quantum cryptography (PQC) and hybrid key exchange mechanisms. At its core, the framework consists of a benchmark controller (benchmark.py) that orchestrates the entire testing process. This controller interfaces with three primary modules: (1) a client module (client.py) responsible for initiating TLS connections, (2) a server module (server.py) handling connection requests, and (3) a utility module (utility.py) providing essential functions for configuration, network setup, and result analysis.

The system utilizes Linux network namespaces (server_ns and client_ns) connected via virtual Ethernet pairs to create isolated testing environments. Network conditions such as packet loss and bandwidth limitations are simulated using traffic control (tc) mechanisms. Configuration is managed through YAML files, specifying test parameters including cryptographic algorithms, network conditions, and test iterations. The framework collects comprehensive metrics including handshake times, CPU usage, and memory consumption, storing results in a structured format for subsequent analysis and visualization.

4.2.2 Experimental Scenarios

First, we conduct step-by-step testing with each step lasting 10 seconds to measure the operations per second (ops/s) for each algorithm. This is followed by complete handshake testing procedures.

For the complete handshake testing, we set the default iteration count to 1,000 rounds, which can be adjusted based on specific experimental requirements. The initial evaluation begins with testing KEM and Signature algorithms without additional constraints, using all algorithms listed in the previous section.

To evaluate network packet loss impact, we conduct tests with packet loss rates of 1%, 3%, 5%, and 10%, while maintaining a consistent bandwidth of 100 mbits/s to control variables.

For bandwidth limitation testing, we implement constraints of 20 kbits/s, 60 kbits/s, 100 kbits/s, 500 kbits/s, and 1 Mbits/s to assess performance under various

bandwidth conditions.

To analyze combined network effects, we examine performance at 500 kbits/s bandwidth with packet loss rates of 3% and 5%. For security level analysis, algorithms are categorized based on their claimed security levels, without conducting additional security testing beyond the algorithms' stated security assertions.

5 Result and Analysis

5.1 Stepwise Handshake Evaluation

5.1.1 KEM Stepwise Evaluation

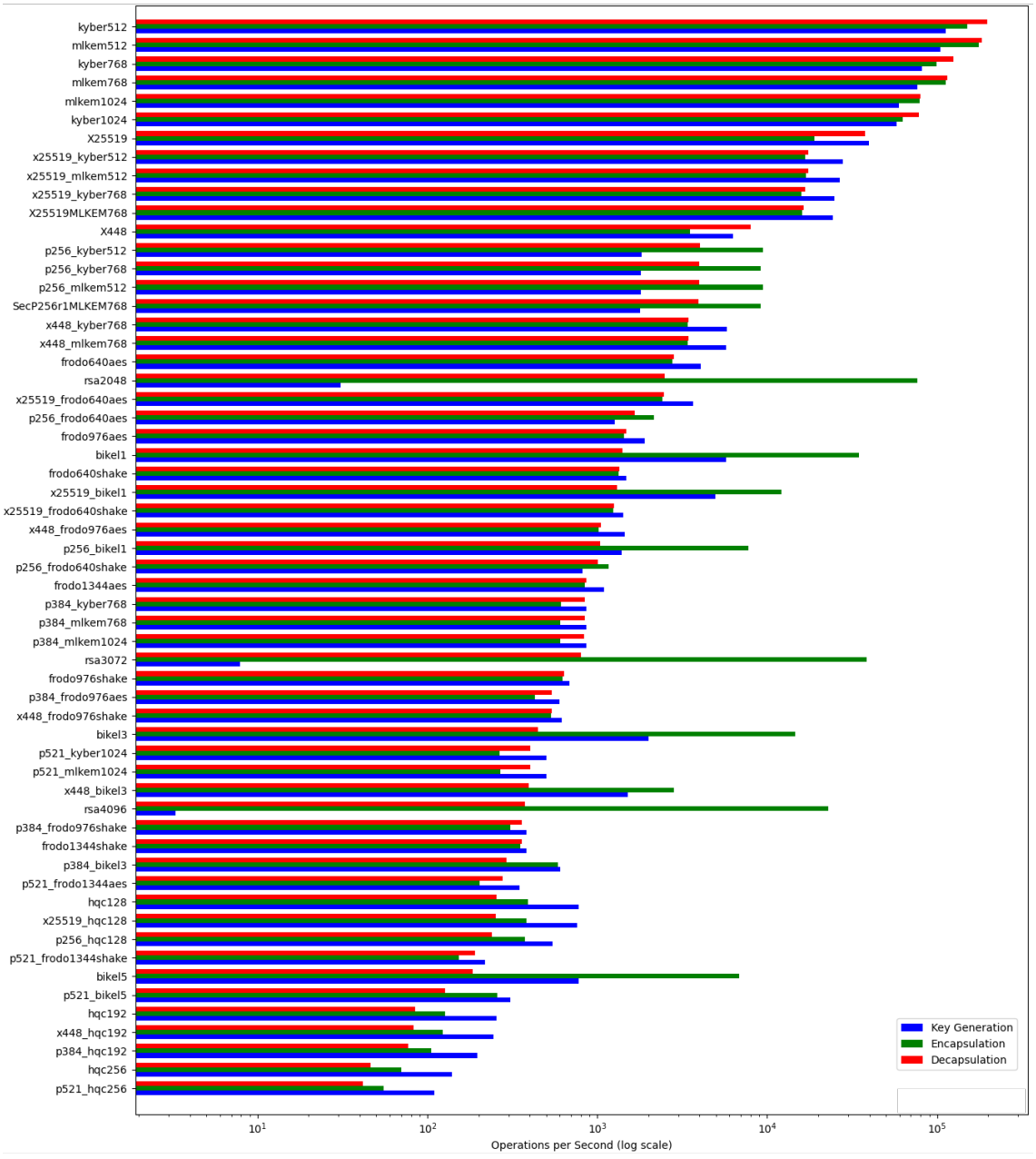


Figure 4: KEM Algorithm Step-by-Step Performance

The performance analysis of traditional algorithms reveals that RSA exhibits significant performance variations across different key sizes. RSA-2048 demonstrates baseline performance with key generation taking 32.57ms, encapsulation 0.013ms, and decapsulation 0.400ms. As the key size increases to 3072 and 4096 bits, we observe approximately quadratic growth in key generation time (125.98ms and 304.24ms respectively) and linear growth in decapsulation time (1.251ms and 2.669ms respectively). However, encapsulation remains remarkably efficient across all key sizes, ranging from 13 to 44 microseconds. In the elliptic curve domain, X25519 showcases exceptional efficiency with very fast operations (keygen: 25s, encaps: 53s, decaps: 27s) and high throughput (39,621 keygens/s, 18,929 encaps/s, 37,613 decaps/s). X448, while slower due to its higher security level, still maintains impressive performance with operation times of 159s, 284s, and 125s for key generation, encapsulation, and decapsulation respectively.

In the post-quantum landscape, Kyber and its standardized version ML-KEM demonstrate outstanding performance across all security levels. At NIST Level 1, both Kyber512 and MLKEM512 achieve remarkable efficiency with operation times under 10 microseconds and throughput reaching 150,000-175,000 operations per second. The performance gracefully degrades as security levels increase, with Level 3 (Kyber768/MLKEM768) maintaining operations under 13 microseconds and Level 5 (Kyber1024/MLKEM1024) still keeping operations under 17 microseconds. FrodoKEM shows interesting variations between its AES and SHAKE variants, with AES variants consistently outperforming SHAKE variants by 40-50%. The base Frodo640AES achieves operation times of 246-361 microseconds, while higher security variants show expected performance decline, with Frodo1344 being 3-4 times slower than Frodo640.

BIKE and HQC represent different approaches in code-based cryptography, with distinctly different performance characteristics. BIKE demonstrates an asymmetric performance profile, particularly evident in BIKE-L1 with very fast encapsulation (29s) but significantly slower decapsulation (714s). This pattern becomes more pronounced at higher security levels, with BIKE-L5 showing encapsulation times of 146s and decapsulation times of 5.41ms. HQC exhibits higher computational overhead across all operations, with HQC-128 requiring milliseconds for each operation (keygen: 1.28ms, encaps: 2.57ms, decaps: 3.92ms) and showing substantial performance degradation at higher security levels, with HQC-256 being approximately 5.5 times slower than HQC-128.

The analysis of hybrid schemes reveals complex performance interactions between classical and post-quantum components. X25519 based combinations consistently demonstrate superior performance across all post-quantum algorithm families. For instance, X25519 combined with Kyber/ML-KEM shows minimal overhead, adding only about 50 microseconds per operation while maintaining practical efficiency of 15,000-17,000 operations per second. The performance impact varies significantly when combining with different post-quantum algorithms X25519 with BIKE shows moderate overhead (keygen: 203s, encaps: 82s), while combinations with HQC demonstrate more substantial performance costs (keygen: 1.31ms, encaps: 2.63ms).

NIST curve combinations present a different performance profile, with P-256 combinations showing moderate performance (around 9,000 encaps/s) but significant overhead in key generation (approximately 550s). The performance impact becomes more pronounced with higher security NIST curves P-384 combinations typically perform about 3 times slower than P-256, while P-521 combinations show approximately 7 times slower performance. This pattern holds across all post-quantum algorithm families, though the relative impact varies. FrodoKEM hybrid schemes with NIST curves show particularly interesting behavior, with P-256_Frodo640AES achieving reasonable performance (keygen: 792s, encaps: 463s) while maintaining a balanced security profile.

Among hybrid combinations with code-based systems, BIKE and HQC show distinct patterns. BIKE's hybrid variants maintain its characteristic asymmetric performance profile, with P-256_BIKE-L1 showing fast encapsulation (129s) but slower decapsulation (964s). HQC hybrid schemes demonstrate additive overhead from both components, with P-384_HQC-192 showing the cumulative impact of both higher security parameters (keygen: 5.07ms, encaps: 9.51ms, decaps: 13.03ms). The performance cost of hybrid schemes appears to be more pronounced in code-based systems compared to lattice-based ones, particularly at higher security levels.

