Investigating Post-Quantum Cryptography to Secure Transmitted Data via Mobile Communication

Rongjie Zhou , Huaqun Guo * , Francis E C Teo

Posted Date: 26 January 2026

doi: 10.20944/preprints202601.1876.v1

Keywords: post-quantum cryptography; mobile communication; 5G; Kyber; Dilithium; Falcon; SPHINCS+; performance metric; quantum attack; quantum computing

 doi:10.20944/preprints202601.1876.v1

Article

Investigating Post-Quantum Cryptography to Secure Transmitted Data via Mobile Communication

Rongjie Zhou [1], Huaqun Guo [2,*] and Francis E C Teo [3]

[1] Singapore Institute of Technology & WizVision Pte Ltd, Singapore
[2] Singapore Institute of Technology, Singapore
[3] WizVision Pte Ltd, Singapore
* Correspondence: huaqun.guo@singaporetech.edu.sg

Abstract

The advent of quantum computing poses significant challenges to traditional cryptographic systems, threatening the confidentiality, integrity and authenticity of digital communications. This paper investigates the integration of Post-Quantum Cryptography (PQC) algorithms into mobile communication systems to address these challenges. The study focuses on evaluating key PQC algorithms shortlisted by the National Institute of Standards and Technology (NIST), including CRYSTALS-Kyber, CRYSTALS-Dilithium, Falcon and SPHINCS+, within the context of 5G and future mobile network architectures. The research encompasses the design and implementation of an experimental framework involving mobile devices, servers, and cloud-based infrastructure to simulate real-world communication scenarios. Performance metrics such as key generation time, signature generation, encryption and decryption speed, and resource consumption were analyzed across various devices to identify algorithms suitable for mobile environments. The findings reveal that lattice-based algorithms, such as Kyber and Dilithium, offer a promising balance between security and efficiency, making them ideal for resource-constrained devices. In contrast, hash-based algorithms like SPHINCS+ exhibit higher computational demands, limiting their practicality in certain applications. This work highlights the importance of algorithm selection and hardware optimization in ensuring secure and efficient communications in the quantum era. By integrating theoretical advancements in PQC with practical applications, this research lays the foundation for quantum-resistant security in mobile networks, ensuring secure and future-ready digital communications.

Keywords: post-quantum cryptography; mobile communication; 5G; Kyber; Dilithium; Falcon; SPHINCS+; performance metric; quantum attack; quantum computing

1. Introduction

Modern secure communication relies on two complementary families of cryptography. Symmetric cryptography uses a single shared key to both encrypt and decrypt data; algorithms such as Advanced Encryption Standard (AES) are fast and well-suited to protect large volumes of information [1]. The long-standing challenge with symmetric schemes is key distributions where both parties must somehow agree on the same secret key without exposing it to eavesdroppers [2]. Asymmetric (public-key) cryptography addresses this by using a mathematically related key pair, a public key that can be shared openly and a private key that remains confidential. Schemes such as RSA (Rivest-Shamir-Adleman) and ECC (Elliptic Curve Cryptography) enable authenticated key establishment and digital signatures, which provide strong guarantees of data origin and integrity [3–6]. In practice, the 5G communication combines these families within protocols such as TLS (Transport Layer Security) [7,8]. During the handshake, the client verifies the server's identity by validating a certificate and runs a public-key key-agreement (commonly ECDHE (Elliptic Curve Diffie-Hellman Ephemeral)) to derive a fresh shared secret [9]. Once established, that secret drives

 doi:10.20944/preprints202601.1876.v1

symmetric authenticated encryption, typically an AEAD (Authenticated Encryption with Associated Data) mode like AES-GCM (Advanced Encryption Standard-Galois/Counter Mode), provides confidentiality and integrity for application data at high speed [1]. Digital signatures with Elliptic Curve Digital Signature Algorithm (ECDSA) are used where non-repudiation or end-to-end authenticity is required.

However, a threat model, "Store now, Decrypt Later", involves a passive but well-resourced bad actors who can eavesdrop and record encrypted traffic today, and store those ciphertexts and handshakes for years. They may not be able to break the cryptography now but are assumed to gain access in the future to large-scale quantum computation together with the use of quantum algorithms such as Shor's or Grover's algorithms that would break today's widely deployed public-key cryptography [10], such as RSA and ECDHE. Table 1 lists the minimum qubits needed to attack public-key cryptography [11]. Once such capability exists, the bad actors can retroactively derive past session keys from recorded handshakes and decrypt archived application data and compromise long-term confidentiality of messages that appeared secure at the time of transmission. This model is especially relevant to mobile applications that exchange personal or sensitive records over TLS.

Table 1. Minimum Qubits needed to Attack Public-Key Cryptography.

Equivalent (classical) bit security	Minimum qubits needed to attack RSA, DSA	Minimum qubits needed to attack ECDSA and similar ECC schemes
112	4098	2042
128	6146	2330
192	15362	3484

For symmetric cryptography, there is no known quantum algorithm that fully breaks standard primitives, instead, Grover's algorithm yields a quadratic speedup [12] for brute-force key search, effectively halving the security strength of a given key size, e.g., the security of AES-128 with a key length of 128 bits in the post-quantum era will be the same as that of AES-64 with a key length of 64 bits in the pre-quantum era [2]. In practice, this is mitigated by choosing larger symmetric keys like AES-256 instead of AES-128 to maintain equivalent security of AES-128 in the post-quantum era. These two results frame the risk model used in this paper where public-key mechanisms face eventual obsolescence under quantum scaling, whereas symmetric encryption remains viable with parameter adjustments.

Post-Quantum Cryptography (PQC) offers proactive defense. Developed to secure communications against both classical and quantum threats. PQC algorithms leverage mathematical constructs resistant to known quantum attacks. Their integration into modern networks, especially mobile environments like 5G and emerging 6G, is vital as these networks underpin critical applications such as telemedicine, autonomous vehicles, and IoT (Internet of Things) systems [13]. Ensuring robust cryptographic security in these contexts is not just a technical necessity but a foundational requirement for the digital age [14]. The NIST has been leading efforts in the standardization of PQC algorithms. In 2016, this initiative aimed to address the weaknesses of traditional cryptographic systems in the face of quantum computing. The project began with 69 algorithm submissions from global researchers [15], narrowing down through rigorous evaluations spanning multiple phases to identify cryptographic algorithms capable of withstanding quantum attacks. In 2022, NIST shortlisted several algorithms for key establishment and digital signatures, marking a critical step toward integrating quantum-resistant standards [14]. The criteria for selection included resistance to quantum attacks, computational efficiency, key size, and adaptability to diverse environments which is a pivotal consideration for mobile networks constrained by bandwidth and computational resources [9]. In 2024, NIST Releases First 3 Finalized Post-Quantum Encryption Standards as the first completed standards from NIST's PQC standardization project. In these three standards, CRYSTALS-Kyber algorithm has been renamed ML-KEM, short for Module-Lattice-Based Key-Encapsulation Mechanism, CRYSTALS-Dilithium algorithm has been renamed

 doi:10.20944/preprints202601.1876.v1

ML-DSA, short for Module-Lattice-Based Digital Signature Algorithm, SPHINCS+ algorithm has been renamed SLH-DSA, short for Stateless Hash-Based Digital Signature Algorithm [16].

Unlike servers or workstations, smartphones operate under tight latency expectations, dynamic CPU frequency scaling, and limited thermal headroom. Foreground interactions must feel instantaneous, background services and the Operating System's scheduler compete for cores, and prolonged cryptographic bursts can heat the device and trigger throttling that slows subsequent operations. Post-quantum algorithms often use larger keys, ciphertexts, or signatures, which increase payload sizes and can affect latency in apps that make many frequent API (Application Programming Interface) calls. For these reasons, the evaluation in this paper tracks three main aspects that directly shape user experience and energy use on actual devices:

- The wall-clock duration of each cryptographic operation (what the user "feels").
- The CPU time consumed (work done irrespective of scheduling noise).
- The temperature delta reported during controlled runs (a proxy for thermal stress and throttling risk)

By grounding the study in these device-centric constraints, later results can show practical choices an Android client must make when adopting Kyber for shared secrets and a suitable PQC signature scheme for message authenticity.

Shor's algorithm and "store now, decrypt later" endanger data encrypted under current cryptographic standards, including sensitive data such as personal, financial, medical, and other information over 5G and future 6G communication [17]. Hence, the objective and scope of this paper is to examine whether post-quantum cryptographic primitives can be integrated on the mobile client in a way that is both practical and measurable. With reference to the NIST's PQC Standardization Process [18,19], this study addresses two research questions that follow.

RQ1: Which post-quantum algorithms are practically usable on commodity smartphones when measured by operation duration, CPU time, and thermal impact.

RQ2 asks how Kyber, for key establishment, and signature algorithms like, Dilithium, Falcon, and SPHINCS+ compare with classical RSA/ECDHE/ECDSA under an identical message-layer workflow.

Despite the growing interest in PQC, there is limited device-centric evidence that implements both a PQC KEM and signature in a single mobile workflow, instruments operations on-device (duration, CPU time, temperature, payload size), and compares multiple NIST-endorsed schemes against classical baselines under identical conditions. This research gap makes it hard for mobile engineers to judge feasibility and cost on real phones. Hence, this study addresses the gap in existing PQC literature. Although many PQC algorithms have been standardized by NIST, there is limited empirical research evaluating their performance on real mobile devices in end-to-end communication workflows. Current studies focus heavily on algorithm design, server-side performance, or simulated benchmarking on desktop class operating systems rather than practical deployment on smartphones. This paper bridges that gap by implementing four NIST-selected PQC algorithms (Kyber, Dilithium, Falcon, and SPHINCS+) on Android devices and evaluating them through an application-level message exchange workflow. The work measures key generation time, signing/verification duration, KEM encapsulation/decapsulation, CPU load, and thermal behaviours that directly affect mobile feasibility but are rarely analyzed in prior studies.

This paper is organized as follows. Section 1 overviews the research background, presents the research objectives and scope, and describes research questions and research gaps. Session 3 presents the advancing Quantum capabilities, related research work and industry adoption. Session 3 presents the PQC algorithm selection, understanding of the chosen algorithms, impact of PQC on mobile clients, security level and key lengths of selected PQCs, and how a Kyber-style KEM session works. Session 4 describes experiment design in details, including setup overview, role of the HTTPS server, experiment workflow, client communication & handshake, exchanging encrypted messages, data collection and performance metrics. Session 5 presents KEM performance, digital signature performance, hardware impacts to the performance and performance results. Session 6 discusses the

 doi:10.20944/preprints202601.1876.v1

findings, while Session 7 presents future work. The research challenges are presented in Session 8. Finally, Session 9 concludes the paper.

2. Related Work

This session presents the advancing quantum capabilities, related research work and Industry adoption of PQC.

2.1. Advancing Quantum Capabilities

As outlined in the Background, Shor's algorithm threatens RSA/ECC once large, fault-tolerant quantum machines exist, while Grover's algorithm reduces the strength of symmetric keys. Recent milestones in quantum hardware underscore the urgency of evaluating post-quantum options. Google's Willow processor, a 105-qubit superconducting chip, demonstrates steady progress in qubit count and control fidelity, pushing the scale at which error-mitigation and calibration can keep circuits coherent [20].

