Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Performance Evaluation of NIST Post-Quantum Cryptographic Key Encapsulation Mechanisms in a Virtualized Software Environment

Chirag Gupta

cjwrits@gmail.com

Dr. A.P.J. Abdul Kalam Technical University

Research Article

Keywords: Post-Quantum Cryptography, CRYSTALS-Kyber, ML-KEM, Key Encapsulation Mechanism, NIST Standardization, Benchmarking, Lattice-Based Cryptography, Virtualization, Cybersecurity

Posted Date: September 30th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11197227/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Performance Evaluation of NIST Post-Quantum Cryptographic Key Encapsulation Mechanisms in a Virtualized Software Environment

Chirag Gupta
B.Tech Information Technology Student
Independent Researcher
Lucknow, Uttar Pradesh, India
Email: cjwrits@gmail.com

***Abstract*—This paper presents a software-based performance evaluation of NIST-standardized post-quantum cryptographic key encapsulation mechanisms. The study benchmarks CRYSTALS-Kyber (Kyber512, Kyber768, and Kyber1024) against classical cryptographic algorithms including RSA-2048 and ECDH-P256 in a controlled Ubuntu virtual machine environment. Performance metrics such as key generation time, encapsulation time, decapsulation time, and public key size are evaluated to analyze the practicality of post-quantum cryptography for future deployment. The experimental results obtained in the virtualized software environment indicate that Kyber achieves substantially faster key generation than RSA-2048 while exhibiting performance comparable to ECDH-P256, demonstrating its practicality for future quantum-resistant deployments.**

***Index Terms*—Post-Quantum Cryptography, CRYSTALS-Kyber, NIST, Quantum Computing, Key Encapsulation Mechanism, Cybersecurity, Benchmarking, Lattice-Based Cryptography**

I. Introduction

The rapid advancement of quantum computing poses an existential threat to modern digital security infrastructures. Classical public-key cryptographic systems, such as Rivest-Shamir-Adleman (RSA) and Elliptic Curve Cryptography (ECC), rely on the computational hardness of integer factorization and discrete logarithms. However, Shor's polynomial-time quantum algorithm [2] renders these mathematical foundations vulnerable, meaning that a sufficiently powerful cryptanalytically useful quantum computer (CRQC) could decrypt encrypted data and forge digital signatures globally.

In response, the National Institute of Standards and Technology (NIST) initiated a multi-year standardization process to identify and standardize quantum-resistant public-key cryptographic algorithms [1]. Among the primary focus areas is the Key Encapsulation Mechanism (KEM), which is crucial for establishing shared secrets over insecure channels. In August 2024, NIST released FIPS 203, officially standardizing the Module-Lattice-Based Key-Encapsulation Mechanism, known as ML-KEM (originally derived from CRYSTALS-Kyber) [1].

As organizations prepare for this paradigm shift, migrating to post-quantum cryptography (PQC) requires thorough empirical evaluation. It is essential to understand the practical performance characteristics of these new algorithms under realistic operational conditions.

A. Motivation

Modern IT infrastructure relies heavily on virtualization, cloud computing, and software-defined environments. A significant portion of web servers, microservices, and cryptographic appliances operate within virtual machines (VMs) rather than bare-metal hardware. Consequently, evaluating how PQC algorithms perform under virtualized CPU and memory constraints is critical.

Lattice-based cryptography, which underlies Kyber/ML-KEM, utilizes matrix multiplications and polynomial rings, which behave differently from the modular exponentiation of RSA or the group operations of Elliptic Curve Diffie-Hellman (ECDH). It is vital to assess whether virtualized software environments introduce performance overheads or stability issues for these operations. Developers need empirical data on execution latency—specifically key generation, encapsulation, and decapsulation—and storage costs (public key sizes) to make informed design decisions when upgrading security protocols like Transport Layer Security (TLS) or Secure Shell (SSH).

B. Research Gap

While there is substantial literature discussing the theoretical complexities of lattice-based cryptography and hardware-accelerated PQC implementations (such as FPGA or ASIC designs), empirical benchmarking of standardized PQC software implementations on commodity virtualized operating systems remains limited.