5.1.2 SIG Stepwise Evaluation

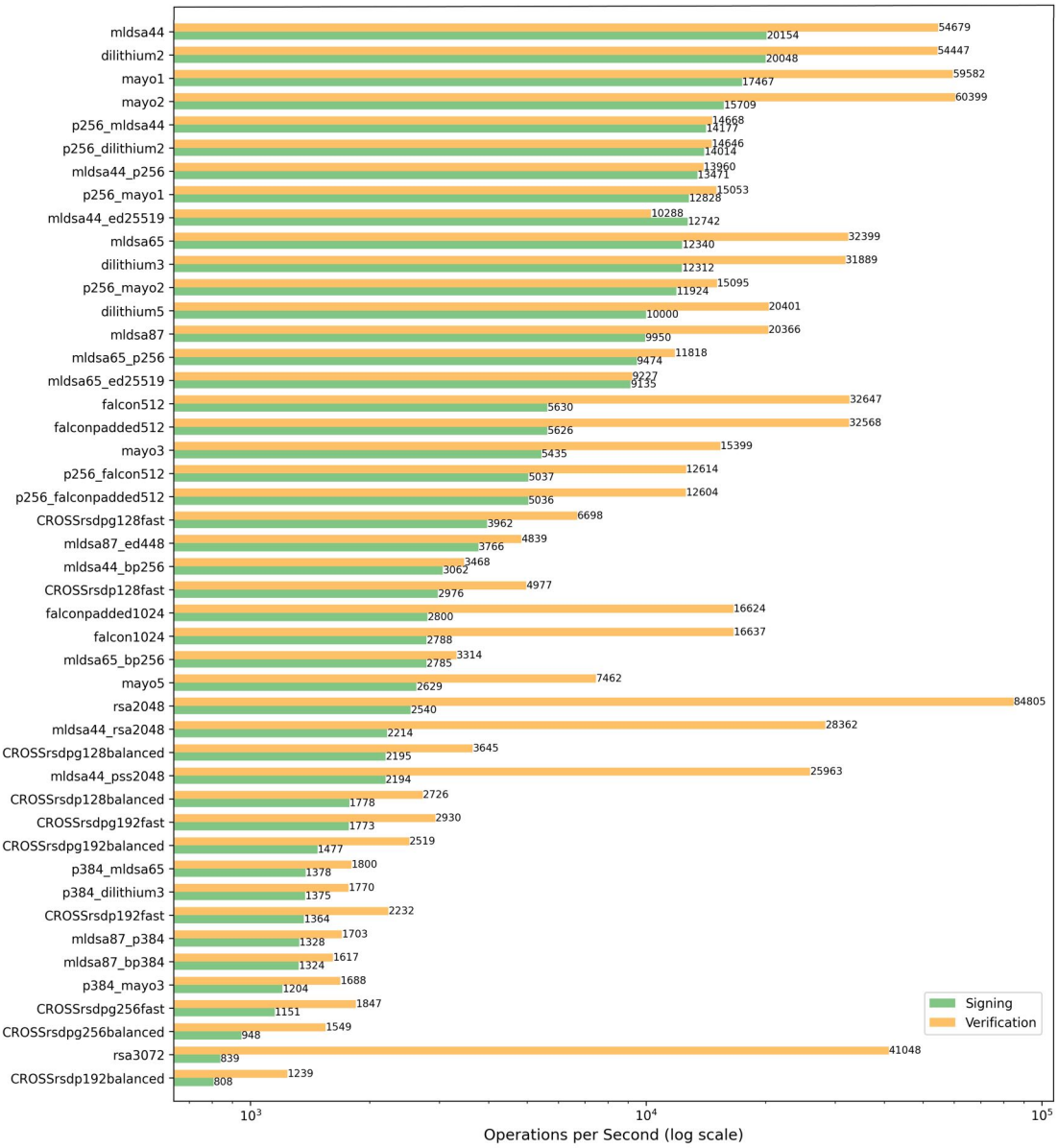


Figure 5: Signature Algorithm Step-by-Step Performance Part 1

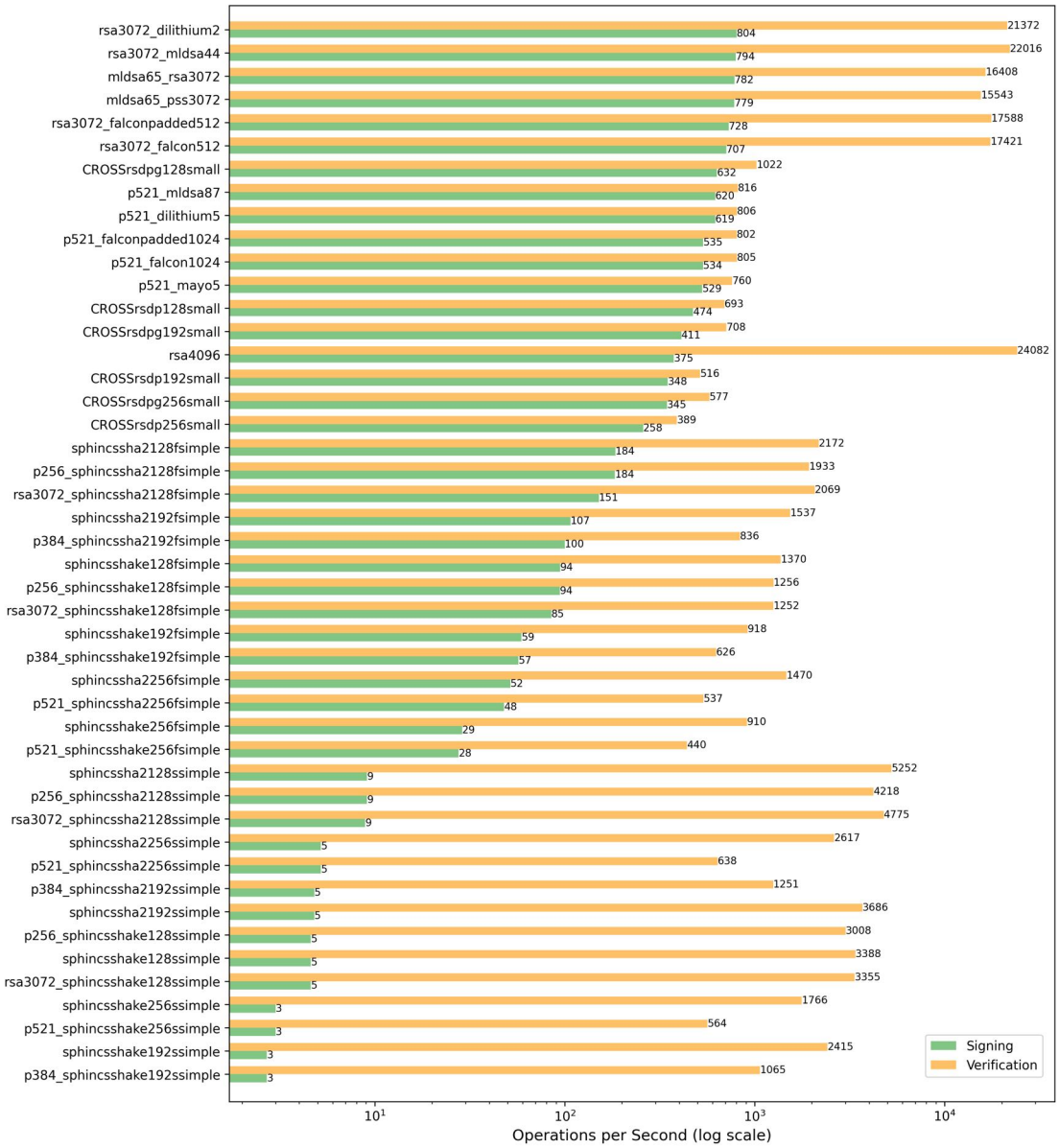


Figure 6: Signature Algorithm Step-by-Step Performance Part 2

In the traditional signature scheme domain, RSA demonstrates predictable performance characteristics across different security levels. RSA-2048 achieves signing speeds of 2,540 operations per second and impressive verification throughput of 84,805 operations per second. As the key size increases to 3072 and 4096 bits, we observe a clear performance trade-off: RSA-3072 shows signing and verification speeds of 838 and 41,047 operations per second respectively, while RSA-4096 further

decreases to 375 signs/s and 24,082 verify/s. This pattern reflects RSA's characteristic asymmetric performance profile, where verification remains significantly faster than signing across all security levels.

The lattice-based signature schemes Dilithium and ML-DSA (standardized version of Dilithium) demonstrate exceptional performance at their base security levels. Dilithium2/MLDSA44 achieves approximately 20,000 signing operations per second and over 54,000 verifications per second, showcasing efficient performance suitable for high-throughput applications. The performance gracefully degrades with higher security levels - Dilithium3/MLDSA65 maintains about 12,300 signs/s and 32,000 verify/s, while Dilithium5/MLDSA87 still provides robust performance with roughly 10,000 signs/s and 20,000 verify/s.

Falcon, another lattice-based signature scheme, presents a different performance profile. Falcon-512 demonstrates asymmetric performance characteristics with moderate signing speed (5,630 operations/s) but excellent verification performance (32,646 operations/s). Falcon-1024 shows expected performance reduction at higher security levels, achieving 2,787 signs/s and 16,636 verify/s. The padded variants of Falcon maintain nearly identical performance to their standard counterparts, suggesting minimal overhead from the padding mechanism.

SPHINCS+, representing hash-based signatures, shows distinctive performance characteristics varying significantly between its parameter sets. The "f" (fast) variants demonstrate better performance than their "s" (small) counterparts, but at the cost of larger signatures. SPHINCS+-SHA2-128f-simple achieves 184 signs/s and 2,171 verify/s, while its small variant drops to 9 signs/s but maintains reasonable verification speed at 5,252 operations/s. Higher security levels show expected performance degradation, with SPHINCS+-SHA2-256f-simple performing at 51 signs/s and 1,469 verify/s. The SHAKE variants consistently show slightly lower performance compared to their SHA2 counterparts.

In the hybrid scheme landscape, we observe various performance patterns depending on the classical component used. X25519 and P-256 combinations with lattice-based signatures maintain reasonable efficiency P256_Dilithium2 achieves 14,013 signs/s and 14,646 verify/s, showing minimal overhead compared to standalone operations. However, combinations with higher security NIST curves demonstrate more substantial performance impact P384_Dilithium3 drops to 1,375 signs/s and 1,770 verify/s, while P521_Dilithium5 further reduces to 618 signs/s and 806 verify/s.

The MAYO signature scheme presents competitive performance at its base se-

curity level, with MAYO-1 achieving 17,467 signs/s and 59,582 verify/s. Higher security versions show gradual performance reduction, with MAYO-5 operating at 2,628 signs/s and 7,462 verify/s. Its hybrid variants with NIST curves follow similar patterns to other hybrid schemes, with P256 combinations maintaining reasonable efficiency while P384 and P521 combinations show more significant performance impact.

The CROSS-RSD schemes introduce an interesting performance profile with their different variants (balanced, fast, and small). The “fast” variants predictably outperform their “balanced” and ”small” counterparts in terms of operation speed. CROSS-RSDP-128-fast achieves 2,976 signs/s and 4,977 verify/s, while its small variant operates at 473 signs/s and 692 verify/s. The “g” variants (CROSS-RSDPG) generally show better performance than their standard counterparts, with CROSS-RSDPG-128-fast reaching 3,961 signs/s and 6,698 verify/s. This performance advantage is maintained across all security levels, though with expected degradation at higher levels.

5.2 Full Handshake Evaluation

5.2.1 KEM Full Handshake Evaluation

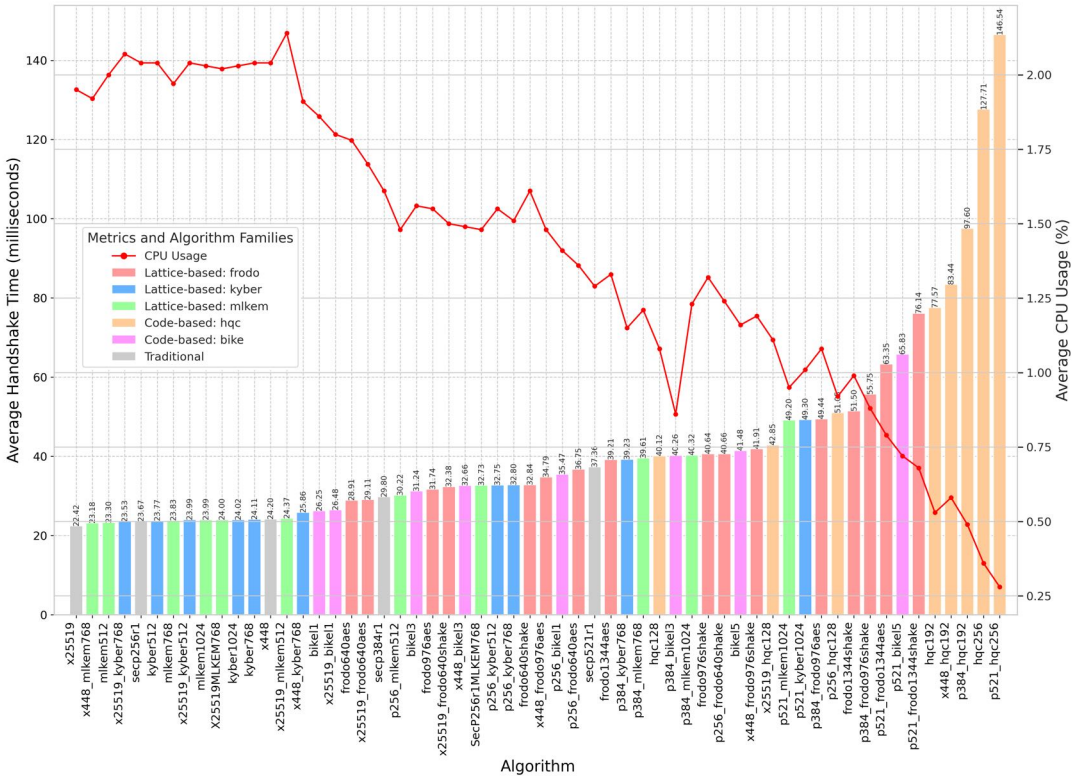


Figure 7: KEM Algorithm Performance

We can see roughly the same results in terms of complete handshake times as in the step-by-step test. The traditional elliptic curve cryptography, represented by x25519, achieves the fastest average handshake time of 22.417ms, serving as our baseline for comparison. Among post-quantum algorithms, MLKEM (the NIST standardized version of Kyber) demonstrates exceptional efficiency, with mlkem768 achieving a handshake time of 23.831ms, nearly matching traditional elliptic curve performance. As the security level increases the handshake time of the kyber series barely increases, which is very rare.

The BIKE family shows interesting performance characteristics across its security levels. The baseline bikel1 implementation achieves a handshake time of 26.248ms,

while bikel3 and bikel5 show increased latency at 31.240ms and 41.480ms respectively. This predictable scaling with security levels makes BIKE a reliable option for systems with varying security requirements. Compared to other post-quantum families like HQC (with times exceeding 120ms), BIKE maintains relatively competitive performance.

The analysis of hybrid schemes reveals nuanced performance impacts when combining traditional and post-quantum algorithms. The performance overhead of adding post-quantum security varies significantly depending on the chosen combination. For instance, x25519_mlkem768 achieves a handshake time of 23.999ms, adding only 1.582ms (7.1%) overhead compared to standalone x25519. Similarly, x25519_bikel1 completes handshakes in 26.476ms, representing a 4.059ms (18.1%) increase. These minimal overheads suggest that hybrid implementations can provide quantum resistance without significant performance penalties.

However, the impact of hybridization becomes more pronounced with higher security levels and certain algorithm combinations. The p384_mlkem768 implementation requires 39.609ms, while p384_bikel3 takes 40.257ms, showing how the choice of elliptic curve significantly influences the hybrid scheme's performance. This pattern continues with p521-based hybrids, where p521_bikel5 requires 65.831ms, demonstrating that the performance cost of hybridization increases more substantially at higher security levels.

The performance landscape shows a distinct stratification when examining different post-quantum approaches. Structured lattice-based systems (Kyber, MLKEM) consistently outperform other PQC approaches. Other families such as the code-based HQC show significantly higher latency. The hqc256 variant, for instance, requires 127.705ms for handshake completion, highlighting the current performance limitations of certain post-quantum approaches. Similarly, FRODO variants, which take a more conservative approach to lattice-based cryptography, show moderate performance with handshake times ranging from 28.910ms (frodo640aes) to 51.498ms (frodo1344shake), reflecting a clear trade-off between conservative security margins and performance.

The reliability aspect shows consistent performance across all implementations, with 100% successful handshakes and stable standard deviations. This reliability extends to hybrid schemes, indicating that the combination of algorithms does not introduce additional stability concerns.