In parallel, IBM reported running quantum computers at about 100-qubit circuits with about 3,000 gates on problems without a predetermined reference solution, suggesting regimes beyond practical brute-force classical simulation and requiring statistical or reduced-model checks rather than exact simulators [21]. IBM has introduced the IBM Condor, a 1,121-qubit quantum processor [21]. Based on their roadmap, Blue Jay, a system will be capable of executing 1 billion gates across 2,000 qubits by 2033 [21].

These advances bring both promise and risk. While quantum systems may unlock new computational capabilities, they also threaten foundations of modern public-key cryptography. Many studies and experiments [22–26] on quantum advancements, present an imminent threat to the confidentiality, integrity and authenticity of digital communications. As quantum computing is advancing rapidly, breaking current cryptography standards is only a matter of time [20]. This highlights the need for cryptographic systems resistant to quantum attacks [4,27]. This motivates timely migration toward quantum-resistant (post-quantum) schemes while such hardware capabilities continue to scale.

2.2. Related Work

In our previous work [28], the survey study provided comprehensive insights into the possible candidates for implementation in mobile communication environments, highlighting their strengths and limitations. Specifically, Kyber and Dilithium emerged as leading candidates due to their computational efficiency and balanced security features, making them particularly well-suited for resource-constrained mobile devices. The survey further identified critical challenges, such as optimizing hardware performance, mitigating latency in real-time communication scenarios, and addressing the resource demands of PQC algorithms.

Moreover, the survey included an in-depth analysis of algorithmic performance across various mobile network contexts, including 5G and 6G architectures. It outlined the importance of selecting algorithms that align with the unique requirements of mobile environments, such as low-latency encryption and efficient key management. These findings laid the foundation for this paper, which expands on the survey's results by implementing and rigorously testing PQC algorithms within a practical framework. Through the analysis of computational requirements, key sizes, and operational metrics, this study aims to validate the feasibility of integrating quantum-resistant cryptography into both existing and future mobile network infrastructures. Additionally, the research contributes empirical data and actionable insights into overcoming the challenges of deploying PQC solutions in real-world scenarios, further advancing the field toward secure quantum-era communications.

Studies, such as [2], have established a foundational understanding of the challenges posed by quantum computing to security protocols. Our study builds upon this by exploring recent advancements and practical applications of PQC algorithms in mobile communications. The

 doi:10.20944/preprints202601.1876.v1

evolution of quantum computing has brought to light the weaknesses in current cryptographic practices, including widely used data encryption standards and TLS/SSL implementations. These weaknesses necessitate a shift to more robust solutions.

Despite the security framework provided by TLS, the advent of quantum computing and the discovery of Shor's Algorithm pose significant challenges to the efficacy of TLS [29]. Shor's Algorithm, executed on a sufficiently powerful quantum computer, with thousands of logical qubits, could potentially break the asymmetric encryption algorithms that form the basis of TLS security.

With the continuous evolution of mobile communication technologies, the security framework must also advance to counteract emerging threats. Traditional cryptographic methods, once deemed secure, are now facing challenges in ensuring the integrity and confidentiality of data against quantum attacks. This situation highlights the importance of exploring PQC algorithms, which are designed to withstand the computational capabilities of quantum computers. The unique constraints of mobile networks, such as limited bandwidth, variable latency, and diverse device capabilities, necessitate a careful selection of PQC algorithms that are not only secure but also efficient and practical for mobile environments.

Another study [8] highlighted the transition to post-quantum TLS introduces additional challenges. Their analysis on some popular Android applications revealed that apps frequently establish numerous TLS connections without fully utilizing optimizations like session resumption, leading to significant computational and bandwidth demands. These findings emphasize the need for careful integration of PQC mechanisms, such as Kyber and Dilithium, to address the larger handshake data sizes and ensure efficiency in resource-constrained mobile environments.

Integrating PQC into mobile communication systems presents unique challenges and opportunities. Mobile devices, constrained by processing power and battery life, require cryptographic solutions that balance security and efficiency [30,31]. Additionally, mobile networks like 5G and 6G demand low-latency encryption protocols to support real-time applications such as autonomous vehicles and telemedicine [29].

2.3. Industry Adoption

As the demand for quantum-resistant solutions grows, leading organizations have begun taking proactive steps to implement PQC in their systems. For instance, Amazon Web Services (AWS) has updated its security infrastructure by incorporating Kyber into core services such as the AWS Key Management Service (KMS), AWS Certificate Manager (ACM), Secrets Manager TLS endpoints, and Secure File Transfer Protocol (SFTP) services. This strategic move extends quantum-safe security protocols to a significant number of mobile applications that rely on AWS for backend services, effectively broadening the reach of quantum-resistant technologies [27,32].

Similarly, IBM has integrated Kyber and Dilithium from the CRYSTALS suite into its IBM z16 System, a flagship hardware platform designed to deliver quantum-safe capabilities. IBM's hardware often underpins telecom network infrastructure, making this adoption highly influential for shaping security standards in future 5G and 6G architectures [33].

Cloudflare, a leader in internet infrastructure, has also taken proactive steps to address the quantum threat by implementing Kyber for key agreement in its TLS 1.3 traffic, including HTTP/3. Their hybrid approach integrates classical cryptographic methods with post-quantum cryptography to balance performance and security. This ensures enhanced protection for data in transit, guarding against potential future quantum-based attacks. Cloudflare's adoption of PQC not only strengthens its infrastructure but also serves as a model for other web service providers, accelerating the adoption of quantum-safe web security standards [34,35].

These examples underscore how theoretical advances in PQC are being translated into practical implementations, driving innovation across key industries. By addressing challenges such as computational overhead and latency, these efforts provide a pathway for integrating PQC into diverse applications, including mobile networks. As quantum technologies continue to evolve, the adoption of PQC will be essential for safeguarding critical digital infrastructure in the quantum era.

 doi:10.20944/preprints202601.1876.v1

3. PQC Algorithm Selection

Session 3 describes the PQC algorithms selected, impact of PQC on mobile clients, and how a Kyber-style KEM session works.

3.1. Understanding PQC Algorithms Selected

The shortlisted algorithms are divided into two primary categories: key encapsulation mechanisms (KEMs) and digital signature algorithms. Each shortlisted algorithm addresses unique aspects of quantum-resistant cryptography.

KEMs are essential for secure key exchange, particularly in protocols like TLS. Among the finalists, CRYSTALS-Kyber emerged as a leading candidate. This lattice-based algorithm stands out for its compact keys and ciphertexts, ensuring both security and efficiency. For example, Kyber's key sizes are significantly smaller than RSA, reducing bandwidth consumption in mobile networks [14], which is a critical factor for 5G and 6G environments.

Digital signature algorithms ensure the authenticity and integrity of digital communications. The NIST shortlist includes CRYSTALS-Dilithium, Falcon, and SPHINCS+. Each algorithm addresses unique needs. CRYSTALS-Dilithium is a lattice-based signature scheme offering balanced performance relatively small signatures and fast verification time [14] and is well-suited for real-time applications like encrypted. Falcon, another lattice-based algorithm optimized for small key sizes and signatures, making it suitable for constrained devices [14]. Lastly, SPHINCS+, a stateless hash-based signature scheme, provides strong security guarantees at the cost of larger signatures and higher computational requirements. Its applications are best suited to scenarios where security outweighs resource efficiency [2].

Each algorithm brings trade-offs. Kyber excels in scenarios demanding low latency, such as video conferencing or real-time data transmission. Dilithium and Falcon balance security with computational efficiency, making them suitable for mobile devices and IoT systems. In contrast, SPHINCS+ may find its niche in applications requiring unparalleled security, such as protecting long-term sensitive data.

3.1.1. How Kyber Works

Kyber is built on a fundamental hard problem of Module-LWE (Learning With Errors) in lattice cryptography. Given a matrix A and a vector $t = A \cdot s + e$, where s is a secret vector and e is a small noise vector. The goal is to recover s. With the added noise, even quantum computers cannot solve this efficiently.

- Simplified Kyber [36,37]

(1) Use Kyber768 domain parameters to show how big or how small parameters are:

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

(2) Kyber Key Generation

Alice does

1) Select $A \in_R R_q^{kxk}$, $s \in_R S_{\eta_1}^k$, and $e \in_R S_{\eta_2}^k$
   A: a public random matrix
   e: a small noise vector
   s: Alice's private key
2) Compute $t = As + e$.
3) Alice's public key is (A, t) and private key is s.
   Note: Computing s from (A, t) is an instance of Module-LWE problem.

(3) Kyber Encryption

 doi:10.20944/preprints202601.1876.v1

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

1) Obtain an authentic copy of Alice's public key (A, t)
2) Randomly select $r \in_R S_{\eta_1}^k$, $e_1 \in_R S_{\eta_2}^k$, and $e_2 \in_R S_{\eta_2}$
   The encryption procedure uses a randomizer polynomial vector r and an error polynomial vector e_1. These polynomial vectors are freshly generated for every encryption. Additionally, an error polynomial e_2 is needed. Three polynomials r e_1, and e_2 are completely random and small.
3) Computer

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

$$v = t^T r + e_2 + \left\lceil \frac{q}{2} \cdot \frac{1}{1} \right\rfloor m$$

4) Cypher $c = (u, v)$, $c \in R_q^k \times R_q$, is sent to Alice.

(4) Kyber Decryption

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

1) Computer $m = Round_q(v - s^T u)$. Alice uses her private key s.

- A Simple Example to Show how Kyber Works

Because we will multiply and add polynomials, we also need a modulus so that their degree would not become too big for us to handle. We use $f = x^4 + 1$ as the polynomial modulus. It is not important why f looks the way it does. The important fact is that by taking a polynomial modulo f, we guarantee that their degree (highest exponent) will be smaller than 4. Below all calculations are implicitly done modulo q (on coefficients) and f (on polynomials).

From the polynomial modulus $f = x^4 + 1$, we obtain $x^4 = -1$.

(1) Parameters

We use the small parameters as an example to show how Kyber works. For actual implementations, we use the original parameters as above for the strong security as original design.

$$q = 17$$
$$n = 4$$
$$k = 2$$
$$\eta_1 = 2$$
$$\eta_2 = 2$$

(2) Kyber Key generation

Alice does

1) Alice selects her private key s

$$s = \begin{bmatrix} -x^3 + x^2 - x \\ x^3 + x \end{bmatrix}$$

4) Compute $t = As + e$.

i. Alice selects a public random matrix A

$$A = \begin{bmatrix} 16x^3 + 6x^2 + 6x + 10 & 9x^3 + 3x^2 + 6x + 5 \\ 3x^3 + 5x^2 + 9x + 2 & 3x^3 + 2x^2 + 6x + 10 \end{bmatrix}$$

ii. Alice selects a small noise vector e

$$e = \begin{bmatrix} x^2 \\ x^2 + x \end{bmatrix}$$

iii. Compute t

$$t = As + e = \begin{bmatrix} 16x^3 + 6x^2 + 6x + 10 & 9x^3 + 3x^2 + 6x + 5 \\ 3x^3 + 5x^2 + 9x + 2 & 3x^3 + 2x^2 + 6x + 10 \end{bmatrix} \cdot \begin{bmatrix} -x^3 + x^2 - x \\ x^3 + x \end{bmatrix} + \begin{bmatrix} x^2 \\ x^2 + x \end{bmatrix}$$

(a) First Calculation

$$(16x^3 + 6x^2 + 6x + 10)(-x^3 + x^2 - x)$$
$$= -16x^6 + 10x^5 - 16x^4 - 10x^3 + 4x^2 - 10x$$
$$= -16x^2x^4 + 10xx^4 - 16x^4 - 10x^3 + 4x^2 - 10x$$