Many existing performance studies focus on bare-metal architectures or embedded devices. Furthermore, comparison analyses often omit direct, concurrent benchmarks against classical baselines (such as RSA-2048 and ECDH-P256) running under identical virtual hypervisor configurations. Evaluating all three standardized Kyber security levels (Kyber512, Kyber768, and Kyber1024) side-by-side with classical counterparts in a single virtualized OS environment is necessary to establish a clear migration blueprint.

C. Contributions

In this paper, we address these gaps by conducting a systematic performance evaluation of CRYSTALS-Kyber against classical cryptographic standards in a virtualized software environment. The primary contributions of this work are:

- We perform a side-by-side benchmarking of Kyber512, Kyber768, Kyber1024, RSA-2048, and ECDH-P256, measuring execution times for key generation, encapsulation/encryption, and decapsulation/decryption under a virtualized Ubuntu Linux environment.
- We present a comparative analysis of public key and ciphertext sizes, highlighting the storage and transmission overheads associated with transitioning to lattice-based cryptography.
- We discuss the implications of our experimental findings, offering insights into why Kyber exhibits superior key generation performance compared to RSA-2048 and how its operational behavior compares to ECDH-P256.

II. Background

A. Classical Cryptography

Modern cybersecurity relies heavily on public-key cryptographic systems to ensure confidentiality, authentication, integrity, and secure key exchange. Among the most widely deployed public-key algorithms are the Rivest-Shamir-Adleman (RSA) cryptosystem and Elliptic Curve Cryptography (ECC). RSA derives its security from the computational difficulty of factoring large composite integers, whereas ECC is based on the hardness of solving the Elliptic Curve Discrete Logarithm Problem (ECDLP). Under classical computing assumptions, both problems are considered computationally infeasible for sufficiently large key sizes, making these algorithms suitable for applications such as secure web communication, digital certificates, banking systems, cloud computing, and government infrastructure.

The security of RSA and ECC has remained reliable against attacks performed using conventional computers. However, their security assumptions are challenged by the emergence of large-scale quantum computers capable of executing quantum algorithms that efficiently solve the mathematical problems on which these cryptosystems rely.

B. Quantum Computing and Shor's Algorithm

Quantum computing introduces a fundamentally different computational model by utilizing quantum bits (qubits), which exploit the principles of superposition and entanglement. Unlike classical bits that exist only in the states 0 or 1, qubits can represent multiple states simultaneously, enabling certain computational problems to be solved more efficiently than on classical computers.

One of the most significant developments in quantum computing is Shor's Algorithm, proposed by Peter Shor in 1994. The algorithm provides a polynomial-time solution for integer factorization and discrete logarithm problems, both of which are computationally difficult for classical computers. Consequently, sufficiently capable fault-tolerant quantum computers could compromise RSA, ECC, and many currently deployed public-key cryptographic protocols. Although practical cryptographically relevant quantum computers are not yet available, continued advances in quantum hardware have accelerated research into quantum-resistant cryptographic solutions.

C. Post-Quantum Cryptography

Post-Quantum Cryptography (PQC) refers to the field of cryptographic algorithms designed to remain secure against attacks performed by both classical and quantum computers. Unlike Quantum Key Distribution (QKD), which requires specialized quantum communication hardware, PQC algorithms are implemented entirely in software and can be deployed on existing computing infrastructure.

The National Institute of Standards and Technology (NIST) initiated a multi-year standardization process to identify and evaluate quantum-resistant public-key algorithms suitable for widespread deployment. Candidate algorithms were assessed based on their security, implementation efficiency, interoperability, and resistance against known classical and quantum attacks. Among various approaches, lattice-based cryptography has emerged as one of the most practical and efficient solutions due to its strong security foundations and favorable computational performance.

One of the primary lattice-based mechanisms standardized by NIST is ML-KEM (formerly CRYSTALS-Kyber), which is designed for secure key establishment. In addition, ML-DSA (formerly CRYSTALS-Dilithium), Falcon, and SLH-DSA provide quantum-resistant digital signature capabilities.