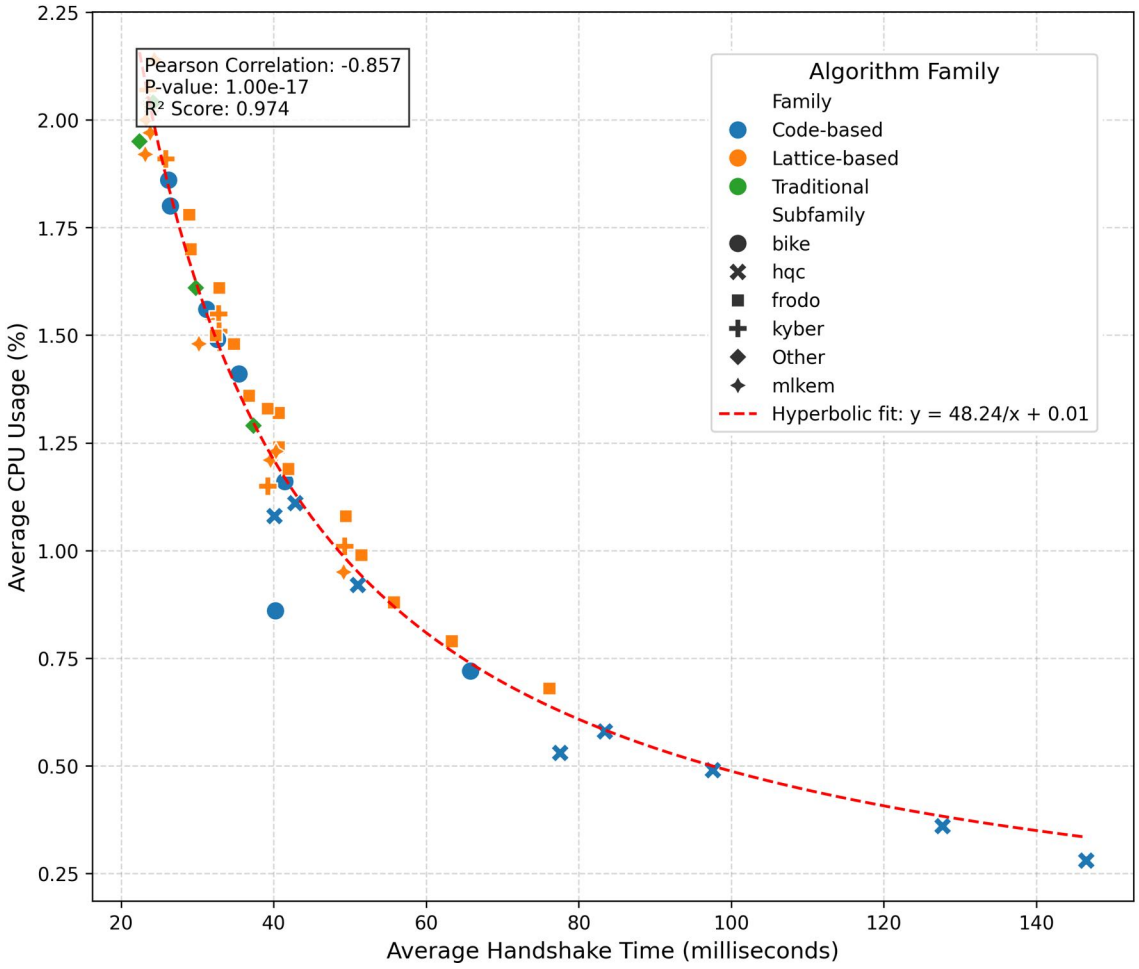


Figure 8: KEM Algorithm Performance

The relationship between handshake time and CPU utilization exhibits a strong negative correlation (r = -0.8566, p = 1.0013e-17), following a hyperbolic pattern described by CPU_usage = 48.2411/handshake_time + 0.0053. The model demonstrates exceptional fit with an R^2 value of 0.9737, explaining nearly 98% of the observed variance. This hyperbolic relationship reveals a fundamental performance trade-off: algorithms achieving shorter handshake times typically demand higher CPU utilization per core, while those with longer handshake durations operate with lower CPU intensity. For example, traditional algorithms like x25519 complete handshakes in 0.022417 seconds with 1.95% CPU usage, while quantum-safe alternatives like hqc256_ed25519 require 0.127705 seconds with only 0.36% utilization. This pattern suggests that faster algorithms achieve their performance through more intensive but shorter duration CPU usage, rather than through reduced overall computational requirements.

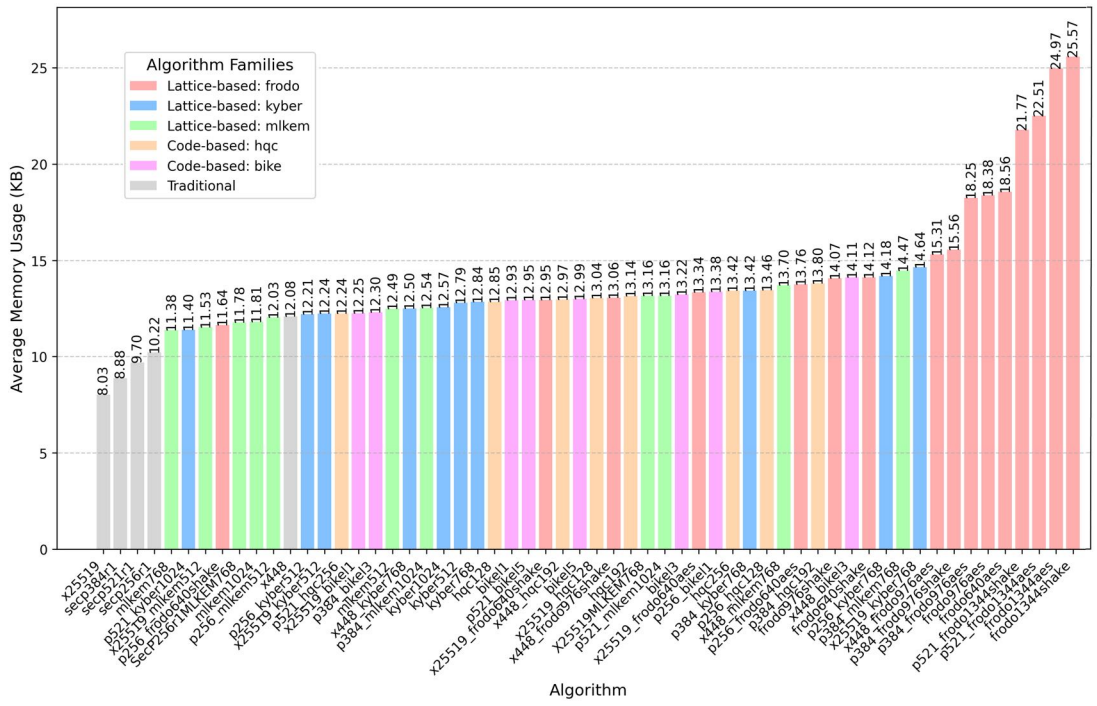


Figure 9: KEM Algorithm Memory Performance

The analysis of memory consumption across different KEM algorithm families reveals several significant patterns. The Frodo family exhibits notably higher memory utilization, with frodo1344shake reaching a peak consumption of approximately 25.57 KB, which can be attributed to its lattice-based cryptographic structure requiring larger matrix operations. However, this peak memory usage remains relatively modest in contemporary computing environments. Traditional algorithms, specifically x25519 and the secp series, demonstrate exceptional efficiency with memory consumption below 10 KB, reflecting the benefits of extensive optimization over time. The post-quantum candidates, including Kyber, MLKEM, and BIKE implementations, maintain moderate memory footprints ranging from 10 to 15 KB, effectively balancing security requirements with resource efficiency. A consistent pattern emerges wherein memory utilization incrementally increases with higher security levels within the same algorithm family, as evidenced by the progression from kyber512 to kyber1024. Notably, hybrid implementations combining traditional and post-quantum algorithms demonstrate only marginal increases in memory overhead compared to their standalone counterparts, suggesting efficient integration characteristics. Based on these observations, while memory consumption varies across algorithms, it does not present a significant constraint in KEM algorithm selection for most practical applications, even in resource-constrained environments.

The findings suggest that future research in post-quantum cryptography implementations may focus more on optimizing computational efficiency and handshake times, as memory consumption appears to be well within acceptable bounds across all evaluated algorithms.

5.2.2 SIG Full Handshake Evaluation

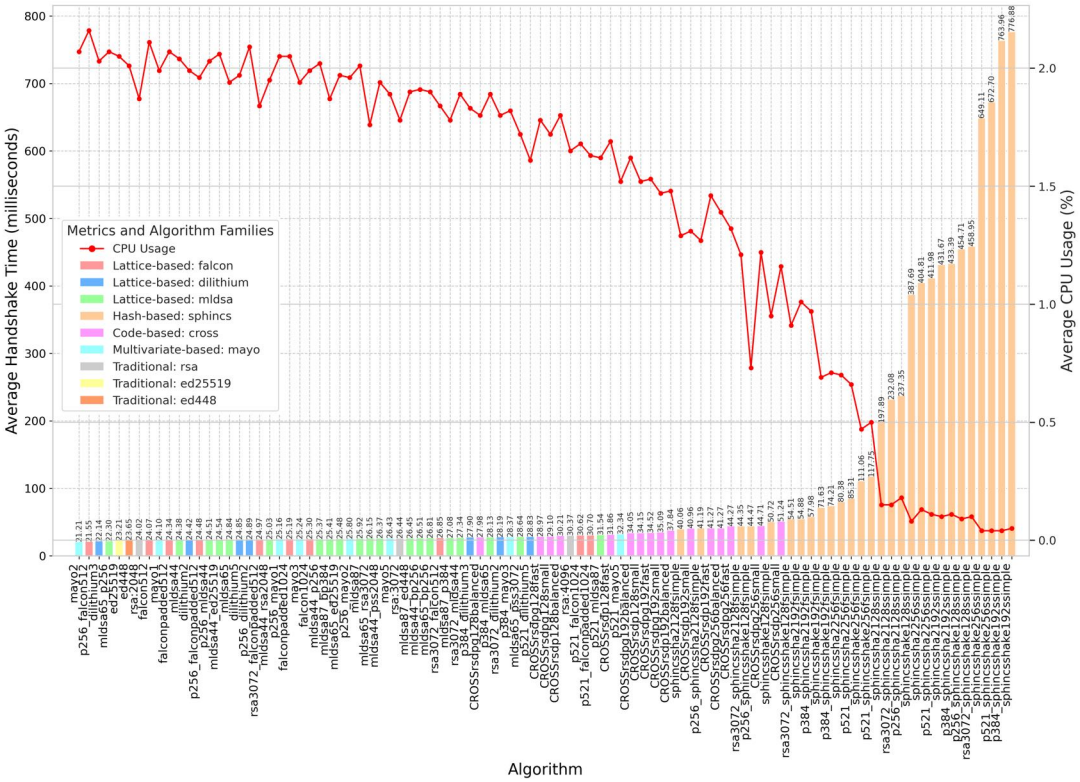


Figure 10: Signature Algorithm Performance

Traditional signature schemes set important baseline performances. RSA maintains competitive performance across different key sizes, with rsa2048 achieving 24.016ms and rsa3072 showing a modest increase to 26.437ms, demonstrating its continued viability. RSA does not have as significant a performance degradation as shown in the step-by-step experiments. The EdDSA variants also perform excellently, with ed25519 completing handshakes in 23.207ms and ed448 in 23.650ms.

Among post-quantum signature schemes, lattice-based solutions demonstrate impressive performance. The Falcon family exhibits exceptional efficiency, with minimal overhead compared to traditional schemes across its security levels. Dilithium

shows consistent performance across its variants, with higher security levels introducing only modest latency increases. Its NIST variant MLDSA follows a similar pattern while maintaining comparable performance characteristics, though with slight implementation differences. Both families demonstrate that lattice-based signatures can achieve performance comparable to traditional schemes while providing quantum resistance.

The Mayo family emerges as a particularly interesting candidate, showing remarkable performance characteristics. Mayo2 achieves an impressive 21.212ms handshake time, outperforming even some traditional schemes. This excellent performance continues across its variants, with mayo1 at 24.100ms and mayo3 at 25.240ms, demonstrating that newer post-quantum approaches can potentially offer superior performance while maintaining security assurances.

The hybrid signature implementation reveals interesting performance patterns. p256_falcon512 combination achieves 21.550ms and maintains the same performance compared to the non-hybrid version. Traditional RSA hybrids maintain reasonable performance, with rsa3072_falcon512 at 26.852ms and rsa3072_dilithium2 at 28.185ms. These results suggest that hybrid schemes can provide quantum resistance with manageable overhead.

Although hash-based signatures exhibit different performance depending on the choice of parameters, SPHINCS+ consumes much more time than the other algorithms. The SPHINCS+ variants show clear distinction between their fast (“f ”) and small (“s”) parameter sets. The fast variants prioritize speed: sphincssha2128fsimple requires 40.057ms, while the small variants prioritize signature size at the cost of speed: sphincssha2128ssimple takes 197.890ms. The pattern continues at higher security levels, with SHAKE variants generally showing higher latency than their SHA2 counterparts.

Looking at alternative post-quantum signatures, the CROSS family offers different performance trade-offs ranging from approximately 28ms to 45ms depending on parameter sets. Like SPHINCS+, CROSS variants also offer fast and small parameter choices, providing flexibility for different use cases.

Reliability analysis shows complete stability across all implementations, with 100% successful handshakes throughout testing. Standard deviations remain particularly low for lattice-based solutions and Mayo variants, indicating stable performance suitable for production environments. While hash-based signatures show higher timing variability, particularly in their small variants.