Apply $x^4 = -1$

$$= -16x^2(-1) + 10x(-1) - 16(-1) - 10x^3 + 4x^2 - 10x$$

 doi:10.20944/preprints202601.1876.v1

$= -10x^3 + 20x^2 - 20x + 16$

Apply modulo $q = 17$

$= (-10 + 17)x^3 + (20 - 17)x^2 + (-20 + 34)x + 16$

$= 7x^3 + 3x^2 + 14x + 16$

(b) Second Calculation with applying $x^4 = -1$ and modulo $q = 17$

$(9x^3 + 3x^2 + 6x + 5)\ (x^3 + x) = 8x^3 + 14x^2 + 2x + 2$

(c) Third Calculation with applying $x^4 = -1$ and modulo q = 17

$(16x^3 + 6x^2 + 6x + 10)(-x^3 + x^2 - x) + (9x^3 + 3x^2 + 6x + 5)\ (x^3 + x) + x^2$

$= 7x^3 + 3x^2 + 14x + 16 + 8x^3 + 14x^2 + 2x + 2 + x^2$

$= 15x^3 + 18x^2 + 16x + 18$

Apply modulo q=17

$= 15x^3 + x^2 + 16x + 1$

(d) Fourth Calculation with applying $x^4 = -1$ and modulo $q = 17$

$(3x^3 + 5x^2 + 9x + 2)(-x^3 + x^2 - x) + (3x^3 + 2x^2 + 6x + 10)(x^3 + x) + (x^2 + x)$

$= 14x^3 + 9x + 15$

Thus, using $q = 17$ and $x^4 = -1$, we can get

$$t = \begin{bmatrix} 15x^3 + x^2 + 16x + 1 \\ 14x^3 + 9x + 15 \end{bmatrix}$$

Now a Kyber key pair for Alice has been obtained:

- Private key: s
- Public key: (A, t)

(3) Kyber Encapsulation (Encryption)

1) Bob generates a random secret message

Bob generates a random secret message 13, which has a binary representation of 1101.

So, the message encoded as binary polynomial is:

$$m = 1x^3 + 1x^2 + 0x^1 + 1x^0 = x^3 + x^2 + 1$$

2) Randomly select small polynomials e_1, e_2 and r:

$$r = \begin{bmatrix} x^3 + x^2 + x \\ x^3 - 1 \end{bmatrix}$$

$$e_1 = \begin{bmatrix} -x^2 + x \\ x \end{bmatrix}$$

$$e_2 = x^3 + x^2 - x$$

3) Compute

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

$$v = t^T r + e_2 + \left\lfloor \frac{q}{2} \cdot \frac{1}{1} \right\rfloor m$$

$$u = \begin{bmatrix} 16x^3 + 6x^2 + 6x + 10 & 9x^3 + 3x^2 + 6x + 5 \\ 3x^3 + 5x^2 + 9x + 2 & 3x^3 + 2x^2 + 6x + 10 \end{bmatrix}^T \cdot \begin{bmatrix} x^3 + x^2 + x \\ x^3 - 1 \end{bmatrix} + \begin{bmatrix} -x^2 + x \\ x \end{bmatrix}$$

$$v = \begin{bmatrix} 15x^3 + x^2 + 16x + 1 \\ 14x^3 + 9x + 15 \end{bmatrix}^T \cdot \begin{bmatrix} x^3 + x^2 + x \\ x^3 - 1 \end{bmatrix} + (x^3 + x^2 - x) + \left\lfloor \frac{17}{2} \cdot \frac{1}{1} \right\rfloor (x^3 + x^2 + 1)$$

Thus, using $q = 17$ and $x^4 = -1$, we can get

$$u = \begin{bmatrix} 4x^3 + 8x^2 + 9x + 12 \\ 4x^3 + 14x^2 + 3x \end{bmatrix}$$

$$v = 12x^3 + 15x^2 + 9x + 4$$

4) Send the cypher $c = (u, v)$ to Alice.

(4) Kyber Decapsulation (Decryption)

After receiving the cypher, Alice uses her private key s to compute

$$m = Round_q(v - s^T u).$$

1) compute a noisy result m.

$$m = v - s^T u$$

In this example, the calculation result is

$$m = 9x^3 + 9x^2 + 14x + 6$$

 doi:10.20944/preprints202601.1876.v1

2) Recovers the message → derives the shared secret.
Let the coefficient c within [-(q-1)/2, (q-1)2] = [-(17-1)/2, (17-1)2] = [-8, 8]. Hence, apply q = 17 to m so that its coefficient c within [-8, 8].

$$m = -7x^3 - 7x^2 - 3x + 6$$

Apply rounding function to polynomials m

$$Round_q(c) = \begin{cases} 0 & if -\frac{q}{4} < c < \frac{q}{4} \\ 1 & otherwise \end{cases} = \begin{cases} 0 & if -\frac{17}{4} < c < \frac{17}{4} \\ 1 & otherwise \end{cases}$$

We get the rounded polynomial m

$$m = 1x^3 + 1x^2 + 0x + 1$$

From m we can just read the bits of the original message, which are $(1101)_2 = (13)_{10}$. Thus, we have recovered the secret message which is 13.

3.1.2. How CRYSTALS-Dilithium Works

Dilithium is built on two fundamental hard problems of Module-SIS (Short Integer Solution) and Module-LWE. In Module-SIS, given a random matrix A, finding a short non-zero vector z such that $A{\cdot}z = 0$ is computationally difficult in high dimensions, and even quantum computers cannot efficiently solve this. In this algorithm, Dilithium can sign a message using a private key and verify a signature using the corresponding public key. The following description is simplified [38], while the more comprehensive version of Dilithium can be referred to the NIST standard [39,40].

1) **Key Generation:**
- Generate a random secret key s_1, s_2.
- Compute a public matrix A and a public key $t = A{\cdot}\, s_1 + s_2$

2) **Signing:**
- Pick a random value y.
- Compute a commitment $w = A{\cdot}y$.
- Derive a challenge c using a hash function: $c = H(w \,||\, message)$.
- Form the response: $z = y + c{\cdot}\, s_1$.
- Output the signature: (z, c).

3) **Verification:**
- Recompute $w' = A{\cdot}z - c{\cdot}t$.
- Hash $w' \,||\, message$ to recompute c': $c' = H(w' \,||\, message)$.
- If $c' == c$, the signature is valid.

3.1.3. How Falcon Works

Falcon's security relies on the hard problem of the Short Integer Solution (SIS) over NTRU lattices. It is easy to generate a private key (a "good" short basis for a lattice), while it is computationally difficult (even for quantum computers) to find this private key (this short basis) given only a "long" public basis (the public key). The following description is high-level [41], while the more comprehensive version of Falcon can be referred to [42].

1) Key generation
- The public key is a long basis of a q-ary lattice.
- The private key is a short basis of the same lattice.

2) Signing
In the signing procedure, the signer:
- generates a random value *salt*.

 doi:10.20944/preprints202601.1876.v1

- computes a target $c = H(message||salt)$, where H is a hash function sending an input to a random-looking point (on the grid).
- uses the known short basis (his private key) to compute a lattice point v close to the target c.
- outputs ($salt$, s), where $s = c - v$.

3) Verification

The verifier accepts the signature ($salt$, s) if and only if:

- The vector s is short.
- $H(message||salt) - s$ is a point on the lattice generated by his public key.

3.1.4. How SPHINCS+ Works

The following description is main idea of SPHINCS+ [43], while the more comprehensive version of SPHINCS+ can be referred to NIST standard [44].

Main idea of SPHINCS+:

- Employ a hypertree with a very large number of leaves.
- The leaves are signature key pairs for a one-time signature (OTS) scheme.
- The signer randomly selects one of these key pairs when signing a message.
- This algorithm has a limit on the total number of messages that can be signed since it needs to ensure that the probability of key reuse is negligibly small.

3.2. Impact of PQC on Mobile Clients

Mobile environments introduce unique constraints, including limited bandwidth, variable latency, and diverse device capabilities. Algorithms like Kyber and Dilithium address these challenges through compact keys, efficient computation, and robust security. Their integration into mobile networks ensures that next-generation technologies, from telemedicine to autonomous vehicles, remain secure even in the quantum era.

As these algorithms begin transitioning into the standardization phase, it is imperative to closely monitor and analyze their development. Standardization involves rigorous testing and evaluation to ensure that the algorithms meet high security and performance benchmarks. This phase also includes extensive public review and feedback, allowing for the identification and rectification of any potential vulnerabilities or inefficiencies.

This study also considers various performance metrics and practical implications, aiming to provide insights into how these algorithms can be effectively integrated into existing and future mobile network infrastructures. The evaluation is crucial for informing decisions about which PQC algorithms to adopt, ensuring that the transition to quantum-resistant cryptography is both secure and efficient in mobile communications.

3.3. Security Level and Key Lengths of Selected PQCs

Tables 2 and 3 list the concrete parameter sets. For Kyber we cover NIST levels 1/3/5 (Kyber512/768/1024). For signatures we include Dilithium2/3/5, Falcon512/1024, and representative SPHINCS+ variants across levels 1/3/5. SPHINCS+ comes in "s" (smaller signature) and "f" (faster signing) variants at the same security level.

Table 2. PQC KEM Algorithms.

Algorithm	**PQC - KEM**			
	Security Level	*Private key length*	*Public key length*	*Ciphertext length*
Kyber512	1	1632	800	768
Kyber768	3	2400	1184	1088
Kyber1024	5	3168	1568	1568

 doi:10.20944/preprints202601.1876.v1

Table 3. PQC Signature Algorithms.

Algorithm	**PQC Digital Signature Schemes**			
	Security Level	*Private key length*	*Public key length*	*Signature length*
Dilithium2	2	2528	1312	2420
Dilithium3	3	4000	1952	3293
Dilithium5	5	4864	2592	4595
Falcon512	1	1281	897	690
Falcon1024	5	2305	1793	1330
SPHINCS+-Haraka-128f	1	64	32	17088
SPHINCS+-Haraka-128s	1	64	32	7856
SPHINCS+-Haraka-192f	3	96	48	35664
SPHINCS+-Haraka-192s	3	96	48	16224
SPHINCS+-Haraka-256f	5	128	64	49856
SPHINCS+-Haraka-256s	5	128	64	29792
SPHINCS+-SHA256-128f	1	64	32	17088
SPHINCS+-SHA256-128s	1	64	32	7856
SPHINCS+-SHA256-192f	3	96	48	35664
SPHINCS+-SHA256-192s	3	96	48	16224
SPHINCS+-SHA256-256f	5	128	64	49856
SPHINCS+-SHA256-256s	5	128	64	29792
SPHINCS+-SHAKE256-128f	1	64	32	17088
SPHINCS+-SHAKE256-128s	1	64	32	7856
SPHINCS+-SHAKE256-192f	3	96	48	35664
SPHINCS+-SHAKE256-192s	3	96	48	16224
SPHINCS+-SHAKE256-256f	5	128	64	49856
SPHINCS+-SHAKE256-256s	5	128	64	29792

In this paper, “security level” follows NIST’s convention for PQC parameter sets. Level 1, Level 3, and Level 5 aim to be as hard to break as attacking AES-128, AES-192, and AES-256, respectively. For KEMs, the goal is what cryptographers call IND-CCA (Indistinguishability security under the Chosen Ciphertext Attack) security, which can be interpreted as “even if an attacker can poke at a decryption box and see how it behaves on many chosen inputs, they still cannot recover the secret shared key.” For signatures algorithms, the goal is EUF-CMA (Existential Unforgeability under Chosen-Message Attack) security, which just be interpreted as “even if an attacker can get you to sign lots of messages of their choosing, they still cannot forge a valid signature on any new message.” These levels are comparison targets that put very different math on a common “work factor” scale and they do not guarantee identical performance, and they are not a rule that you must also change your symmetric cipher. E.g. choosing a Level 5 KEM does not force you to move from AES-128 to AES-256.