D. NIST Standardization

Recognizing the long-term risks posed by quantum computing, NIST formally published its first post-quantum cryptography standards in 2024. These standards define algorithms intended to replace vulnerable public-key cryptographic schemes in future communication systems. ML-KEM was standardized as FIPS 203 for key encapsulation, ML-DSA as FIPS 204 for digital signatures, and SLH-DSA as FIPS 205 for stateless hash-based digital signatures.

The transition toward post-quantum cryptography is expected to occur gradually over the coming years as governments, industries, and technology providers begin integrating quantum-resistant algorithms into existing protocols such as TLS, VPNs, secure messaging platforms, and cloud services. Evaluating the computational performance of these algorithms on contemporary software platforms is therefore essential to understanding their practical deployment characteristics. This motivation forms the basis of the experimental analysis presented in this work.

III. Related Work

The growing threat posed by quantum computing has motivated extensive research into post-quantum cryptography (PQC), particularly in the evaluation of lattice-based cryptographic algorithms. Following the launch of the NIST Post-Quantum Cryptography Standardization Project, numerous

studies have focused on analyzing the performance, security, and implementation characteristics of candidate algorithms across both software and hardware platforms.

Several researchers have investigated the hardware implementation of lattice-based Key Encapsulation Mechanisms (KEMs) such as CRYSTALS-Kyber, Saber, and NTRU using FPGA platforms. These studies primarily evaluate metrics including execution time, hardware resource utilization, latency, and area-time product. Their results indicate that CRYSTALS-Kyber and Saber demonstrate superior computational efficiency compared to NTRU while maintaining comparable security levels. However, these investigations mainly target dedicated hardware implementations and provide limited insight into the practical software performance of these algorithms on conventional computing systems [1].

Other studies have examined the security implications of quantum computing on existing public-key cryptographic systems. These works discuss how Shor's Algorithm threatens widely deployed cryptographic protocols such as RSA and Elliptic Curve Cryptography (ECC), and they review emerging post-quantum alternatives including lattice-based, hash-based, and code-based cryptographic schemes. Although these surveys provide comprehensive theoretical discussions regarding quantum-resistant cryptography, they generally focus on security properties, algorithmic foundations, and migration considerations rather than detailed experimental benchmarking of standardized PQC algorithms. [4].

Software-based benchmarking has also been explored using the Open Quantum Safe (liboqs) library. Previous work evaluated several NIST Round-3 candidate algorithms by measuring key generation and TLS handshake performance on resource-constrained devices. These studies demonstrated the feasibility of deploying post-quantum cryptographic algorithms within existing communication protocols. Nevertheless, their primary emphasis was on embedded systems and TLS performance rather than a direct comparison of standardized PQC algorithms with widely deployed classical cryptographic techniques such as RSA-2048 and ECDH-P256 on a general-purpose computing environment [3].

The existing literature demonstrates significant progress in evaluating post-quantum cryptography from both hardware and protocol perspectives. However, relatively few studies provide a straightforward software-level performance comparison between standardized lattice-based KEMs and conventional public-key algorithms using an easily reproducible experimental setup. This work addresses that gap by benchmarking Kyber512, Kyber768, and Kyber1024 against RSA-2048 and ECDH-P256 within a controlled Ubuntu virtual machine environment using the Open Quantum Safe (liboqs) framework. The evaluation focuses on practical metrics including key generation time, encapsulation time, and public key size, providing insights into the deployment characteristics of post-quantum cryptography on contemporary software platforms.

IV. Methodology

This section describes the experimental setup, software environment, benchmarking methodology, and evaluation metrics used to compare the performance of post-quantum cryptographic algorithms against widely deployed classical public-key cryptographic schemes. The primary objective of the experimental evaluation is to analyze the practicality of standardized lattice-based Key Encapsulation Mechanisms (KEMs) for software deployment by measuring computational performance under a controlled environment.