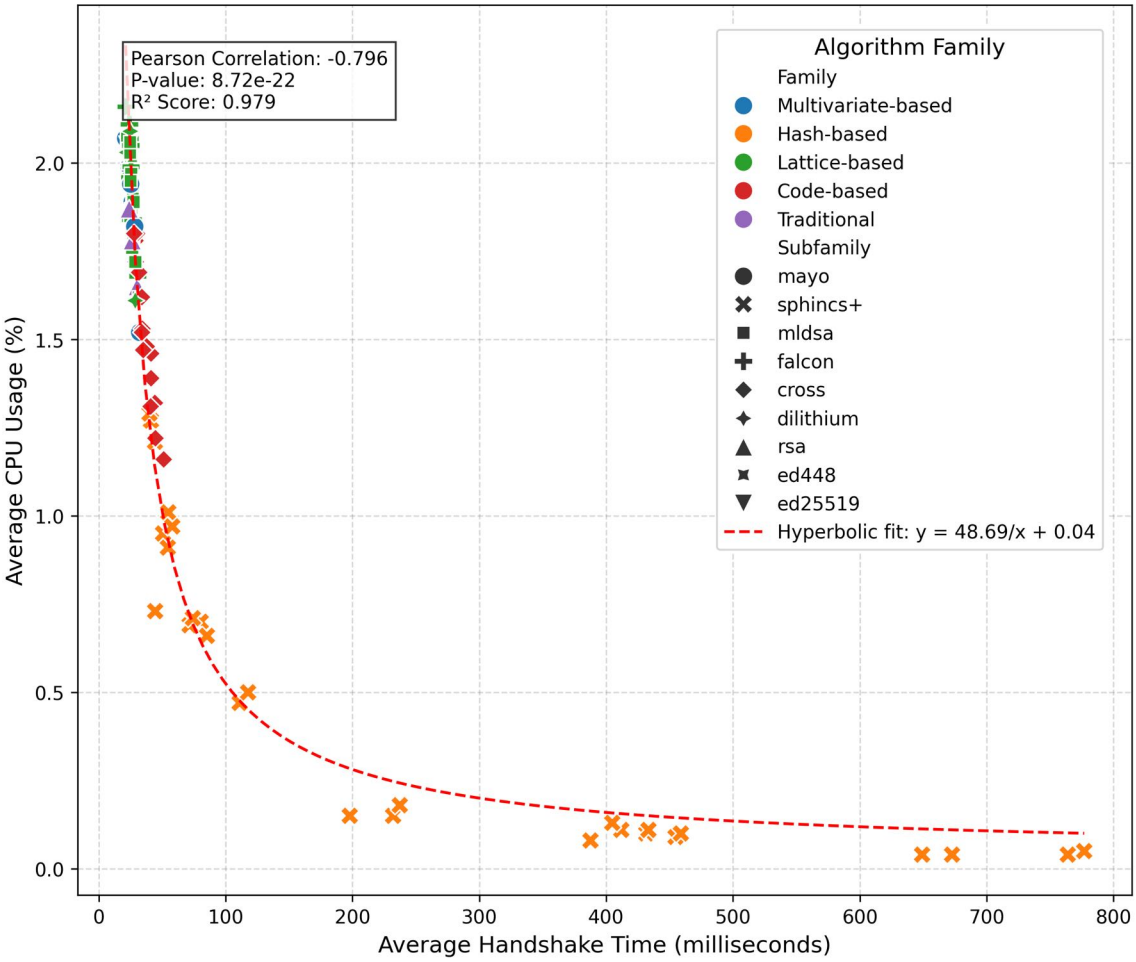


Figure 11: Signature Algorithm Performance

The relationship between handshake time and CPU utilization for signature algorithms demonstrates a strong negative correlation (r = -0.7962, p = 8.7224e-22), following a hyperbolic pattern described by CPU_usage = 48.6902/handshake_time + 0.0379. The model shows excellent fit with an R value of 0.9790, indicating that 97.90% of the variance is explained. This hyperbolic relationship is particularly evident when comparing traditional algorithms like X25519_ed25519 (0.0232s, 2.05% CPU) with post-quantum alternatives such as X25519_sphincsshake192ssimple (0.7769s, 0.05% CPU). The remarkably high R^2 value suggests that this performance trade-off between handshake time and CPU utilization is a fundamental characteristic across different signature algorithm families, rather than being specific to particular implementations.

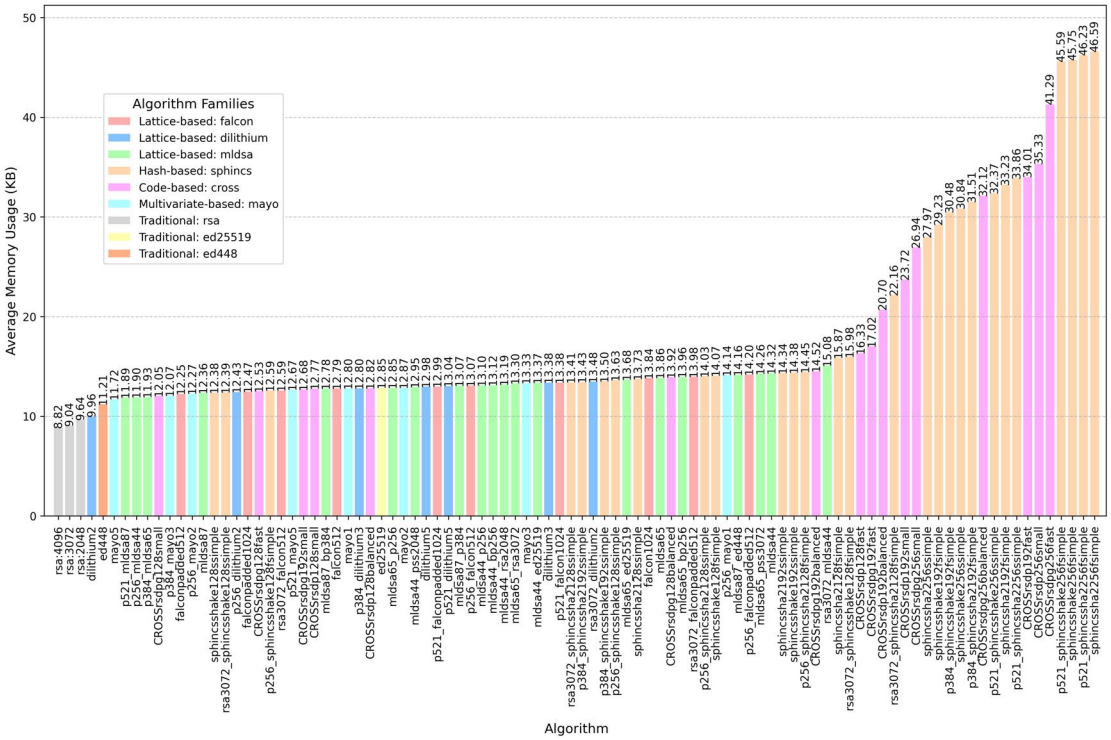


Figure 12: Signature Algorithm Memory Performance

Memory consumption patterns across signature algorithms reveal distinct characteristics that merit attention. The SPHINCS+ family demonstrates notably higher memory requirements, with the fast variants (fsimple) generally consuming more memory than their small counterparts (ssimple). For instance, sphincssha2256fsimple reaches approximately 46.59 KB, while sphincssha2256ssimple uses around 27.97 KB. This pattern is consistent across different parameter sets and hash function choices (SHA2 vs SHAKE), reflecting the inherent trade-off between performance and memory efficiency in hash-based signatures. The CROSS signature family exhibits interesting memory utilization patterns across its variants, with fast versions requiring higher memory (30-41 KB) compared to their balanced (20-32 KB) or small (23-27 KB) counterparts, demonstrating clear trade-offs between performance optimization and memory efficiency. Conversely, traditional algorithms such as RSA (rsa:2048, rsa:3072, rsa:4096) exhibit remarkably efficient memory utilization, consistently maintaining usage below 10 KB, which reflects decades of implementation optimization. The lattice-based candidates show consistent and efficient memory profiles:

- DILITHIUM/MLDSA (standardized version of DILITHIUM) implementations

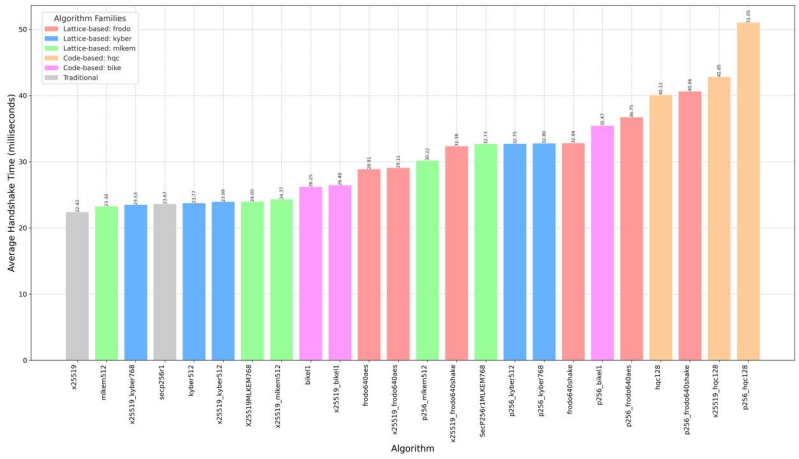
maintain moderate memory footprints (9-14 KB), with minimal variation across security levels. This consistency in memory usage across security levels is particularly noteworthy for resource planning.

- FALCON implementations show similar efficiency (12-14 KB)

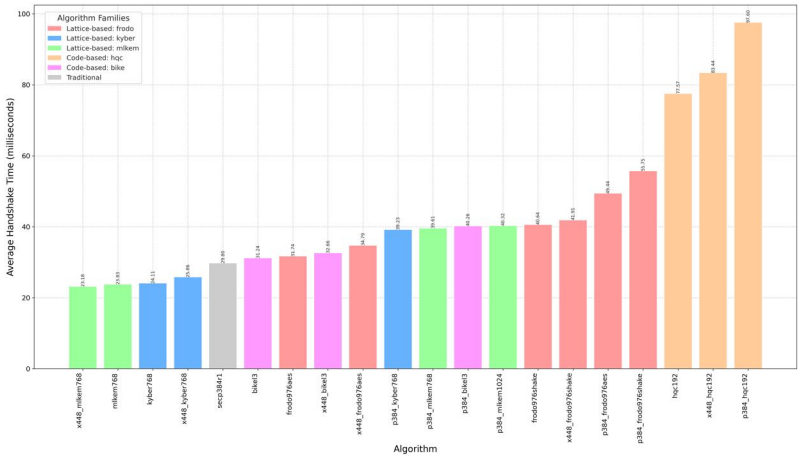
Notably, hybrid implementations combining classical and post-quantum algorithms show only marginal increases in memory overhead compared to their standalone counterparts. For instance, p256_falcon512 requires approximately 13.07 KB, only slightly higher than the base falcon512 implementation at 12.79 KB. This suggests that hybrid approaches, which provide a security hedge during the transition to post-quantum cryptography, come with minimal memory overhead. The memory analysis indicates that while there are significant variations across different signature algorithms, the absolute memory requirements remain modest for modern computing environments. Even the most memory-intensive implementations rarely exceed 50 KB, suggesting that memory consumption should not be a primary constraint in algorithm selection for most applications.

5.2.3 Security Level Analysis

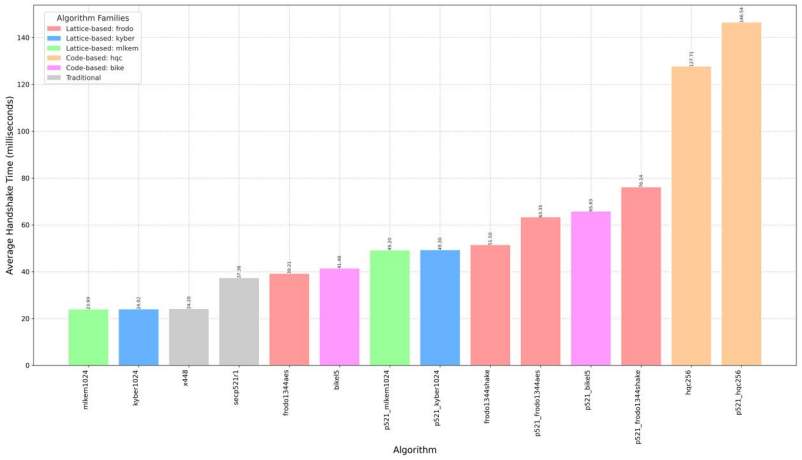
KEM Security Level Analysis



(a) Level = 1



(b) Level = 3

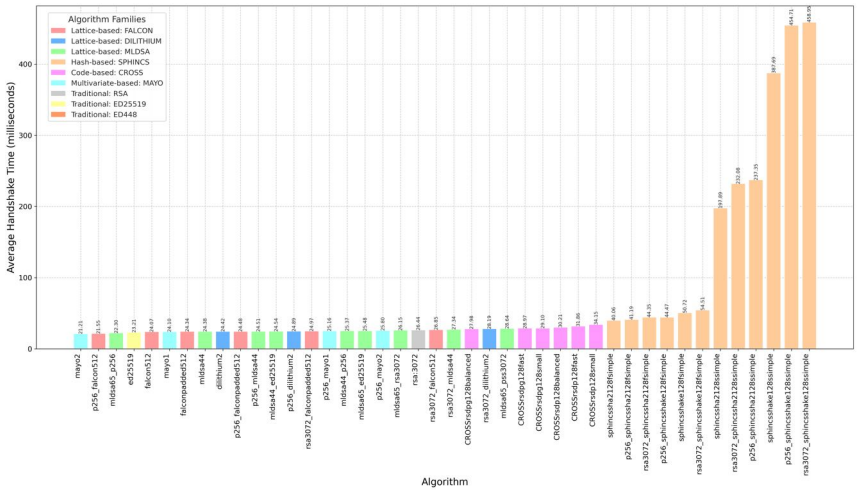


(c) Level = 5

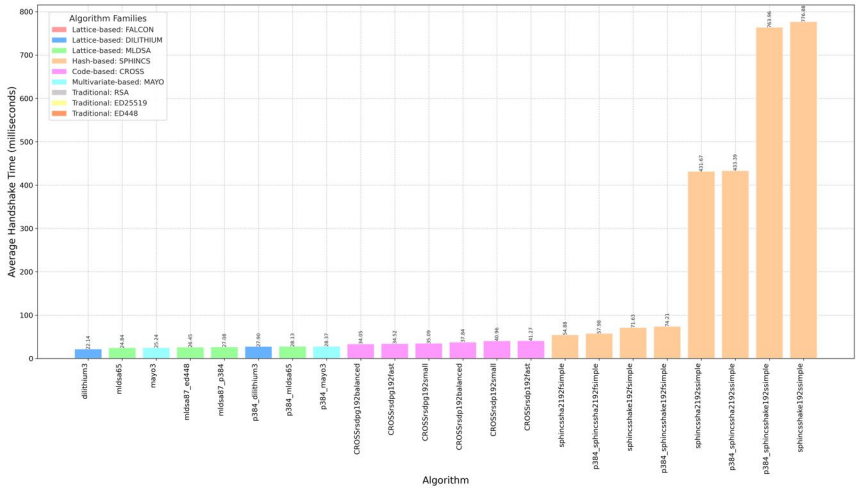
Figure 13: KEM Algorithm Handshake Time under Different Security Level