3.4. How a Kyber-Style KEM Session Works

A post-quantum KEM such as Kyber, uses the Module-LWE problem to generate the key, plays the same role as classical ECDHE. They establish a shared secret used for symmetric encryption, but the mechanics differ [45]. The recipient first generates a public/secret key pair and publishes the public key. When a sender wants to start a session, it runs encapsulation against that public key: from fresh randomness it produces a ciphertext and, internally, a shared secret. The sender transmits only the ciphertext. The recipient runs decapsulation with its secret key to recover the same shared secret. Both sides then feed that value into a KDF (Key Derivative Function) and obtain AEAD keys for protecting messages.

The security levels of various key exchange algorithms are listed in Table 4.

 doi:10.20944/preprints202601.1876.v1

Table 4. Key Exchange Algorithm Security Levels.

Key Exchange Algorithm	Security Level
RSA	Weak against quantum attacks
DHE	Moderate (large keys needed)
ECDHE	Stronger than DHE with smaller keys
Kyber (PQC)	Quantum-resistant

4. Experiment Design

The details for experiment design are described in this session.

4.1. Setup Overview

This experiment is designed to rigorously and repeatedly test the integration and operational performance of PQC algorithms within a real-world mobile communication scenario. The objective is to evaluate the compatibility, efficiency, and security of PQC in the context of modern mobile networks such as 5G, with future implications for 6G systems. To do so, we will be implementing PQC algorithms directly on mobile devices as Android mobile devices do not natively support PQC key exchange or signature protocols, capturing performance metrics, and comparing results under identical communication workflows. The user interface and functions are designed to perform specific operations such as key exchange, key generation etc. to better capture these metrics for analysis.

Figure 1 provides a visual schematic of the experimental setup. The mobile phones (clients) are shown connecting to the HTTPS server (central node). The server is depicted as an EC2 instance with associated security measures and communication protocols. This visual representation aids in understanding the flow of data and the role of each component within the experimental framework.

The experimental setup consists of implementing several critical components, including mobile devices, a server infrastructure, and cryptographic protocols. Two 5G-enabled Android mobile devices serve as communication endpoints, both supporting PQC algorithms for encryption and decryption processes. These devices are configured to initiate secure HTTP communications with a server, employing PQC-based key exchange and digital signatures to establish a secure connection.

A Flask-based HTTPS server runs on an AWS EC2 instance, managing the key exchange protocols and routing encrypted traffic between the devices. The server is preconfigured to support a hybrid cryptographic model, where both traditional RSA/ECDSA algorithms and PQC algorithms (Kyber, Dilithium, Falcon, SPHINCS+) are used interchangeably for performance evaluation.

An Android application is developed to facilitate the collection of metrics such as CPU time, duration, temperature changes and key generation operations by allowing the selection of each algorithm with a click of a button. Each function at UI is independently executed for repeated data collection for metrics data for further analysis.

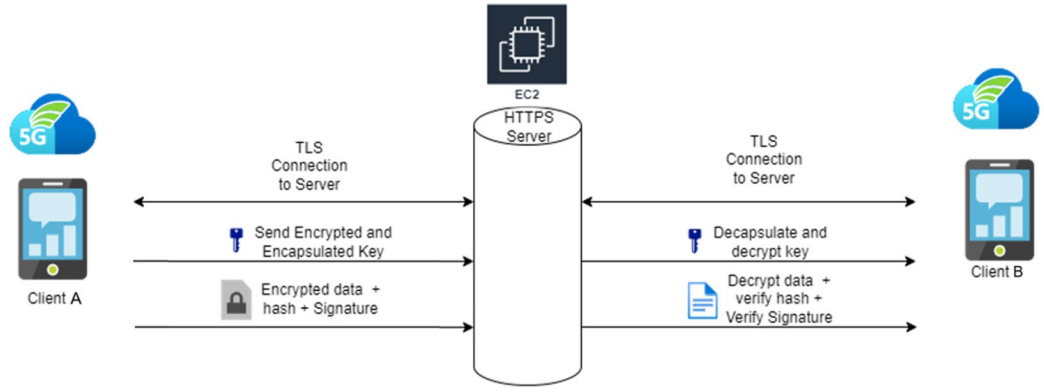


Figure 1. Experiment Architecture.

 doi:10.20944/preprints202601.1876.v1

4.2. Role of HTTPS Server

The HTTPS server plays a central role in this setup, handling PQC algorithms and managing secure communication channels with the clients. The server handles the key exchange protocols and manages the routing of traffic to the respective clients. This dual capability allows for comprehensive testing of both established and emerging cryptographic systems.

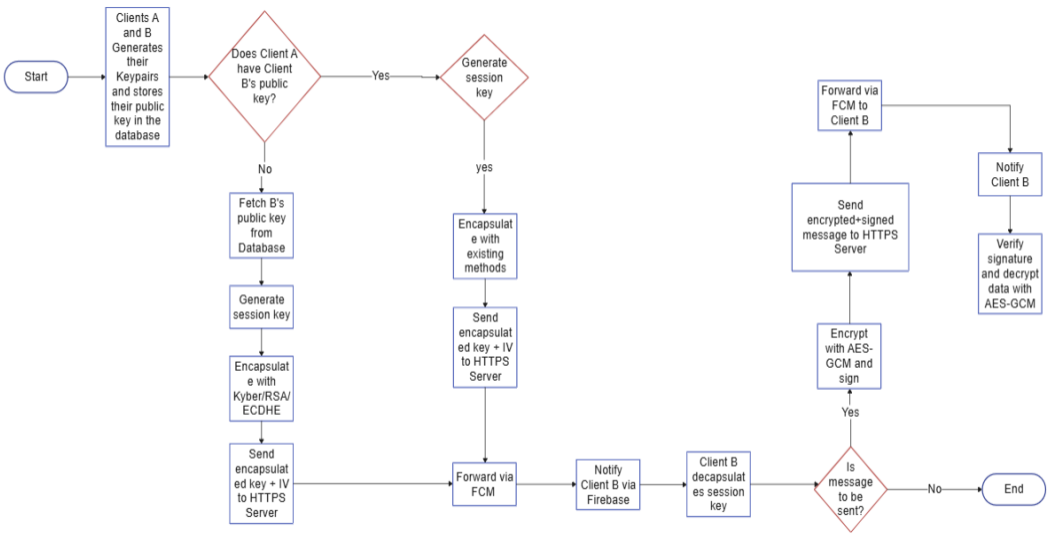
Public key storage is another crucial element of the experiment. In the absence of Certificate Authorities (CAs), public keys for the clients are stored in a centralized database on the server. During communication, the server retrieves the relevant public key from this database to establish a secure session. This method not only simplifies the infrastructure but also highlights the practicality of a CA-free public key management approach.

The experiment employs a blend of post-quantum algorithms, such as KEMs like Kyber and digital signatures like Dilithium, Falcon and SPHINCS+ alongside traditional cryptographic protocols such as RSA, ECDHE, and ECDSA. This hybrid approach ensures compatibility with existing systems while benchmarking the transition to quantum-safe methods.

Benchmarking the transition to quantum-safe methods is necessary because, in practical deployments, PQC algorithms will not replace classical algorithms overnight. Instead, organizations will operate in a hybrid phase where PQC and classical cryptography coexist. During this period, systems must maintain interoperability, ensure backward compatibility with legacy infrastructures, and avoid introducing new performance bottlenecks. The objective of benchmarking is therefore to quantify how PQC algorithms behave when integrated into existing mobile communication workflows, identify performance gaps relative to classical RSA, ECDHE or ECDSA, and determine whether their computational cost is acceptable for real-world usage on commodity smartphones. These insights help developers, operators, and policymakers understand the trade-offs involved in PQC adoption and guide decision-making during the gradual migration to quantum-resistant standards.

4.3. Experiment Workflow

The workflow (Figure 2) begins with initialization, where both clients begin by generating their cryptographic keypairs (public key and private key). This includes both PQC keys like Kyber for key encapsulation, and signature keys like Dilithium, Falcon, or SPHINCS+, as well as classical alternatives such as RSA and ECDSA. Each client registers with Firebase Cloud Messaging (FCM) to obtain a unique token for receiving notifications. The public keys and FCM tokens are then stored in Firebase Firestore database, making them accessible to other clients in the system. This setup creates a distributed public key infrastructure where clients can discover and retrieve each other's public keys.



 doi:10.20944/preprints202601.1876.v1

Figure 2. Experimental Workflow.

4.4. Client Communication & Handshake

Figure 3 illustrates the key exchange process on how the shared key is being shared between the communicating clients. Note that the TLS handshake processes are omitted from Figures 3 and 4.

1) For both clients to initiate communication, both Client A and Client B will generate their Kyber keypairs (pk, sk). The pk of these clients are then stored in the Firebase database.
2) Client A initiates a key exchange process. Client B's pk is retrieved from the Firebase database. Client A uses Client B's public key (pkB) to encapsulate the AES session key. The key exchange message is sent via HTTP POST to the server. The server then uses FCM to route the key exchange message to Client B.
   HTTP POST payload:

```
{
    "sender": Client A,
    "encapsulatedKey": c,
    "iv": IV,
    "clientId": recipient
}
```

3) Client B receives the key exchange message and begins decapsulation the session key using its own private key (skB).
4) The shared key is now stored and handled by the "SessionData.kt". Both parties can now use the shared key to encrypt and send data.

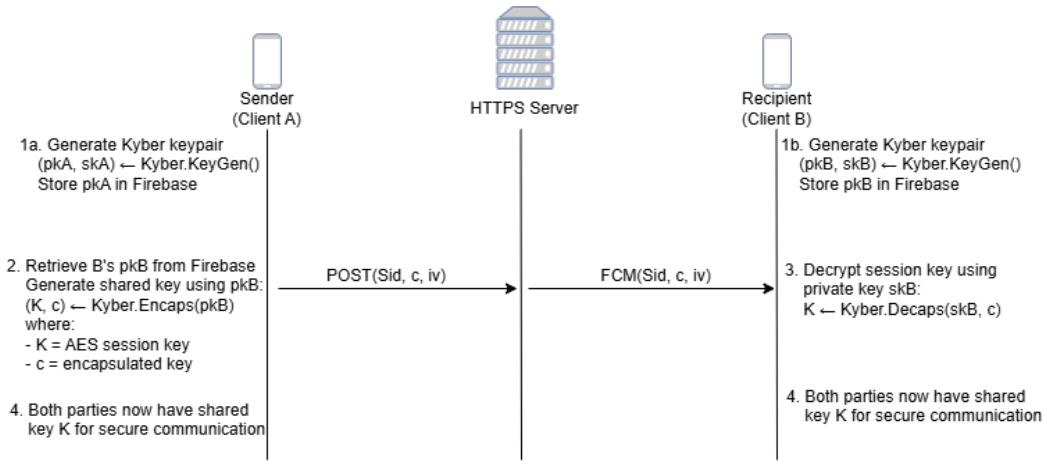


Figure 3. Key Exchange Implementation.

With the shared key (K) established, Figure 4 shows the process of sending data from Client A to Client B.