A. Experimental Environment

All experiments were conducted on an Ubuntu 24.04.3 LTS virtual machine running inside Oracle VirtualBox. The virtual machine was configured with four virtual CPU cores and 8 GB of allocated memory. The underlying processor was an Intel Core i5-13450HX (13th Generation) operating on the x86_64 architecture. The virtual machine utilized a 40 GB virtual storage volume with approximately 27 GB of available disk space during experimentation.

The benchmarking environment was intentionally virtualized to provide a reproducible and isolated software platform while minimizing interference from unrelated background processes. Although virtualization introduces minor execution overhead, it provides a practical environment for evaluating software implementations of post-quantum cryptographic algorithms.

Table II summarizes the experimental platform.

B. Software Configuration

The benchmarking framework was implemented in Python 3.12.3. Post-quantum cryptographic operations were performed using the Open Quantum Safe (liboqs-python) library (Version 0.14.0), which provides software implementations of NIST-standardized post-quantum algorithms. Classical cryptographic operations were implemented using Python's Cryptography library.

Experimental data were processed using the Pandas library, while Matplotlib was employed to generate performance visualization graphs. All benchmark scripts were executed within an isolated Python virtual environment to ensure consistent package versions and reproducibility.

The evaluated post-quantum algorithms correspond to the NIST-standardized Module-Lattice-Based Key Encapsulation Mechanism (ML-KEM), formerly known as CRYSTALS-Kyber. Three standardized security parameter sets were evaluated:

- ML-KEM-512 (Kyber512)
- ML-KEM-768 (Kyber768)
- ML-KEM-1024 (Kyber1024)

For comparison with currently deployed public-key cryptographic systems, the following classical algorithms were also evaluated:

- RSA-2048
- ECDH using the NIST P-256 elliptic curve

TABLE I
COMPARISON OF EXISTING STUDIES WITH THE PROPOSED WORK

Reference	Evaluated Algorithms	Platform	Focus	Scope / Limitation
Basu et al. [6]	Kyber, Saber, NTRU	FPGA	Hardware Benchmark	Dedicated hardware; software metrics not evaluated.
Bernstein et al. [7]	RSA, ECC, PQC	Analytical	Security Analysis	Theoretical survey; empirical runtime omitted.
Paquin et al. [8]	Kyber, Dilithium	Embedded Devices	TLS Benchmark	Microcontroller-focused; no direct x86 VM baselines.
This Work	**Kyber (512/768/1024), RSA-2048, ECDH-P256**	**Ubuntu VM**	**Software Benchmark**	**Focuses on VM software layer; native hardware excluded.**

TABLE II
EXPERIMENTAL SYSTEM SPECIFICATIONS

Operating System	Ubuntu 24.04.3 LTS
Architecture	x86_64
Processor	Intel Core i5-13450HX
Allocated CPU Cores	4
Allocated Memory	8 GB
Python Version	3.12.3
OpenSSL Version	3.0.13
Virtual Disk Size	40 GB

C. Benchmark Methodology

The benchmarking process was designed to evaluate the computational performance of both post-quantum and classical public-key cryptographic algorithms under identical software conditions. Each benchmark was executed for 1000 independent iterations to reduce measurement noise and obtain statistically stable average execution times.

For each Kyber parameter set, three cryptographic operations were measured:

- Key generation
- Key encapsulation
- Key decapsulation

For classical cryptography, key generation performance was measured for RSA-2048 and ECDH-P256 using equivalent benchmarking procedures. The execution time of every iteration was recorded using Python's high-resolution timing functions, after which the average execution time was computed.

The benchmark scripts exported the collected measurements into CSV files for subsequent statistical analysis and visualization.

D. Performance Metrics

The performance evaluation considered multiple quantitative metrics that influence the practical deployment of cryptographic algorithms.

- **Key Generation Time**: Measures the computational cost required to generate a public-private key pair.
- **Encapsulation Time**: Represents the time required to establish a shared secret using the Key Encapsulation Mechanism.
- **Decapsulation Time**: Measures the computational effort required by the receiver to recover the shared secret.
- **Public Key Size**: Indicates the storage and transmission overhead associated with distributing public keys.
- **Ciphertext Size**: Represents the communication overhead introduced during key encapsulation.
- **Shared Secret Size**: Reports the size of the generated symmetric secret established by the KEM.