From our experimental results, the Kyber and MLKEM families consistently demonstrate superior performance across increasing security levels, maintaining the shortest execution times within their respective security categories. The general trend shows that as security levels increase, execution times for algorithms within the same family gradually increase, with the time gap between faster and slower algorithms becoming more pronounced. Notably, the introduction of hybrid algorithms at equivalent security levels shows negligible impact on execution time.

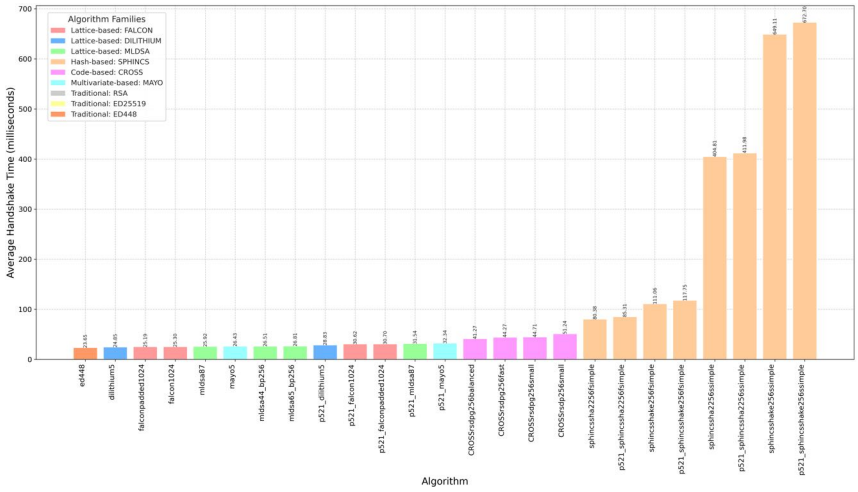
SIG Security Level Analysis



(a) Level = 1



(b) Level = 3



(c) Level = 5

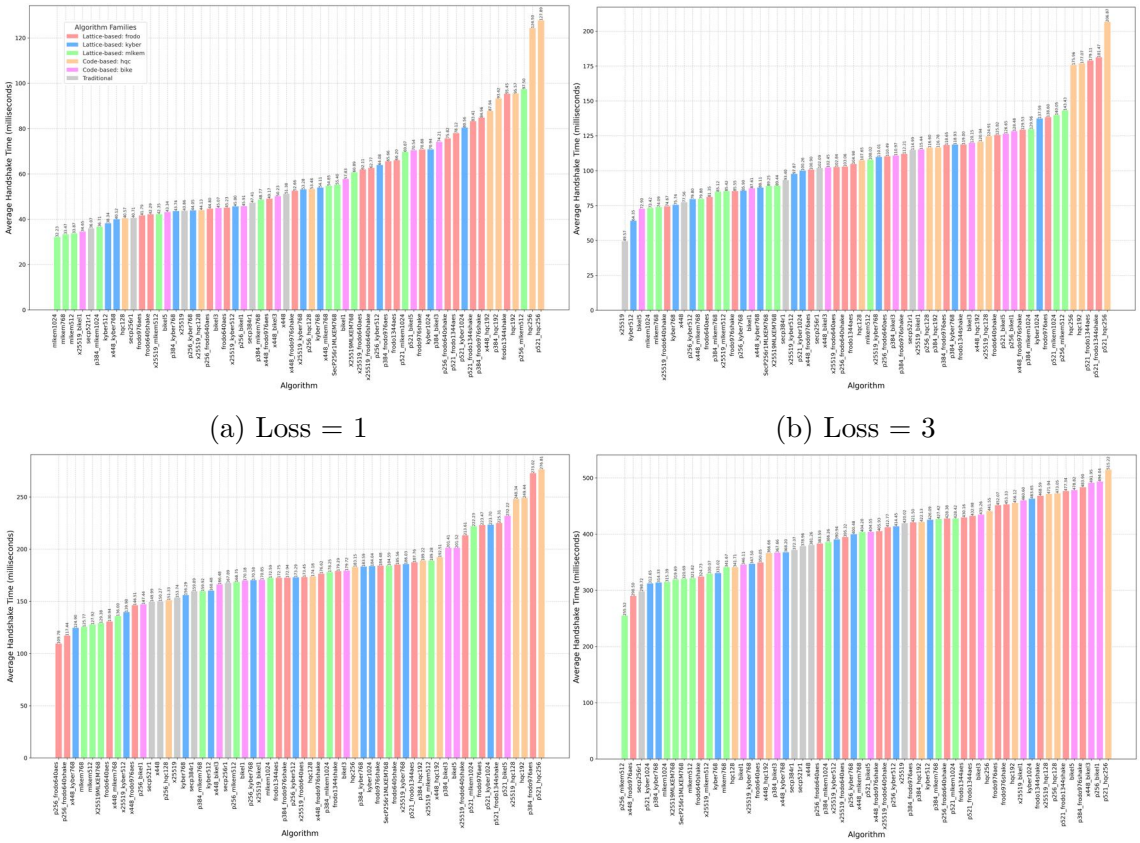
Figure 14: SIG Algorithm Handshake Time under Different Security Level

The SPHINCS+ family consistently exhibits the longest execution times across all security levels, with its "small" variants requiring even more computational resources. This observation reinforces our finding regarding hybrid implementations: the combination of post-quantum algorithms with traditional cryptographic methods introduces minimal temporal overhead. The performance analysis clearly demonstrates that the Dilithium family, along with MLDSA, MAYO, and Falcon series, all exhibit excellent performance characteristics.

5.3 Network Impact

5.3.1 Impact of Packet Loss

KEM Algorithm Impact



(a) Loss = 1

(b) Loss = 3

(c) Loss = 5

(d) Loss = 10

Figure 15: KEM Algorithm Performance under Different Loss Values

Our empirical analysis reveals that packet loss rates ranging from 1% to 10% do not affect handshake success rates but significantly impact handshake latency. CPU and memory utilization remain relatively stable across different packet loss conditions. However, the introduction of network uncertainty leads to a notable increase in the standard deviation of handshake times across all algorithms, warranting a focused analysis on handshake latency patterns.

The impact of increasing packet loss rates on handshake times exhibits non-linear characteristics. Interestingly, as packet loss rates increase, the relative performance gap between the fastest and slowest algorithms gradually narrows, decreasing from a three-fold difference at 1% packet loss to approximately two-fold at 10%. This observation suggests that as network conditions deteriorate, the algorithmic differences become less significant compared to the dominant effect of packet loss on overall handshake latency.

Traditional algorithms maintain relatively stable performance under lower packet loss rates, preserving their handshake speed advantage. However, they show slightly increased degradation at higher loss rates, particularly when compared to Kyber and MLKEM families. These lattice-based families demonstrate exceptional resilience to packet loss, with different security levels exhibiting remarkably consistent performance across varying loss rates. Moreover, they maintain superior performance under high packet loss conditions, even outperforming traditional algorithms in some cases.

Different algorithm families exhibit distinct behavioral patterns under packet loss conditions. The HQC family, while consistently requiring the longest handshake times, shows an interesting trend. For instance, p521_hqc256 reaches handshake times of up to 515ms at 10% packet loss. However, the relative performance gap with other algorithms diminishes compared to ideal network conditions, though its base performance limitations persist.

The BIKE family shows particular sensitivity to packet loss, experiencing severe degradation at higher loss rates. Notably, at 10% packet loss, BIKE algorithms approach HQC-level latencies, becoming among the slowest performers. This stands in stark contrast to their stable performance scaling with security levels under ideal conditions, indicating a particular vulnerability to network degradation.

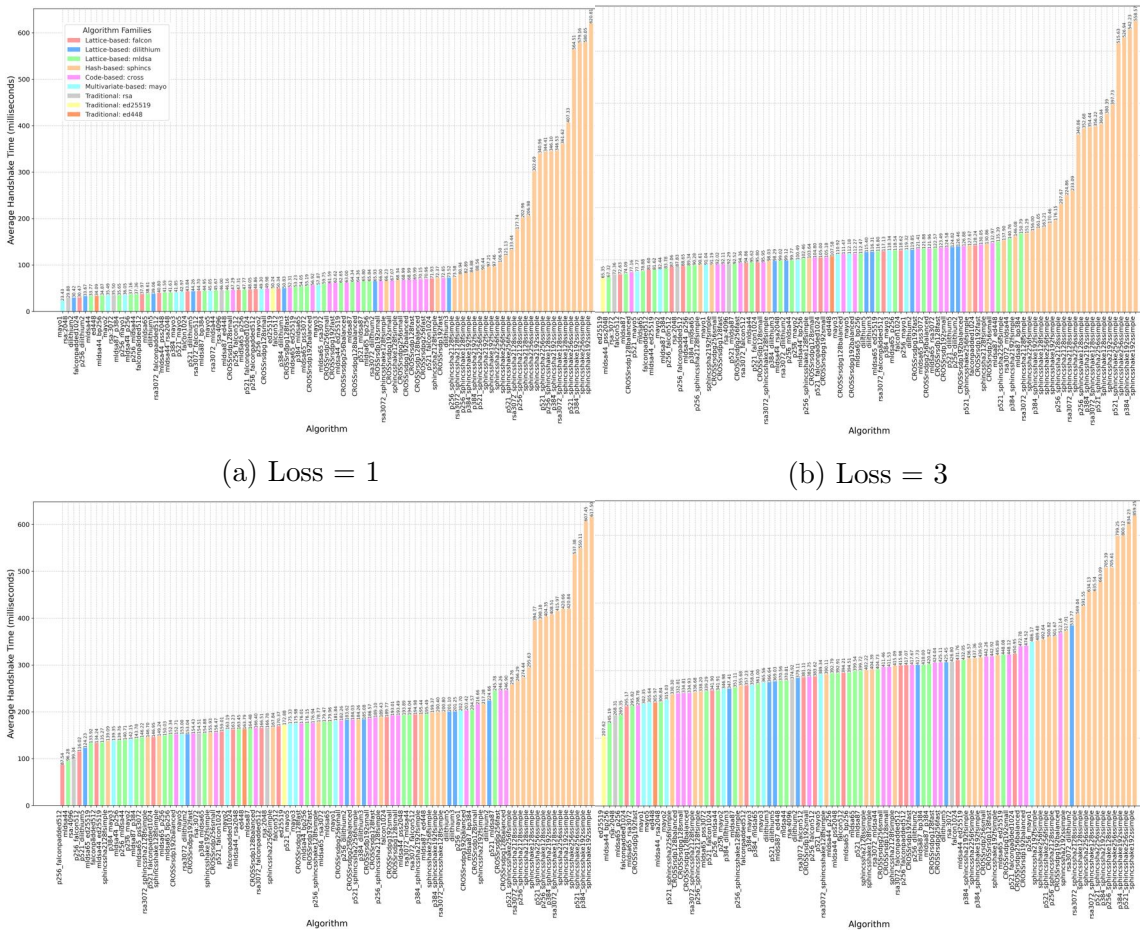
The FRODO family maintains consistent relative performance distribution across different packet loss rates, demonstrating good uniformity. However, at higher security levels, its handshake times still fail to provide a competitive advantage over other approaches.

Analysis of hybrid schemes reveals an interesting trend: while hybridization generally increases handshake times, the relative stability of traditional algorithm degradation means that performance differences primarily stem from the post-quantum component. The hybrid combination itself does not significantly exacerbate the impact of packet loss on handshake times.