1) With the shared key established in Figure 3, the clients can start to exchange data messages. The defined message (M) by Client A is ready to be sent to the recipient, Client B.
2) The message (M) will be encrypted to provide confidentiality and integrity. Firstly, the random IV for AES-GCM mode is generated. The message is then encrypted using the shared key K:

$$\text{cipher} = \text{AES-GCM}(K, IV, M)$$

3) The hash value of encrypted message (h) is computed:

$$h = \text{SHA256}(\text{cipher})$$

The hash of the message (h) is then signed using Client A's Dilithium private key:

 doi:10.20944/preprints202601.1876.v1

$$\sigma = \text{Dilithium.Sign}(skA, h)$$

The HTTP POST payload is then sent to the server:
HTTP POST payload:

```
{
   "sender": Client A,
    "encryptedMessage": cipher,
    "iv": IV,
    "signature": σ,
    "algorithm": "dilithium"
}
```

Once the server receives the payload, it will forward it to the recipient Client B using FCM.

4) The recipient, Client B, receives the payload and begins processing. Firstly, the hash value h' of the encrypted message (cipher) is computed. Client A's public key (pkA) is then retrieved from the Firebase database to begin verification of the signature:

$$\text{valid} = \text{Dilithium.Verify}(pkA, h', \sigma)$$

5) If the signature is valid, the message will be decrypted with the shared key (K):

$$M' = \text{AES-GCM-Dec}(K, IV, \text{cipher})$$

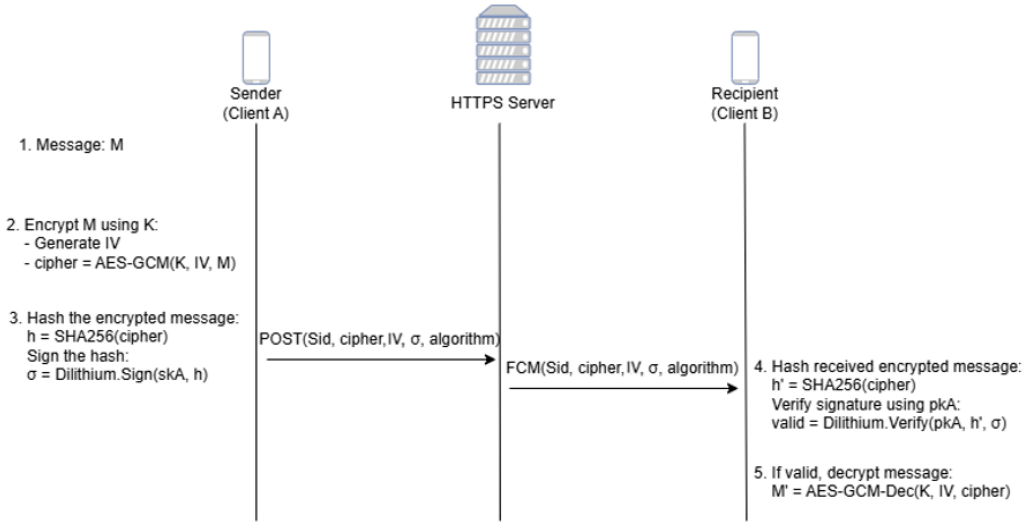


Figure 4. Secure Data Exchange Implementation.

4.5. Exchanging Encrypted Messages

Once the session is established, clients can exchange secure messages. When sending a message, the sender first encrypts the message using AES-GCM (Advanced Encryption Standard - Galois/Counter Mode) with the session key and IV. The encrypted message is then hashed using SHA-256, and this hash is signed using the sender's chosen signature algorithm (Dilithium, Falcon, SPHINCS+, RSA, or ECDSA). The encrypted message, signature, and signature method are packaged together and sent through the HTTPS server, which forwards it via FCM to the recipient.

When receiving a message, the client first verifies the signature using the sender's public key (retrieved from Firebase). This ensures the message's authenticity and integrity. If the signature is valid, the client proceeds to decrypt the message using the session key and IV. The decrypted message is then displayed to the user. If either the signature verification or decryption fails, an error message is shown instead. This process ensures both the confidentiality and authenticity of all communications.

 doi:10.20944/preprints202601.1876.v1

4.6. Data Collection

Firebase Firestore is selected as it provides easy integration with android application as well as real-time data synchronization, enabling seamless synchronization across the connected devices within milliseconds. This is a very crucial aspect for this research as there will be many database operations when the various operations are invoked.

The public keys, performance metrics and messages are stored in Firebase Firestore database. Google encrypts data at rest, but this does not prevent insider attacks or an attacker with sufficient access from reading unencrypted content. Firebase authentication and security rules provide important safeguards. However, if an attacker compromises an authenticated account, misconfigures rules, or obtains admin credentials, the attacker may gain read or write access to stored data. Under those threats, the public keys should not be affected since they should be treated as public; The messages are encrypted using shared session keys, and hence the attacker cannot know the plain text of messages; The performance metrics are stored temporarily for performance analysis, but they are not sensitive data. So, this paper uses Firebase Firestore and trusts Google's operational security, logging, insider-threat protections, and vulnerability-management processes, which is a reasonable assumption for this paper.

With the collected data, analysis can be done to determine possible candidates suitable for implementation into mobile communications.

4.7. Performance Metrics

The metrics are chosen as they directly impact user experience, battery life, and thermal throttling behaviour on mobile devices, making them the most relevant for PQC evaluation. The core performance metrics include:

1) "**Key Generation Duration**" that measures the time taken for each algorithm to generate cryptographic key pairs.

PQC algorithms often have significantly more complex key generation than classical cryptography. Since long key generation times negatively impact practical deplorability and energy consumption, this metric is selected.

2) "**Digital Signature Generation**" that measures the time taken for each algorithm to generate a digital signature.

Digital signatures are among the most frequently executed operations in mobile environments (authentication, secure updates, payments). It is good to evaluate different PQC signature schemes to show performance variation and decide which one is the most suitable. Hence this metric is selected.

3) "**Encrypt and Sign Duration**" that records the time to encrypt and sign the data hash with a digital signature algorithm.

Same reason as 2) to select this metric.

4) "**Verify and Decrypt Duration**" that records the time to verify the signature and decrypt the data.

Same reason as 2) to select this metric.

5) "**CPU Time**" that records and monitors the device's CPU utilization time during cryptographic operations to assess the computational load imposed by PQC algorithms. This is also the time CPU spends, excluding idle or wait times, actively processing instructions for a specific cryptographic operation.

It directly reflects computational cost and hence this metric is selected.

6) "**Temperature Delta**" that tracks the change in device temperature before and after a specific cryptographic operation. This metric provides an indirect measurement of resource consumption and potential thermal throttling.

It is unique to mobile environments. Increased temperature may affect mobile devices and their batteries.

 doi:10.20944/preprints202601.1876.v1

Other metrics, e.g., network latency is independent of cryptographic algorithms, are not prioritized.

These metrics are recorded for each algorithm under varying conditions to assess performance across different hardware and algorithm types.

5. Results

The evaluation of cryptographic algorithms was conducted to assess their performance in terms of key generation, signature generation, encapsulation/decapsulation, and operational tasks such as decryption and signature verification. The evaluation testing set consists of multiple repeated executions of all implemented cryptographic operations across both mobile devices. For each PQC algorithm and each classical routine, the Android application performed repeated operation cycles using the built-in instrumentation pipeline.

Each execution produced a structured measurement record that captured the operation duration (ms), CPU processing time (ms), device temperature delta (°C), and metadata such as algorithm name, operation type, device identifier, timestamp, and measured values. All measurements were stored as individual documents in Firebase Firestore, forming a complete dataset that aggregates results across multiple runs and both devices.

This results section presents the findings derived from this dataset and compares the performance of the PQC algorithms against commonly used classical cryptographic primitives such as RSA, ECDHE, and ECDSA, highlighting differences in computational behaviour, consistency across runs, and practical feasibility on mobile hardware.

5.1. KEM Performance

In terms of key exchange mechanisms, Kyber demonstrated promising key generation times and resource consumption in comparison with the classical algorithms like RSA and ECDHE. Mobile clients with better hardware are seen to affect this performance as well. Overall, this efficiency underscores its suitability for latency-sensitive applications such as real-time data encryption.

5.2. Digital Signature Performance

In the aspects for digital signature, Falcon and Dilithium performed comparably in signature generation and verification, offering a balance between speed and resource efficiency. These algorithms are well-suited for mobile networks where quick authentication is crucial. SPHINCS, on the other hand, while providing better security, exhibited longer signature generation times, limiting its application in mobile communications.

5.3. Hardware Impacts to the Performance

The experiment highlighted the significant role of hardware in cryptographic performance since most of the heavy lifting was processed on mobile devices. The MeowNet client of the model RedMagic 7s Pro, powered by a Snapdragon Gen 1 processor, consistently outperformed the Pixel 6a, demonstrating the importance of hardware optimization for resource-intensive algorithms like SPHINCS+. The Pixel 6a, powered by Google Tensor (5 nm), and despite its limitations, manages to efficiently handle PQC algorithm, Kyber and Dilithium, reinforcing their practicality in diverse hardware environments.

5.4. Performance Results

The subsections below will share the result findings and analysis based on the performance metrics defined.

 doi:10.20944/preprints202601.1876.v1

5.4.1. Key Generation Time

Figure 5 provides a visual presentation of the performance of each algorithm which is evaluated and compared with the different hardware clients, Pixel 6a and MeowNet.

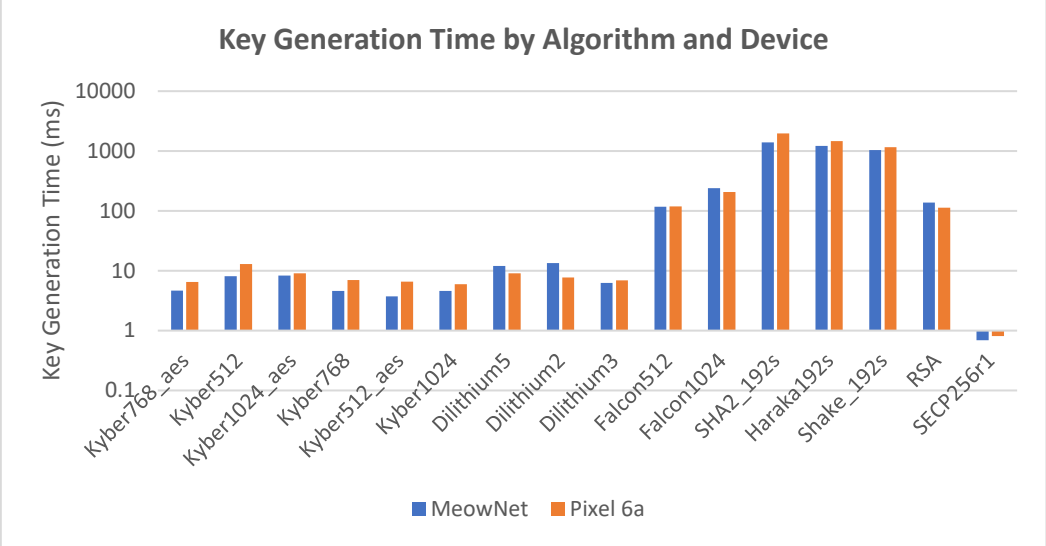


Figure 5. Key Generation Time in Log_{10} Scale by Algorithm and Device.