These metrics collectively provide insight into the computational efficiency and communication overhead of post-quantum cryptographic algorithms compared with conventional public-key cryptography.

V. EXPERIMENTAL RESULTS

This section presents the benchmarking results obtained from the experimental evaluation of post-quantum and classical public-key cryptographic algorithms. The benchmark was executed for 1000 iterations per algorithm, and the average execution time was computed for each performance metric. The experimental results provide insights into the computational efficiency of ML-KEM (Kyber) compared with widely deployed classical public-key cryptographic algorithms.

A. Key Generation Benchmark

Table III summarizes the average key generation time measured for each evaluated algorithm.

TABLE III
AVERAGE KEY GENERATION TIME

Algorithm	Average Time (ms)
Kyber512	0.01505
Kyber768	0.01762
Kyber1024	0.02297
ECDH-P256	0.01346
RSA-2048	33.95681

The experimental results indicate that the three Kyber parameter sets exhibit very low key generation latency. Among the evaluated post-quantum algorithms, Kyber512 achieved the lowest average key generation time of 0.01505 ms, followed by Kyber768 and Kyber1024 with average execution times of 0.01762 ms and 0.02297 ms, respectively.

When compared with classical algorithms, ECDH-P256 demonstrated the fastest key generation performance with an average execution time of 0.01346 ms. RSA-2048 required substantially more computation, with an average execution time of 33.95681 ms. This indicates that RSA key generation is several orders of magnitude slower than both Kyber and ECDH under the evaluated software environment.

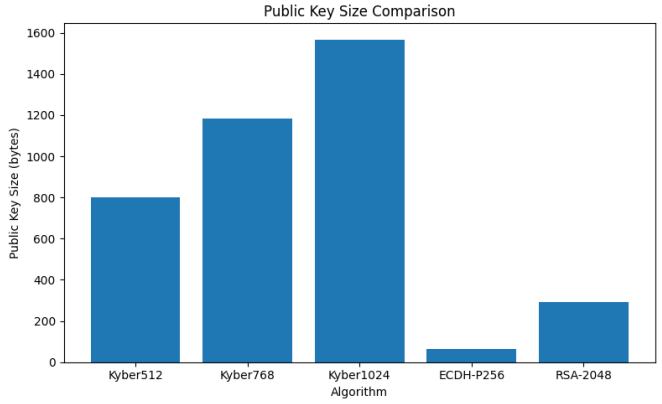


Fig. 1. Comparison of public key sizes for Kyber512, Kyber768, Kyber1024, ECDH-P256, and RSA-2048.

B. Encapsulation and Decapsulation Benchmarks

The encapsulation and decapsulation benchmarks evaluate the computational overhead required to establish and recover the shared symmetric key using ML-KEM. Encapsulation is executed by the sender using the recipient's public key, whereas decapsulation is performed by the recipient using their private key to recover the 32-byte shared secret from the ciphertext.

Table IV presents the measured execution times averaged over 1000 iterations for both operations.

TABLE IV
AVERAGE ENCAPSULATION AND DECAPSULATION TIME

Algorithm	Encapsulation (ms)	Decapsulation (ms)
Kyber512	0.01287	0.01810
Kyber768	0.01776	0.01915
Kyber1024	0.02171	0.02155

As shown in Table IV, both encapsulation and decapsulation complete in fractions of a millisecond across all security levels. Decapsulation requires slightly more execution time than encapsulation for Kyber512 and Kyber768 due to the additional verification step in the Fujisaki-Okamoto transform, but scales consistently, reaching approximately 0.02155 ms for Kyber1024. These latencies demonstrate that the computational cost of establishing quantum-safe symmetric keys in a virtualized software stack is negligible for modern network endpoints.

C. Public Key Size Comparison

Table V presents the public key sizes produced by each Kyber parameter set.