Key findings from our analysis include:

- The Kyber and MLKEM families demonstrate superior packet loss resilience
- Base algorithm performance does not necessarily correlate with packet loss resilience
- In high packet loss environments, performance degradation characteristics should take precedence over ideal-condition performance when selecting algorithms

SIG Algorithm Impact



(a) Loss = 1 (b) Loss = 3

(c) Loss = 5 (d) Loss = 10

Figure 16: Signature Algorithm Memory Performance under Different Loss Values

Based on our data analysis, we arrive at conclusions similar to the previous section: packet loss rates from 1% to 10% do not affect handshake success rates, with the only significant impact being on handshake latency. As packet loss rates increase, all algorithms experience notable increases in both handshake times and standard deviations.

This increase demonstrates non-linear characteristics, with the performance gap between the fastest and slowest algorithms gradually narrowing. As packet loss rates rise, the algorithmic influence on handshake times progressively diminishes, with packet loss becoming the dominant factor affecting handshake latency.

SPHINCS+, due to its inherently lower baseline performance, continues to exhibit the longest handshake times even with increased packet loss. Notably, hybrid

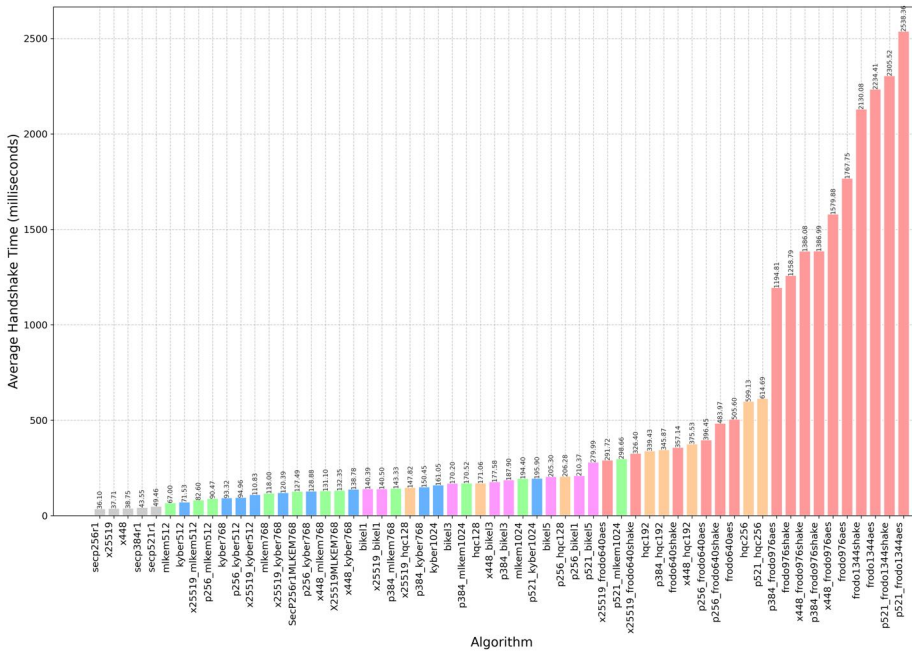
combinations of post-quantum and traditional algorithms do not show additional sensitivity to packet loss beyond their individual components.

Other algorithms do not display any abnormal degradation patterns. Algorithms that perform well under ideal conditions maintain their relative advantage even under packet loss conditions. The Mayo, Falcon, Dilithium, and MLDSA families all demonstrate excellent performance, while the CROSS family shows moderate performance - though all significantly outperform the SPHINCS+ family.

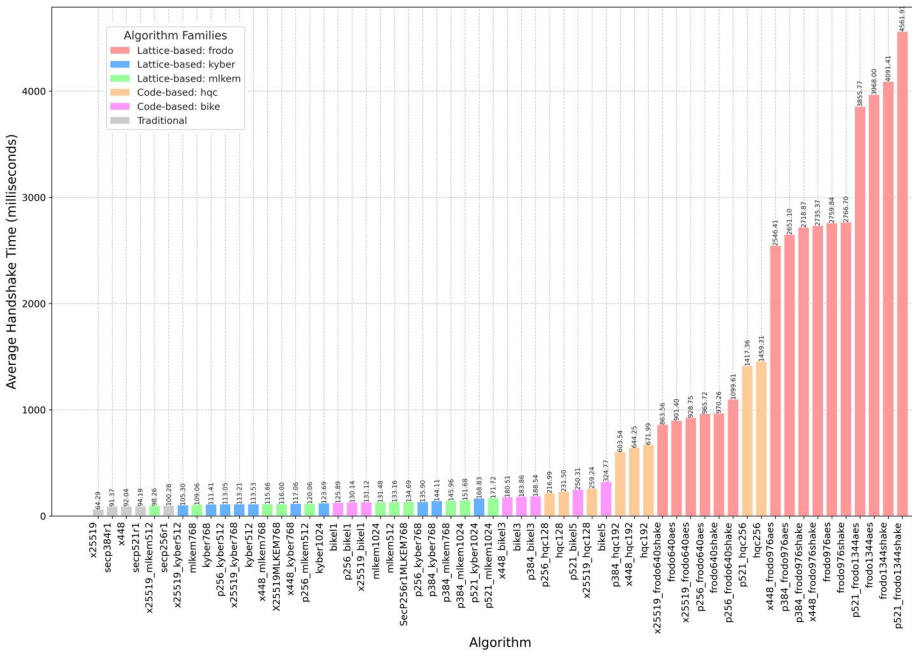
In high packet loss environments, we observe a convergence trend in performance differences across algorithm families, suggesting that network conditions become increasingly dominant over algorithmic characteristics in determining overall handshake latency.

5.3.2 Impact of Bandwidth

KEM Algorithm Impact

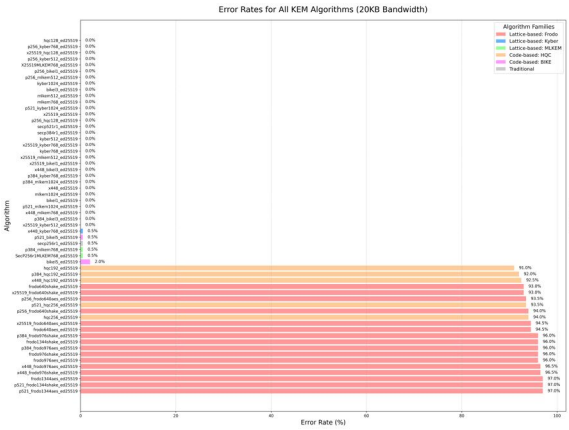


(a) Bandwidth = 1mbits

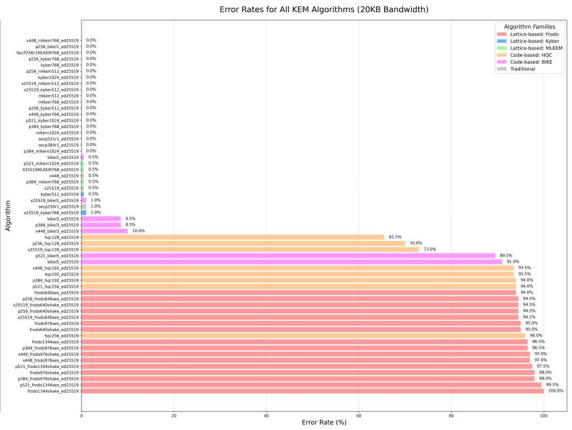


(b) Bandwidth = 500kbits

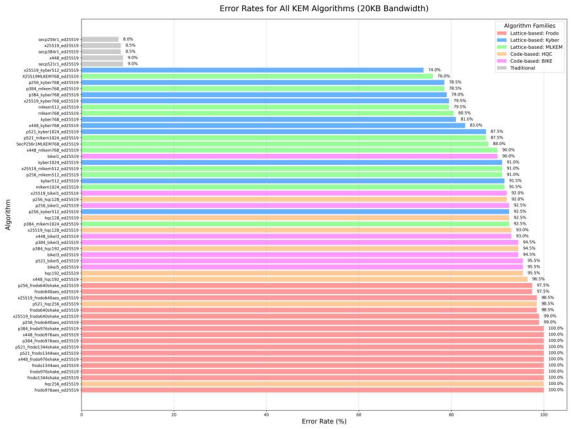
Figure 17: KEM Algorithm Performance under Different Bandwidth Values



(a) Bandwidth = 100kbits



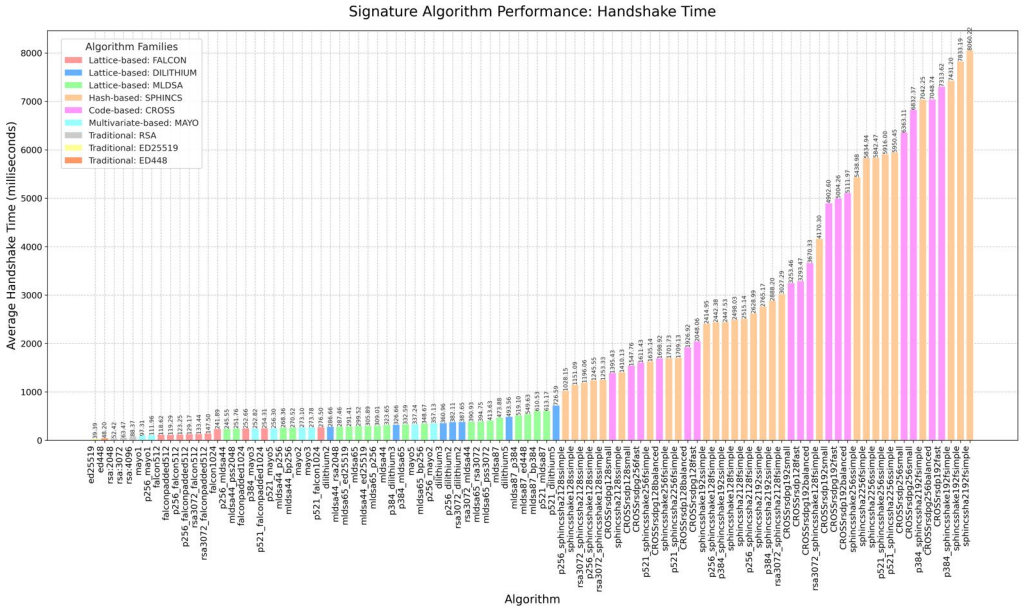
(b) Bandwidth = 60kbits



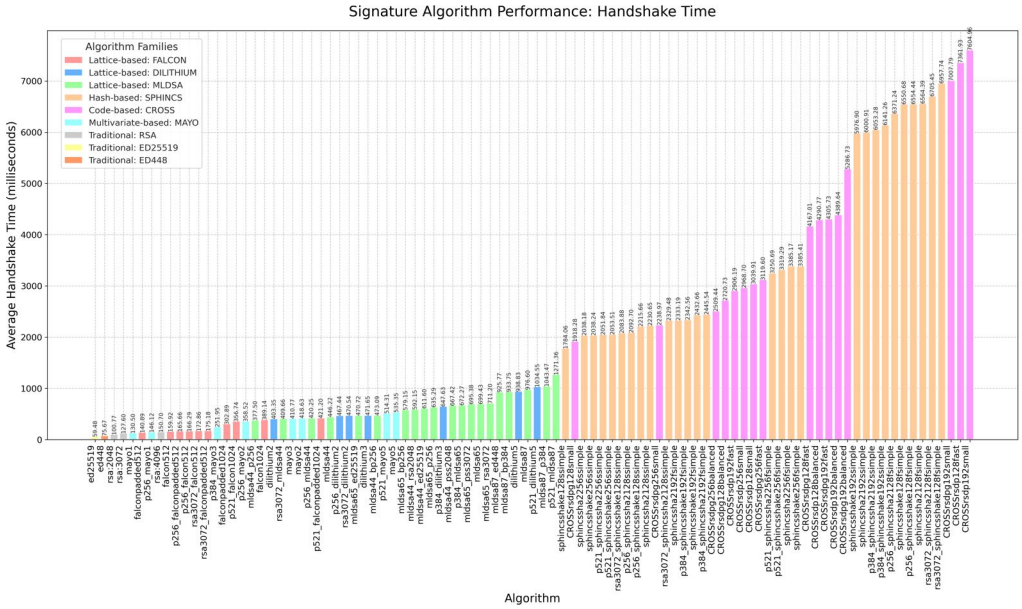
(c) Bandwidth = 20kbits

Figure 18: KEM Algorithm Error Rate under Different Bandwidth Values

SIG Algorithm Impact

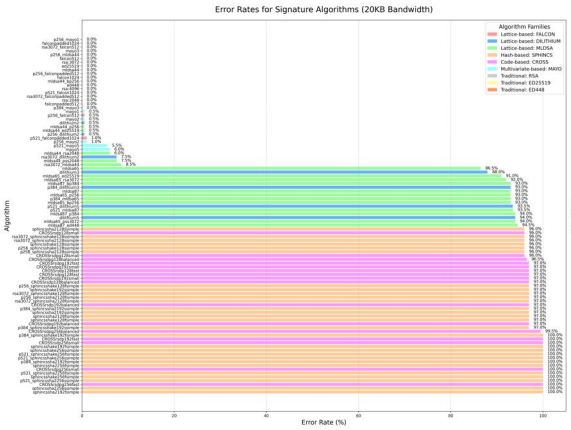


(a) Bandwidth = 1mbits

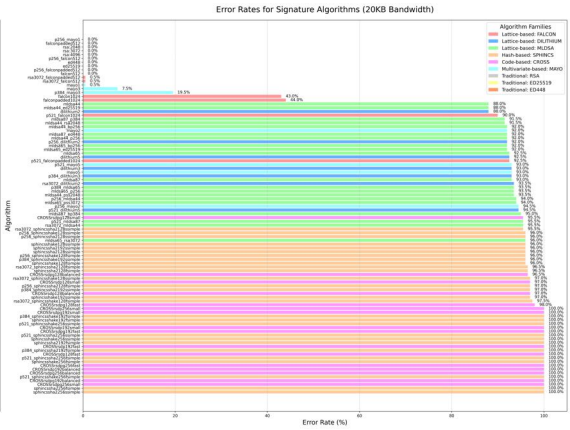


(b) Bandwidth = 500kbits

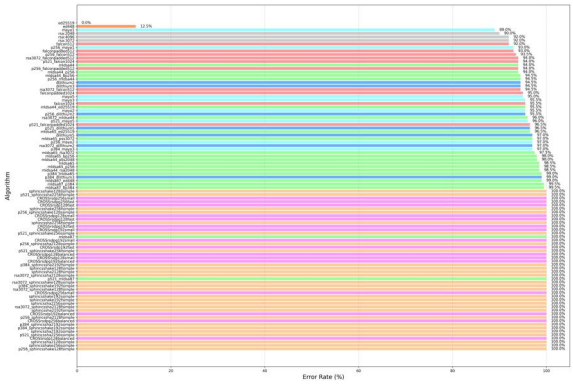
Figure 19: SIG Algorithm Performance under Different Bandwidth Values