The key generation performance analysis highlights several critical observations regarding device capabilities and algorithm efficiency. The client, MeowNet, is equipped with a Snapdragon Gen 1 processor, demonstrates consistently faster key generation time than the Pixel 6a, Google Tensor (5 nm). This disparity underscores the importance of hardware optimization for cryptographic operations, as devices with higher processing power, like the MeowNet, are better suited for resource-intensive tasks such as frequent key generation or complex cryptographic protocols. This advantage positions the MeowNet as a more viable option for applications that demand high throughput or low latency.

SPHINCS+ variants, Sha2_192s, haraka192s and shake_192s, exhibits notably higher key generation time, which may pose challenges in scenarios requiring rapid or repeated key generation. This limitation could restrict its application in latency-sensitive use cases such as real-time secure messaging or dynamic session establishment.

Figure 6 further narrows the focus to PQC algorithms only. The results reaffirm Kyber's superiority in key generation efficiency, particularly on the client MeowNet. Dilithium and Falcon follow with moderate performance, striking a balance between security and computational requirements. However, SPHINCS+ remains the least efficient, further suggesting its limited applicability in mobile contexts. This analysis indicates Kyber's readiness for integration into resource-constrained environments, while Dilithium and Falcon may require selective optimization for specific use cases.

Subsequently, Figure 7 provide highlights on how PQC algorithms perform relative to classical algorithms. In this figure, the performance of PQC algorithms is compared to classical cryptographic algorithms like RSA and ECDHE. Classical algorithms like ECDHE and ECDSA continue to exhibit low key generation time, reflecting their extensive optimization for existing hardware. However, RSA shows slower performance, especially on less powerful devices such as the Pixel 6a. This highlights a key limitation of RSA in modern cryptographic implementations, particularly when compared to newer, more efficient algorithms. The contrast between these classical algorithms and PQC options underscores the potential advantages of transitioning to PQC for improved performance alongside quantum resistance.

 doi:10.20944/preprints202601.1876.v1

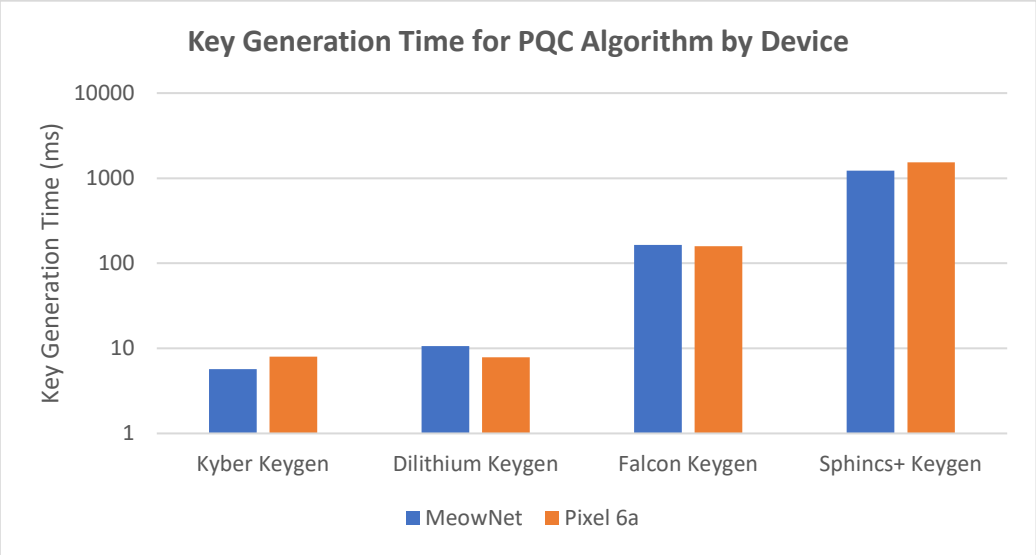


Figure 6. Key Generation Time in Log_{10} Scale for PQC Algorithm by Device.

Kyber demonstrates competitive performance, often matching or exceeding classical algorithms in efficiency. Notably, RSA lags behind both Kyber and Falcon in key generation times, especially on the Pixel 6a, illustrating its computational limitations in modern applications. These findings highlight the dual benefit of PQC algorithms: quantum resilience and comparable, if not superior, performance to classical algorithms.

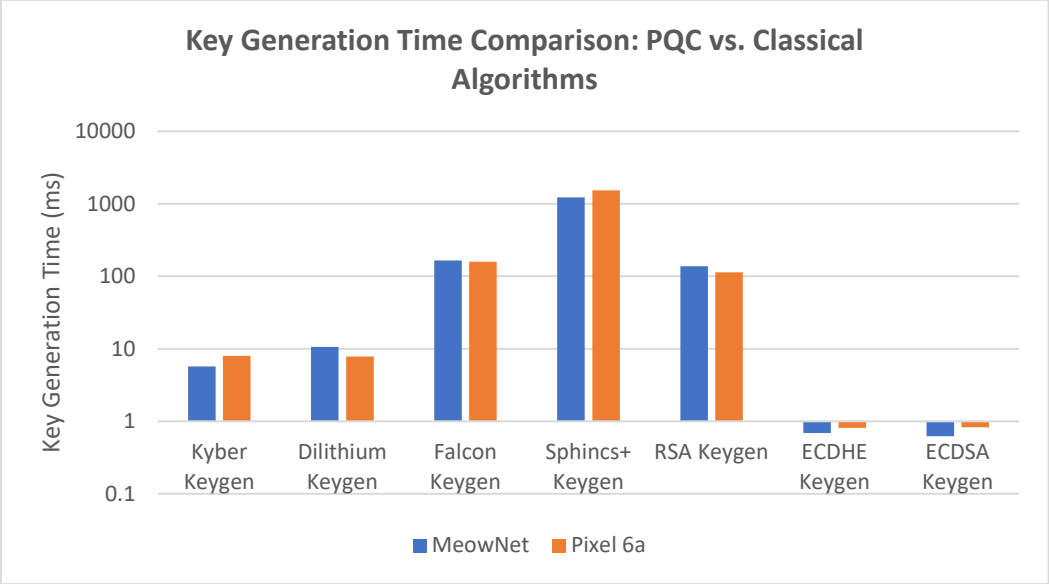


Figure 7. Key Generation Time Comparison in Log_{10} Scale: PQC vs. Classical Algorithms.

5.4.2. Digital Signature Generation Time

Next, we will analyze the results of the performance of the PQC signature algorithms against the classical algorithms. Figure 8 focuses on the generation of digital signatures.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 January 2026 doi:10.20944/preprints202601.1876.v1

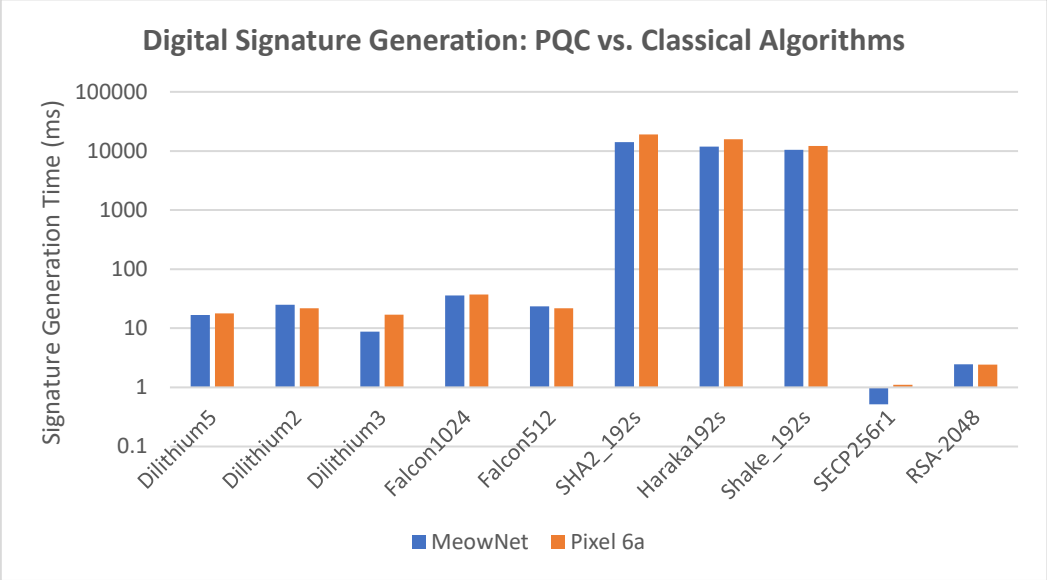


Figure 8. Digital Signature Generation in Log_{10} Scale: PQC vs. Classical Algorithms.

Dilithium and Falcon emerge as the fastest among the PQC algorithms, making it highly suitable for applications in mobile communication. Dilithium and Falcon provide a good compromise between speed and security, while SPHINCS+ variants, Sha2_192s, haraka192s and shake_192s, again shows longer signature generation times, reinforcing its limited practicality in time-sensitive applications. Classical algorithms like RSA and ECDSA indicated by secp256r1in the figure retain their efficiency, with the lowest signature generation times overall, showcasing their optimization for current hardware.

Figure 9 provides an overview with SPHINCS+ comparison removed for better analysis between the other PQC algorithms against the classical algorithms. The removal of less practical algorithms for SPHINCS+ such as SHA2_192s, Haraka192s, and SHAKE_192s streamlines the analysis, allowing a clearer focus on algorithms with practical relevance and manageable signature generation times. This exclusion sharpens the comparison, emphasizing the strengths and weaknesses of the remaining algorithms for real-world applications.

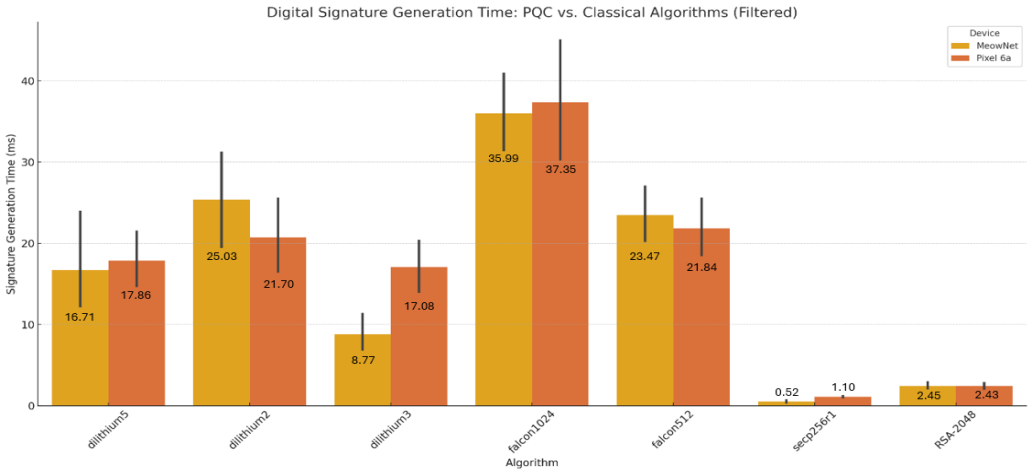


Figure 9. Digital Signature Generation: PQC vs. Classical Algorithms (Filtered).

Among the remaining post-quantum cryptography (PQC) algorithms, Dilithium offers balanced performance. Classical algorithms like ECDSA remain highly optimized and perform consistently across devices. This streamlined analysis underscores Falcon and Dilithium as leading candidates for PQC integration in mobile environments. Classical algorithms remain benchmarks for efficiency.

 doi:10.20944/preprints202601.1876.v1

ECDSA continues to dominate in terms of performance, achieving the lowest signature generation times overall, reaffirming its optimization for existing hardware and protocols. While RSA is slower than ECDSA, it remains competitive when compared to PQC algorithms like Dilithium and Falcon, particularly on devices with limited processing power such as the Pixel 6a. This reinforces the enduring utility of classical algorithms in environments where transition to PQC is still underway.