TABLE V
PUBLIC KEY SIZES

Algorithm	Public Key Size (Bytes)
Kyber512	800
Kyber768	1184
Kyber1024	1568

The public key size increases with the selected security level. Kyber512 produces an 800-byte public key, whereas Kyber768 and Kyber1024 require 1184 bytes and 1568 bytes, respectively. Although these key sizes are larger than those typically observed in classical elliptic-curve cryptography, they remain practical for deployment in modern communication systems.

The ciphertext sizes also increase proportionally with the security level, ranging from 768 bytes for Kyber512 to 1568 bytes for Kyber1024, while all parameter sets generate a 32-byte shared secret.

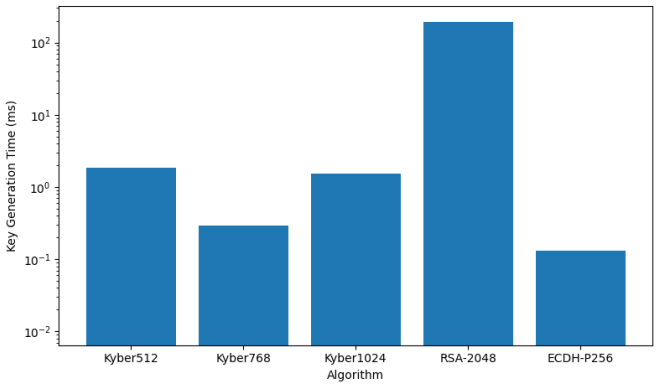


Fig. 2. Comparison of key generation time for Kyber512, Kyber768, Kyber1024, ECDH-P256, and RSA-2048 on a logarithmic scale.

D. Discussion of Results

The experimental evaluation demonstrates that the standardized ML-KEM algorithms achieve computational performance suitable for software deployment. Despite employing significantly larger public keys than classical cryptographic algorithms, the measured execution times remain extremely low.

Among the evaluated algorithms, Kyber512 provides the fastest overall performance while Kyber1024 offers the highest security level at the cost of slightly increased computational overhead. The measured increase in execution time across the three parameter sets is relatively small, indicating good scalability with increasing security levels.

A comparison with classical public-key cryptography reveals that the standardized ML-KEM algorithms (Kyber512, Kyber768, and Kyber1024) achieve key generation performance comparable to ECDH-P256 while significantly outperforming RSA-2048. In particular, RSA-2048 required approximately 34 ms on average, whereas all three ML-KEM parameter sets completed key generation in less than 0.03

ms. These findings indicate that standardized lattice-based key encapsulation mechanisms provide an efficient software-based alternative to conventional public-key cryptography while offering resistance against attacks from future cryptographically relevant quantum computers.

VI. Limitations

Although the experimental evaluation provides valuable insights into the software performance of post-quantum cryptographic algorithms, several limitations should be acknowledged. First, all experiments were conducted within a virtualized Ubuntu environment using Oracle VirtualBox. While virtualization offers an isolated and reproducible testing platform, the measured execution times may differ from those obtained on native hardware due to virtualization overhead.

Second, the scope of this work is limited to the evaluation of the NIST-standardized ML-KEM algorithms (Kyber512, Kyber768, and Kyber1024) and a comparison with classical public-key cryptographic algorithms, namely RSA-2048 and ECDH-P256. Other standardized post-quantum algorithms, including ML-DSA (CRYSTALS-Dilithium), Falcon, and SLH-DSA, were not considered in this study.

Furthermore, the evaluation primarily focused on computational performance metrics such as key generation time, encapsulation time, decapsulation time, and key sizes. Additional factors including memory consumption, processor utilization, power consumption, network latency, and real-world TLS handshake performance were outside the scope of this work. Consequently, the presented results should be interpreted as a software-level performance evaluation rather than a complete deployment analysis.