(a) Bandwidth = 100kbits



(b) Bandwidth = 60kbits

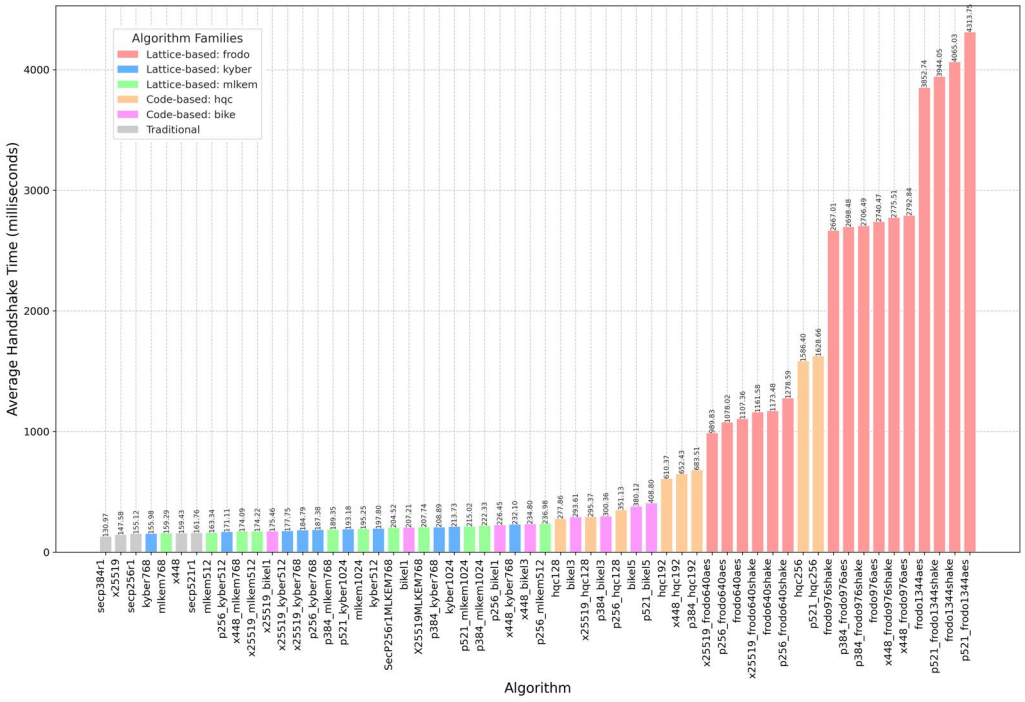


(c) Bandwidth = 20kbits

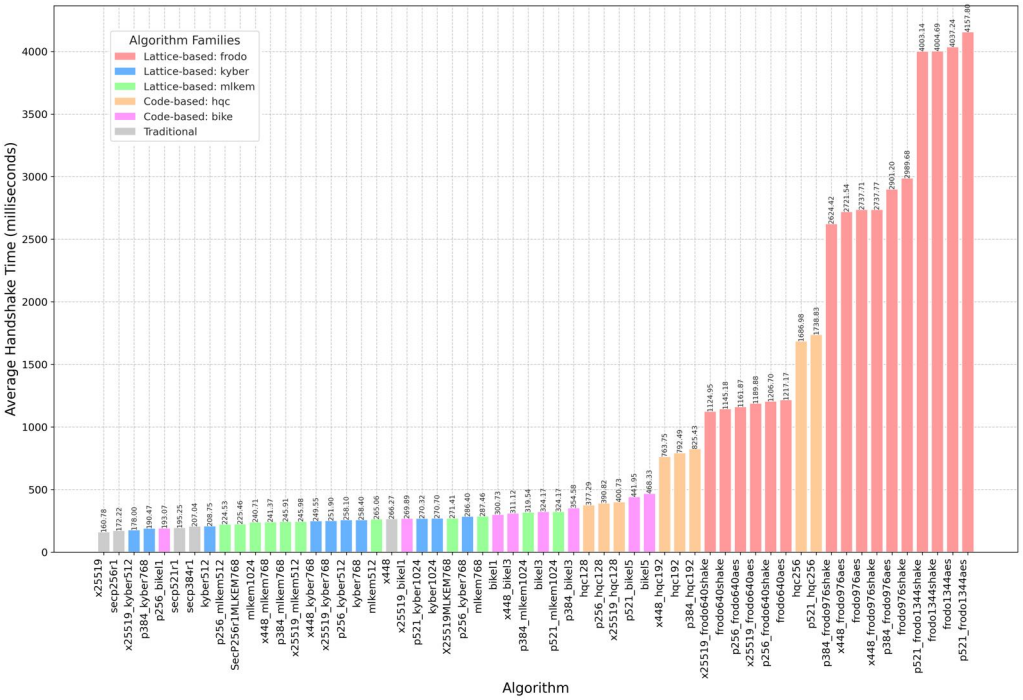
Figure 20: SIG Algorithm Error Rate under Different Bandwidth Values

5.3.3 Comprehensive Network Impact

KEM Algorithm Impact



(a) Bandwidth = 500kbits Loss = 3

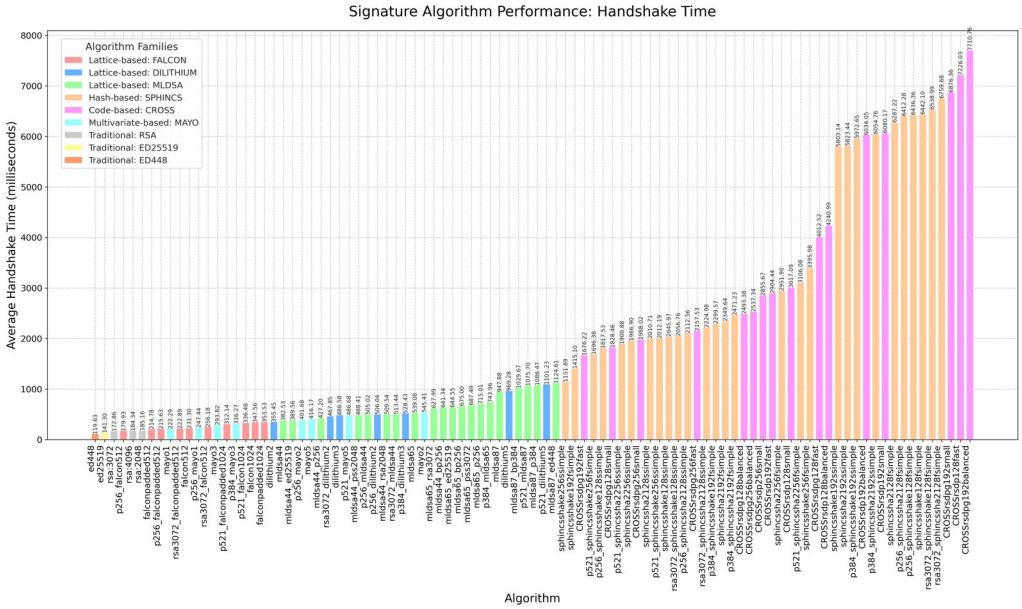


(b) Bandwidth = 500kbits Loss = 5

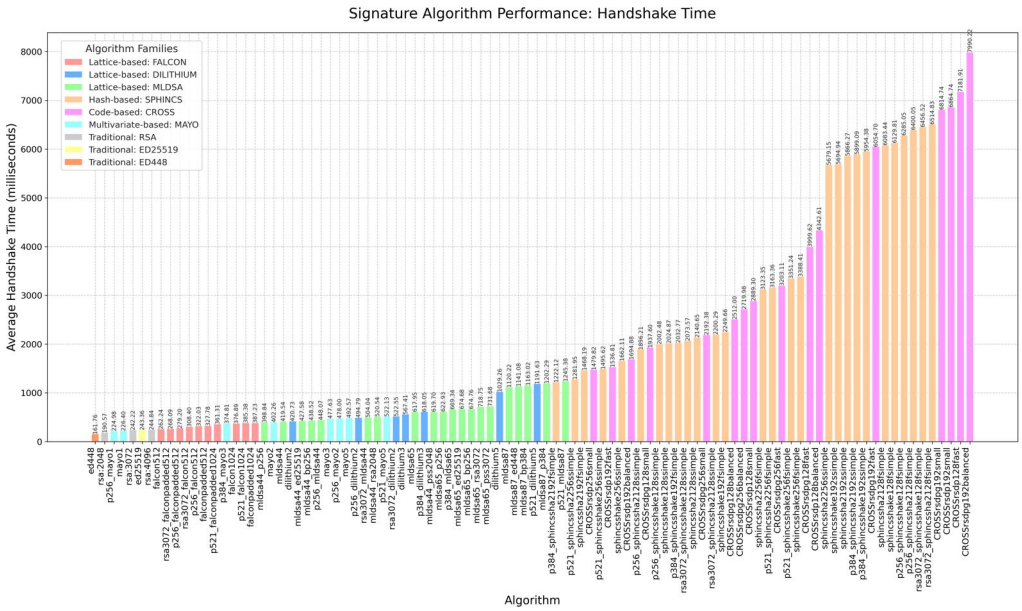
Figure 21: KEM Algorithm Performance under Different Network conditions

We can observe that under low bandwidth conditions, increased packet loss affects handshake times, but the algorithmic trends and degradation patterns remain consistent. As error rates show no significant variation, we have chosen to omit these graphs from the main text.