In the following sessions, we will dive into the operational metrics such as encrypt and sign, signature verification and decrypt processes.

5.4.3. Encrypt and Sign Duration, CPU Time and Temperature Delta

Figure 10 illustrates the time taken for the combined operation of encryption and signing. Kyber and Dilithium stand out with lower durations, demonstrating their suitability for secure mobile communications. SPHNICS+, however, requires significantly more time, particularly on the Pixel 6a, further emphasizing its resource-intensive nature. These results suggest that Kyber and Dilithium are better suited for real-time applications where low latency is critical.

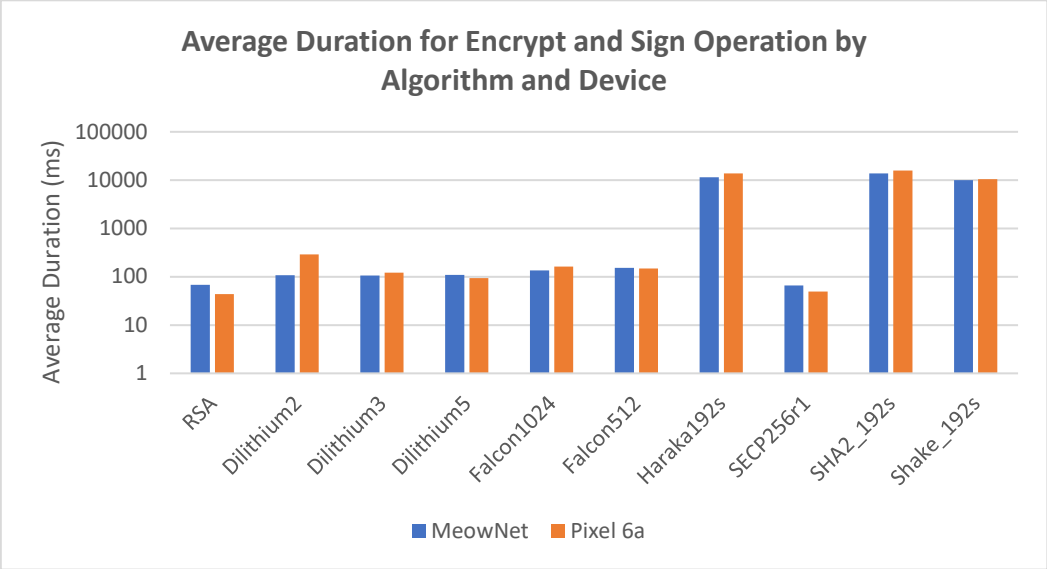


Figure 10. Average Duration in Log_{10} Scale for Encrypt and Sign Operation by Algorithm and Device.

Figures 11 and 12 also provide insights into the CPU time and temperature delta when performing the encrypt and sign operations. The average CPU time consumed during encrypt-and-sign operations. Kyber and Dilithium again perform well, indicating minimal strain on device resources. Conversely, SPHINCS+ demands higher CPU time, which could lead to reduced performance and increased energy consumption in mobile devices. The change in device temperature during encrypt-and-sign operations shows lower temperature deltas for Kyber and Dilithium indicate their energy efficiency, while SPHINCS+ shows higher deltas, potentially impacting device longevity and user experience in prolonged sessions.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 January 2026 doi:10.20944/preprints202601.1876.v1

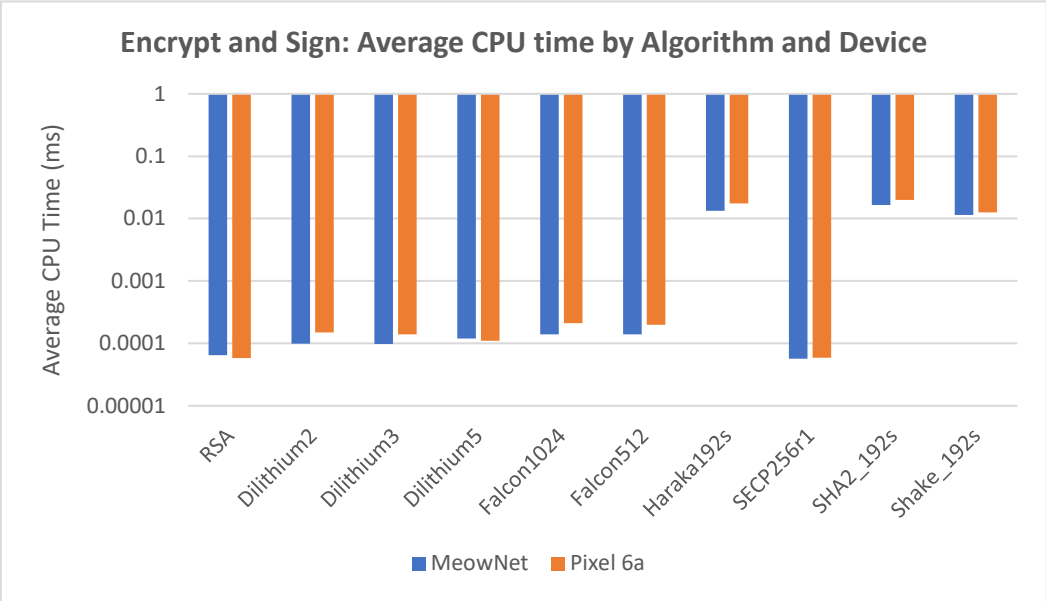


Figure 11. Encrypt and Sign: Average CPU time in Log_{10} Scale by Algorithm and Device.

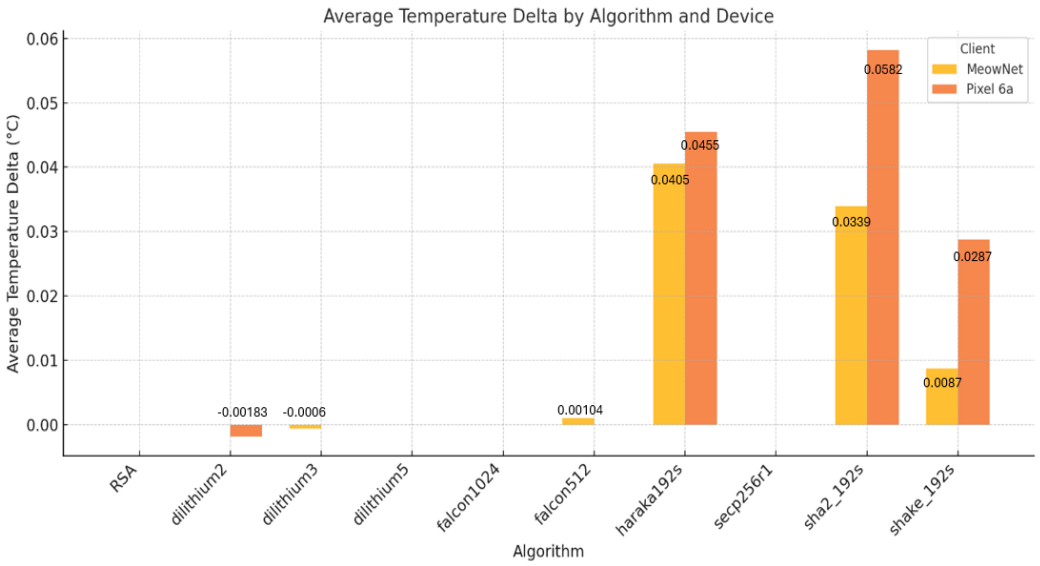


Figure 12. Average Temperature Delta by Algorithm and Device.

5.4.4. Verify and Decrypt Duration, CPU Time and Temperature Delta

Results from Figure 13 focus on the time required for signature verification operations. Dilithium provides a reasonable balance. SPHINCS+, especially for shake_192s seems to outperform Dilithium in the signature verification operations.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 26 January 2026 doi:10.20944/preprints202601.1876.v1

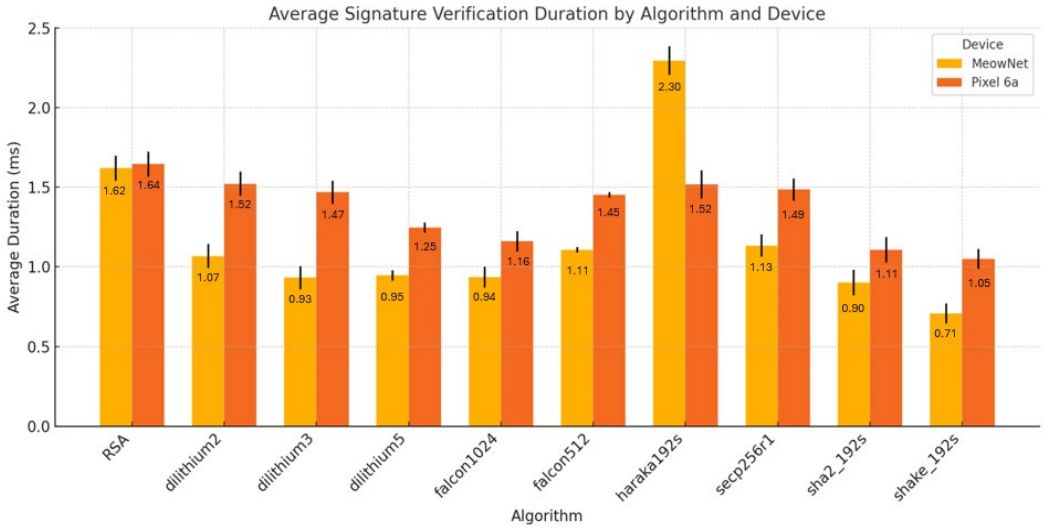


Figure 13. Average Signature Verification Duration by Algorithm and Device.

The average CPU time for verify-and-decrypt operations in Figure 14 shows that SHPINCS+ are performing slightly better than Dilithium and Falcon. Figure 15 shows that there are no temperature changes during the verify and decrypt operations on the different clients. This suggests that the different clients are handling the verification of signatures well with minimal resource consumption.