VII. Future Work

Future research can extend this work by evaluating additional NIST-standardized post-quantum cryptographic algorithms, particularly digital signature schemes such as ML-DSA (CRYSTALS-Dilithium), Falcon, and SLH-DSA. A comprehensive comparison between post-quantum key encapsulation mechanisms and digital signature algorithms would provide a broader understanding of their practical deployment characteristics.

Another promising direction is to perform benchmarking on native hardware platforms instead of virtualized environments to eliminate virtualization overhead and obtain more representative execution times. Future studies may also investigate the performance of post-quantum cryptography on embedded systems, mobile devices, Internet of Things (IoT) platforms, and cloud computing infrastructures.

In addition, integrating standardized post-quantum algorithms into real-world communication protocols such as Transport Layer Security (TLS), Virtual Private Networks (VPNs), and secure messaging applications would provide valuable insights into deployment feasibility. Finally, evaluating metrics such as memory utilization, energy consumption, network overhead, and scalability under different workloads would contribute to a more comprehensive assessment of post-quantum cryptographic systems.

VIII. Conclusion

The rapid advancement of quantum computing presents a significant challenge to the security assumptions of widely deployed public-key cryptographic algorithms such as RSA and Elliptic Curve Cryptography. The development of quantum-resistant cryptographic algorithms has therefore become an essential step toward ensuring long-term information security. In response to this challenge, the National Institute of Standards and Technology (NIST) standardized the ML-KEM family of lattice-based Key Encapsulation Mechanisms as a practical solution for secure key establishment in the post-quantum era.

This research presented a software-based performance evaluation of the standardized ML-KEM algorithms, namely ML-KEM-512 (Kyber512), ML-KEM-768 (Kyber768), and ML-KEM-1024 (Kyber1024), and compared their performance with the widely deployed classical algorithms RSA-2048 and ECDH-P256. The evaluation was conducted using the Open Quantum Safe (liboqs-python) framework within a controlled Ubuntu virtual machine environment.

The experimental results demonstrated that the evaluated ML-KEM algorithms achieve exceptionally low execution times for key generation, encapsulation, and decapsulation while maintaining practical communication overhead. In particular, the results showed that ML-KEM demonstrated substantially lower key-generation latency than RSA-2048 under the evaluated software environment. Although higher security parameter sets introduce larger public keys and ciphertext sizes, the additional computational overhead remains relatively small, indicating the practical suitability of standardized lattice-based cryptography for software deployment.

Overall, the findings support the growing adoption of standardized post-quantum cryptographic algorithms as organizations prepare for the transition toward quantum-resistant security infrastructures. While further evaluation under diverse deployment scenarios remains necessary, the results presented in this work demonstrate that ML-KEM provides an efficient and practical foundation for future secure communication systems.

References

[1] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," Federal Information Processing Standards Publication (FIPS) 203, Aug. 2024. doi: 10.6028/NIST.FIPS.203.

[2] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM J. Comput.*, vol. 26, no. 5, pp. 1484–1509, Oct. 1997. doi: 10.1137/S0097539795293172.

[3] Open Quantum Safe, "liboqs: C library for prototyping and experimenting with quantum-resistant cryptography," Open Quantum Safe Project, 2026. [Online]. Available: https://github.com/open-quantum-safe/liboqs.

[4] G. Alagic *et al.*, "Status Report on the Fourth Round of the NIST Post-Quantum Cryptography Standardization Process," Natl. Inst. Stand. Technol., Gaithersburg, MD, USA, NIST IR 8545, Mar. 2025. doi: 10.6028/NIST.IR.8545.

[5] R. Avanzi *et al.*, "CRYSTALS-Kyber Algorithm Specifications and Supporting Documentation (Round 3)," 2020. [Online]. Available: https://pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.

[6] K. Basu, D. Soni, M. Nabeel, and R. Karri, "NIST Post-Quantum Cryptography- A Hardware Evaluation Study," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 10, pp. 4064–4077, 2021.

[7] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017.
[8] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking Post-Quantum Cryptography in TLS," in *Post-Quantum Cryptography (PQCrypto 2020)*, ser. Lecture Notes in Computer Science, vol. 12100. Springer, 2020, pp. 72–91.