SIG Algorithm Impact

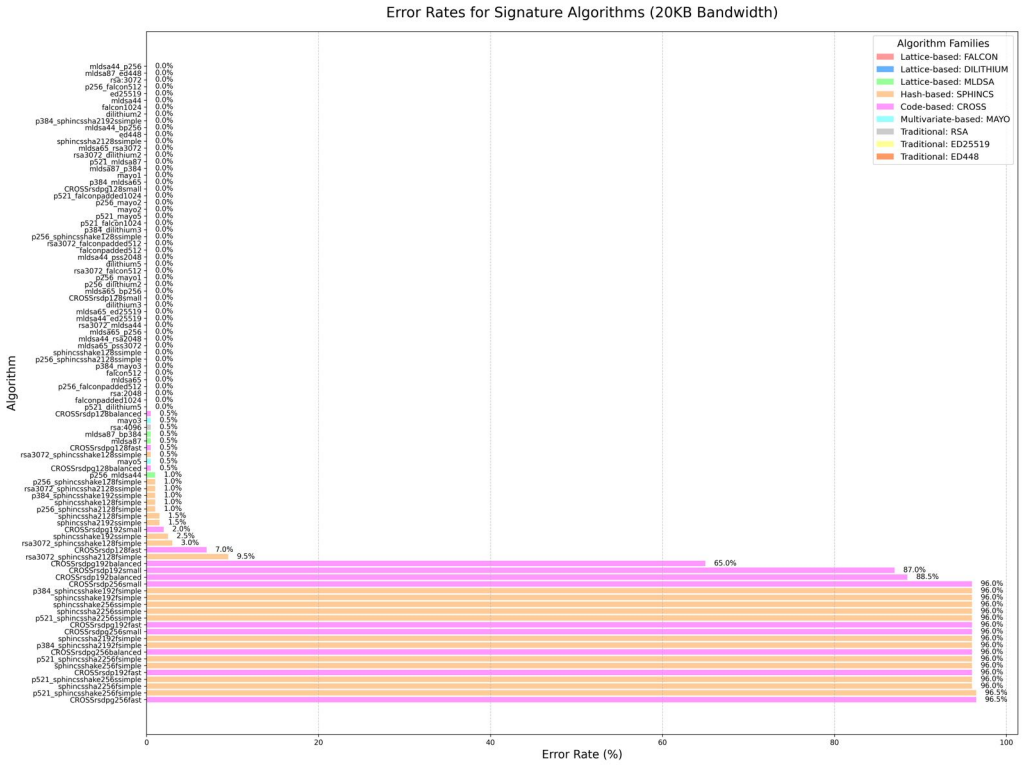


(a) Bandwidth = 500kbits Loss = 3

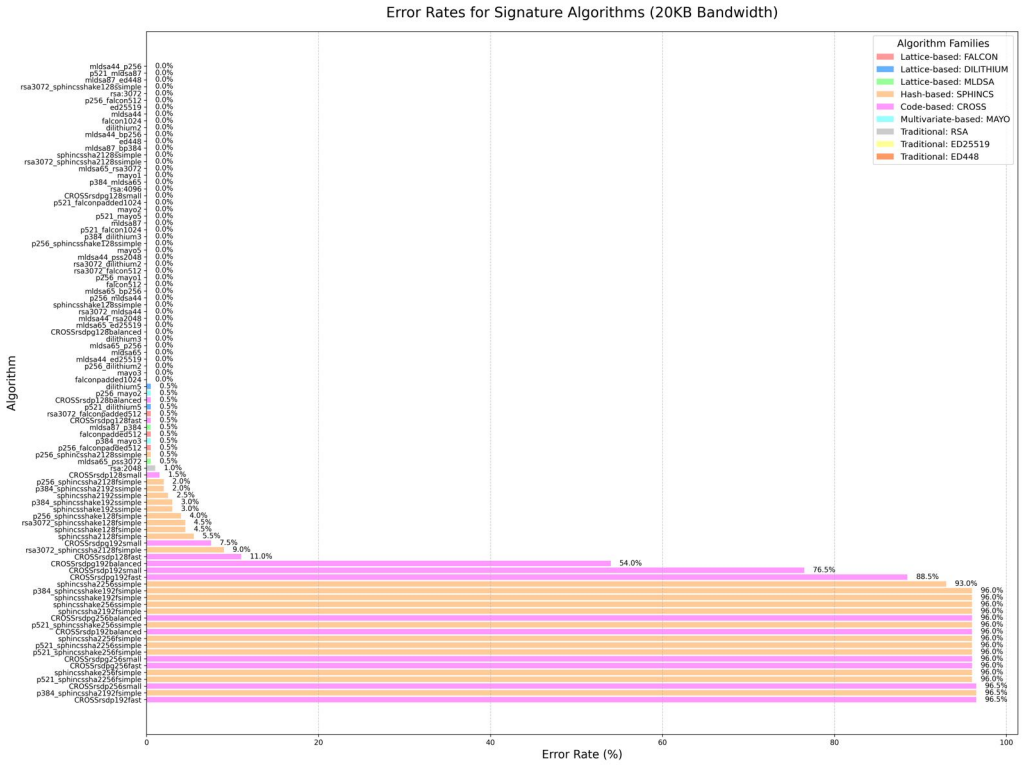


(b) Bandwidth = 500kbits Loss = 5

Figure 22: SIG Algorithm Performance under Different Network conditions



(a) Bandwidth = 500kbits Loss = 3



(b) Bandwidth = 500kbits Loss = 5

Figure 23: SIG Algorithm Error Rate under Different Network conditions

We arrive at conclusions similar to the previous section: increased packet loss only impacts handshake times, though for signature algorithms, there is a slight increase in error rates.

6 Discussion

6.0.1 Performance Bottleneck Analysis

Through observation, we find that algorithms in the TLS handshake process show higher sensitivity to bandwidth limitations than to packet loss. Low bandwidth conditions lead to increased handshake failure rates and longer handshake times, while packet loss only affects handshake duration. Within the same algorithm family, handshake times increase with higher security levels, indicating that key size is one of the critical factors affecting performance.

The use of hybrid algorithms only introduces minimal performance overhead and slight increases in handshake times, thus not constituting a performance bottleneck. The underlying principles of post-quantum algorithms represent a primary factor in handshake time variations, with hash-based algorithms generally consuming more time and lattice-based encryption algorithms requiring shorter processing times.

6.0.2 Algorithm Selection Guidelines

Based on all the results and analyses, for KEM algorithms, we recommend using the NIST-standardized MLKEM algorithm in combination with traditional algorithms, such as elliptic curve cryptography. MLKEM achieved the best performance in handshake times, including in step-by-step testing. Its memory usage is the lowest among post-quantum algorithms, second only to traditional algorithms. It outperforms other algorithms in speed across all security levels. Although it shares the same principles as Kyber, its NIST standardization makes it more aligned with future TLS standards. Before undergoing further validation, we recommend using hybrid algorithms to ensure dual security, as they do not significantly impact performance. MLKEM also demonstrates superior performance under network constraints, exhibiting excellent performance even under extreme bandwidth limitations, achieving the highest success rate among post-quantum algorithms, though still requiring further optimization to match traditional algorithms.

For Signature algorithms, we recommend using the Falcon family combined with traditional algorithms. While Mayo and NIST-standardized MLDSA are also worthy considerations, the Falcon family demonstrates the best overall performance, par-

ticularly showing the lowest failure rates under low bandwidth conditions compared to other post-quantum algorithms. In other aspects, such as handshake time and memory usage, it ranks among the best performers. Under packet loss conditions, it maintains consistent performance with very low handshake times.

6.0.3 Limitations and Challenges

The current project has several limitations in terms of accuracy and realism in experimental setup. First, the simulation of real-world conditions could be improved by employing two separate devices as server and client, with time measurements taken using more precise hardware timestamps, which would enhance the credibility of the findings. Additionally, the project does not yet encompass a comprehensive set of algorithms. Integrating emerging quantum algorithms into the OQS provider remains a significant challenge but offers a valuable direction for future work.

Moreover, rewriting the entire project in C could yield more accurate data due to lower-level system access. Testing across different chip frameworks is also essential to evaluate algorithmic performance on varied hardware, as this would provide insights into cross-platform behavior. Replacing the dependency library with alternatives like Botan, and incorporating real low-power devices to simulate energy constraints, are further areas where the project could be enhanced to increase both applicability and precision.

7 Conclusions

Through extensive algorithmic comparisons and environmental simulations, this experiment has evaluated the performance of current mainstream algorithms. From our analysis, we can draw meaningful conclusions: hybrid algorithms only introduce minimal impact on the TLS handshake process, while bandwidth limitations have a more significant effect on handshake performance compared to packet loss, both in terms of latency and success rates.

Under ideal conditions, the algorithmic principles and key sizes remain the primary factors affecting algorithm performance. Through comparative analysis, this experiment concludes that MLKEM (combined with traditional algorithms) for key exchange and Falcon (combined with traditional algorithms) for signatures represent the most promising algorithmic choices during the transition phase to post-quantum cryptography.

References

[1] Mahto, D., & Yadav, D. K. (2017). RSA and ECC: A comparative analysis. *International Journal of Applied Engineering Research*, 12(19), 9053-9061.

[2] Ugwuishiwu, C. H., Orji, U. E., Ugwu, C. I., & Asogwa, C. N. (2020). An overview of quantum cryptography and Shors algorithm. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(5).

[3] Long, G. L. (2001). Grover algorithm with zero theoretical failure rate. *Physical Review A*, 64(2), 022307.

[4] Daemen, J. (1999). AES Proposal: Rijndael.

[5] Alkim, E., Bos, J. W., Ducas, L., Longa, P., Mironov, I., Naehrig, M., Nikolaenko, V., Peikert, C., Raghunathan, A., & Stebila, D. (2021). Learning With Errors Key Encapsulation.

[6] FIPS203, N. I. S. T. (2023). Module-Lattice-based Key-Encapsulation Mechanism Standard. *Federal Information Processing Standards Publication*.

[7] Avanzi, R., Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., ... & Stehl, D. (2019). CRYSTALS-Kyber algorithm specifications and supporting documentation. *NIST PQC Round*, 2(4), 1-43.

[8] FIPS204, N. I. S. T. (2024). Module-Lattice-Based Digital Signature Standard. *Federal Information Processing Standards Publication*. DOI: 10.6028/NIST.FIPS.204

[9] Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., & Stehl, D. (2018). Crystals-dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 238-268.

[10] Kiningham, K., Levis, P., Anderson, M., Boneh, D., Horowitz, M., & Shih, M. (2019). FalconA flexible architecture for accelerating cryptography. *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)* (pp. 136-144). IEEE.

[11] Bernstein, D. J., Hlsing, A., Klbl, S., Niederhagen, R., Rijneveld, J., & Schwabe, P. (2019). The SPHINCS+ Signature Framework. *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 21292146.

[12] McEliece, R. J. (1978). A public-key cryptosystem based on algebraic. *Coding Thv*, 4244, 114-116.

[13] Aragon, N., Barreto, P., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Ghosh, S., Gueron, S., Gneysu, T., et al. (2022). BIKE: bit flipping key encapsulation.

[14] Melchor, C. A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J.-C., Gaborit, P., Persichetti, E., Zmor, G., & Bourges, I. C. (2018). Hamming quasi-cyclic (HQC). *NIST PQC Round*, 2(4), 13.

[15] Beullens, W. (2021). MAYO: practical post-quantum signatures from oil-and-vinegar maps. In *International Conference on Selected Areas in Cryptography* (pp. 355-376). Springer.

[16] Maurer, U. M., & Wolf, S. (2000). The DiffieHellman Protocol. *Designs, Codes and Cryptography*, 19(2), 147-171.

[17] Open Quantum Safe. (2024). liboqs. Retrieved from https://github.com/open-quantum-safe/liboqs

[18] Open Quantum Safe. (2024). oqs-provider. Retrieved from https://github.com/open-quantum-safe/oqs-provider

[19] Sikeridis, D., Kampanakis, P., & Devetsikiotis, M. (2020b). Post-Quantum Authentication in TLS 1.3: A Performance Study. *Proceedings 2020 Network and Distributed System Security Symposium*.

[20] Sosnowski, M., Wiedner, F., Hauser, E., Steger, L., Schoinianakis, D., Gallenmller, S., & Carle, G. (2023). The Performance of Post-Quantum TLS 1.3. *Companion of the 19th International Conference on Emerging Networking EXperiments and Technologies*, 1927.

[21] Raavi, M., Wuthier, S., Chandramouli, P., Balytskyi, Y., Zhou, X., & Chang, S.-Y. (2021). Security Comparisons and Performance Analyses of Post-quantum Signature Algorithms. In K. Sako & N. O. Tippenhauer (Eds.), *Applied Cryptography and Network Security* (Vol. 12727, pp. 424447). Springer International Publishing.

[22] Paquin, C., Stebila, D., & Tamvada, G. (2020). Benchmarking Post-quantum Cryptography in TLS. In J. Ding & J.-P. Tillich (Eds.), *Post-Quantum Cryptography* (Vol. 12100, pp. 7291). Springer International Publishing.

[23] Dowling, B., Fischlin, M., Gnther, F., & Stebila, D. (2021b). A Cryptographic Analysis of the TLS 1.3 Handshake Protocol. *Journal of Cryptology, 34*(4), 37.

[24] Lee, H., Kim, D., & Kwon, Y. (2021). TLS 1.3 in Practice: How TLS 1.3 Contributes to the Internet. *Proceedings of the Web Conference 2021*, 7079.

[25] Roma, C. A., Tai, C.-E. A., & Hasan, M. A. (2021). Energy Efficiency Analysis of Post-Quantum Cryptographic Algorithms. *IEEE Access, 9*, 7129571317.

[26] Gan, L., & Yokubov, B. (2023). A Performance Comparison of Post-Quantum Algorithms in Blockchain. *The Journal of The British Blockchain Association*, 6(1), 110.

[27] Raavi, M., Chandramouli, P., Wuthier, S., Zhou, X., & Chang, S.-Y. (2021). Performance Characterization of Post-Quantum Digital Certificates. 2021 International Conference on Computer Communications and Networks (ICCCN), 19.

[28] Schffel, M., Lauer, F., Rheinlnder, C. C., & Wehn, N. (2021). On the Energy Costs of Post-Quantum KEMs in TLS-based Low-Power Secure IoT. *Proceedings of the International Conference on Internet-of-Things Design and Implementation*, 158168.

[29] Schwabe, P., Stebila, D., & Wiggers, T. (2020). Post-Quantum TLS Without Handshake Signatures. *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, 14611480.

[30] Sikeridis, D., Kampanakis, P., & Devetsikiotis, M. (2020a). Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH. *Proceedings of the 16th International Conference on Emerging Networking EXperiments and Technologies*, 149156.

[31] Tasopoulos, G., Dimopoulos, C., Fournaris, A. P., Zhao, R. K., Sakzad, A., & Steinfeld, R. (2023). Energy Consumption Evaluation of Post-Quantum TLS 1.3 for Resource-Constrained Embedded Devices. *Proceedings of the 20th ACM International Conference on Computing Frontiers*, 366374.

[32] Tasopoulos, G., Li, J., Fournaris, A. P., Zhao, R. K., Sakzad, A., & Steinfeld, R. Performance Evaluation of Post-Quantum TLS 1.3 on Embedded Systems.

[33] Tzinos, I., Limniotis, K., & Kolokotronis, N. (2022). Evaluating the performance of post-quantum secure algorithms in the TLS protocol. *Journal of Surveillance, Security and Safety*, 3(3), 101127.

[34] Hoffstein, J., Pipher, J., & Silverman, J. H. (1998, June). NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium* (pp. 267-288). Berlin, Heidelberg: Springer Berlin Heidelberg.

[35] Becker, G. (2008). Merkle signature schemes, merkle trees and their cryptanalysis. *Ruhr-University Bochum, Tech. Rep*, 12, 19.

[36] Liu, Peidong. (2024). Performance Analysis of Post-Quantum and Variant Algorithm on TLS 1.3. Retrieved from https://github.com/EscapedShark/TLS_PQC_performance

Acknowledgements

I would like to sincerely thank my mentor, Dr. Dong Naipeng, for his continuous support, guidance, and valuable insights throughout the research process. Her expertise and encouragement played a significant role in the completion of this thesis. Secondly, I would like to thank my parents and uncle for their support and assistance in many other aspects during this time. Finally, I would like to thank the University of Queensland for providing a good learning environment.

Additionally, I would like to thank my girlfriend Ms. Zixun Huang, who is far away in Adelaide, for her emotional support.