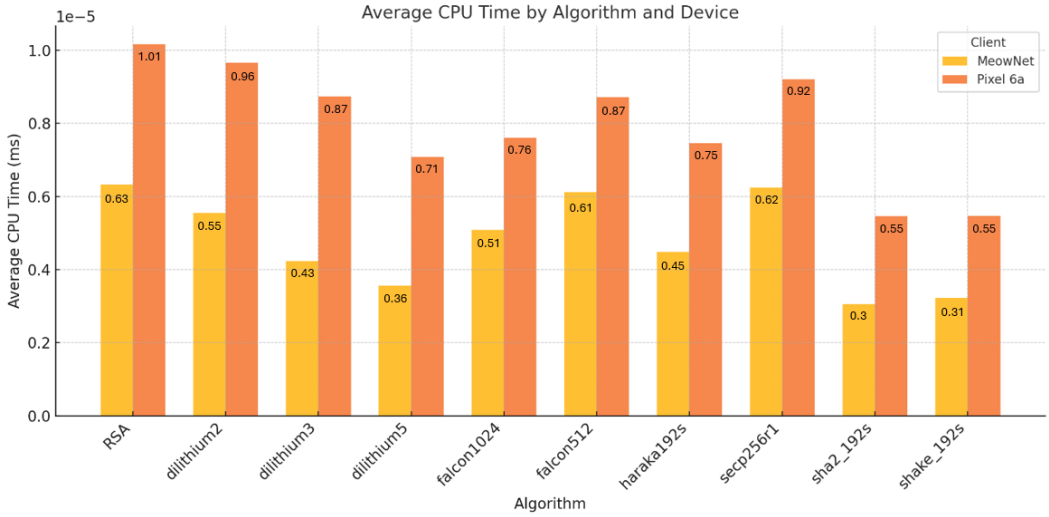
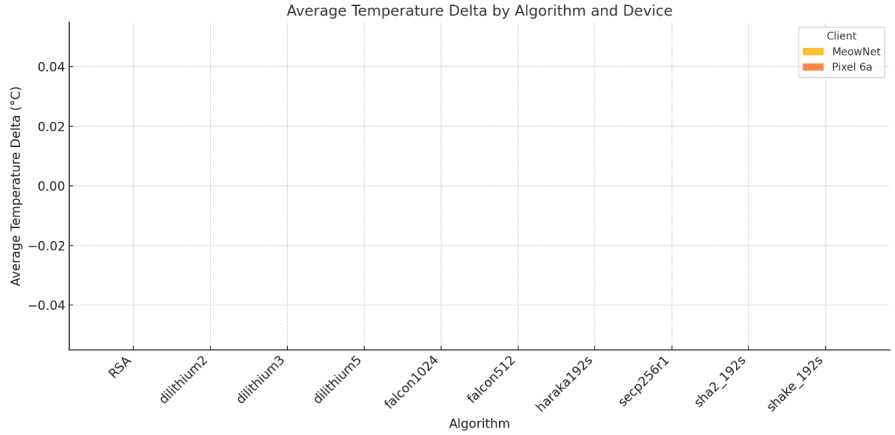


Figure 14. Verify and Decrypt: Average CPU Time by Algorithm and Device.



 doi:10.20944/preprints202601.1876.v1

Figure 15. Verify and Decrypt: Average Temperature Delta by Algorithm and Device.

5.4.5. Individual Signature Generation Duration, CPU Time and Temperature Delta

Figures 16–18 detail the duration, CPU time, and temperature delta for individual signature generation and verification tasks. Consistently, among the PQC algorithms Falcon and Dilithium prove efficient, balancing speed and resource utilization. SPHINCS+ struggles with longer durations and higher resource consumption and significant increase in temperature, reaffirming its limited practical use in mobile contexts.

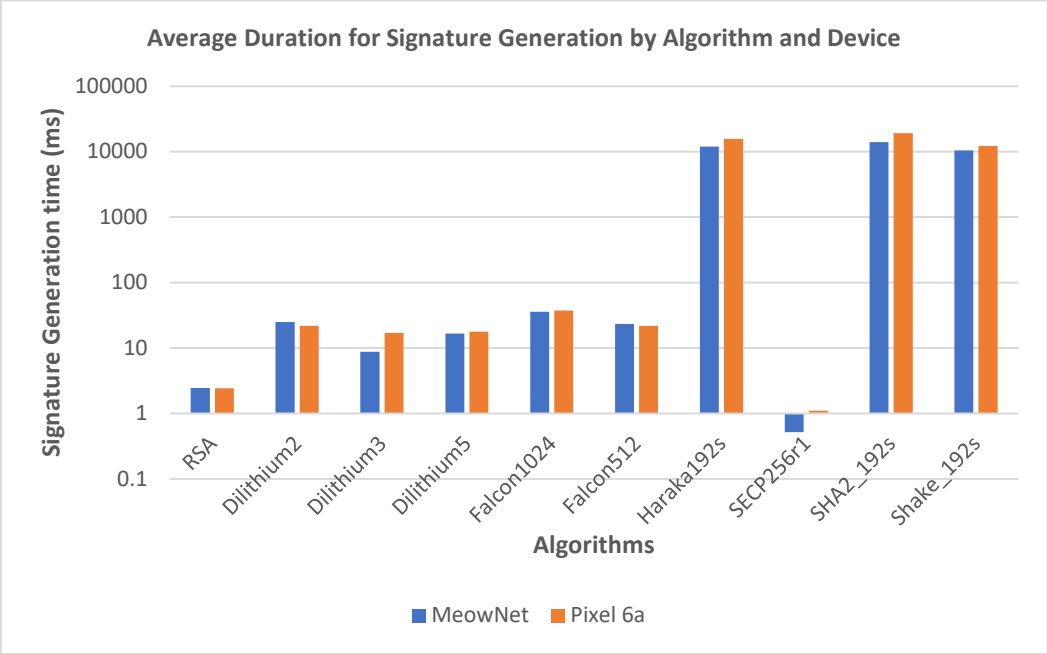


Figure 16. Average Duration in Log_{10} Scale for Signature Generation by Algorithm and Device.

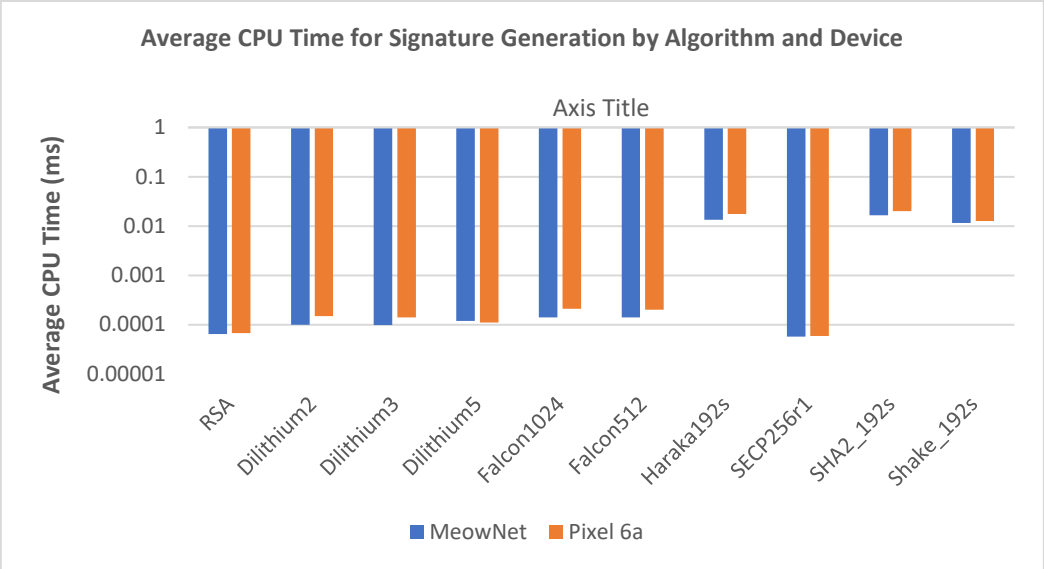


Figure 17. Average CPU Time in Log_{10} Scale for Signature Generation by Algorithm and Device.

 doi:10.20944/preprints202601.1876.v1

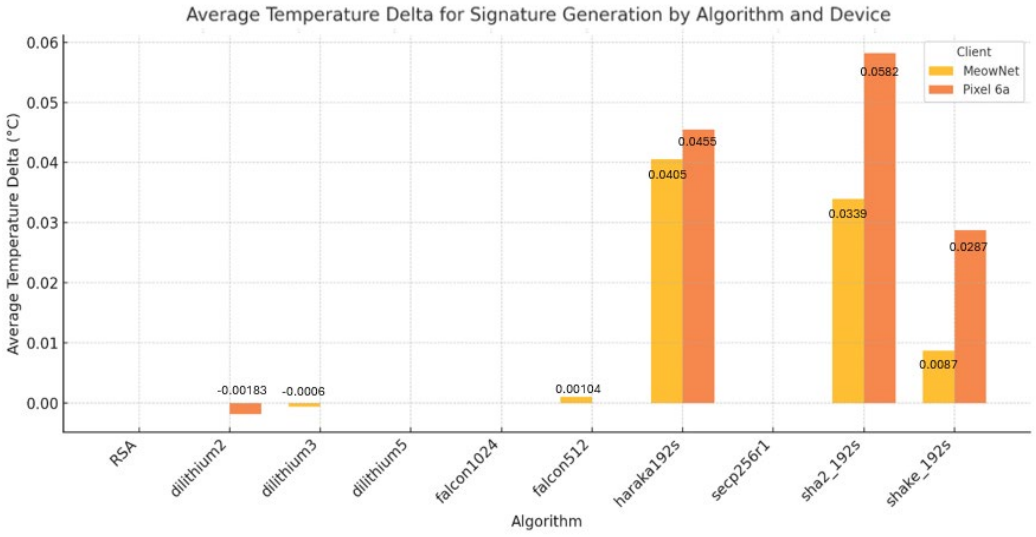


Figure 18. Average Temperature Delta for Signature Generation by Algorithm and Device.

5.4.6. Verify and Decrypt Operation Duration in Signature Verification

Figure 19 breaks down the verify and decrypt operation, focusing on the signature verification aspects. Compared to Figure 13, on average, the signature verification process takes up about one-third to half of the time required for the verify and decrypt operation.

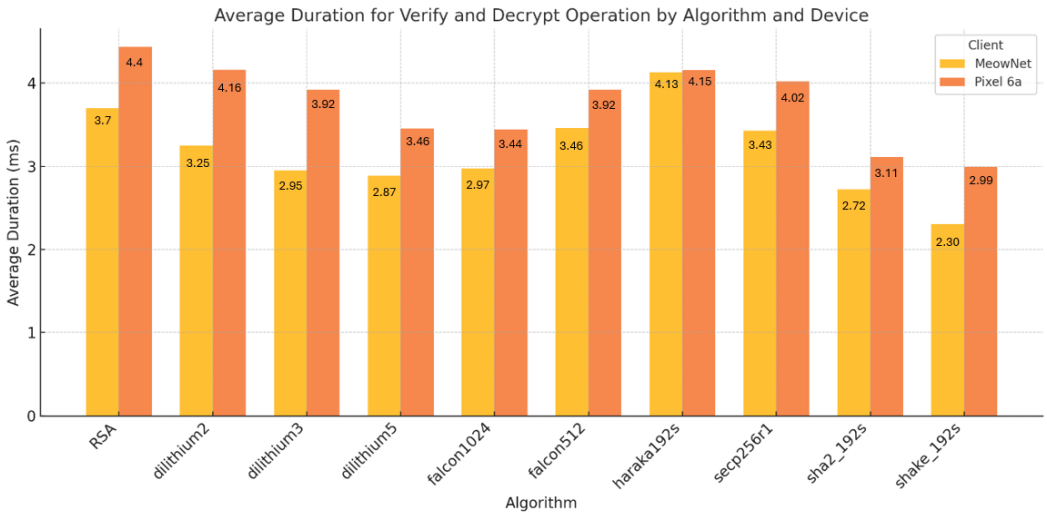


Figure 19. Average Duration for Verify and Decrypt by Algorithm and Device.

5.4.7. Key Exchange Duration, CPU Time and Temperature Delta

Figures 20–22 evaluate key exchange operations, Kyber requires more time in terms of key exchange but is a viable candidate in terms of performance across all metrics. The CPU time is comparable to classical KEM algorithms and there was no temperature changes detected during the KEM operations. Kyber's efficiency in duration, CPU time, and temperature delta positions it as the most practical choice for secure, quantum-resistant key exchanges in mobile communication systems.

 doi:10.20944/preprints202601.1876.v1

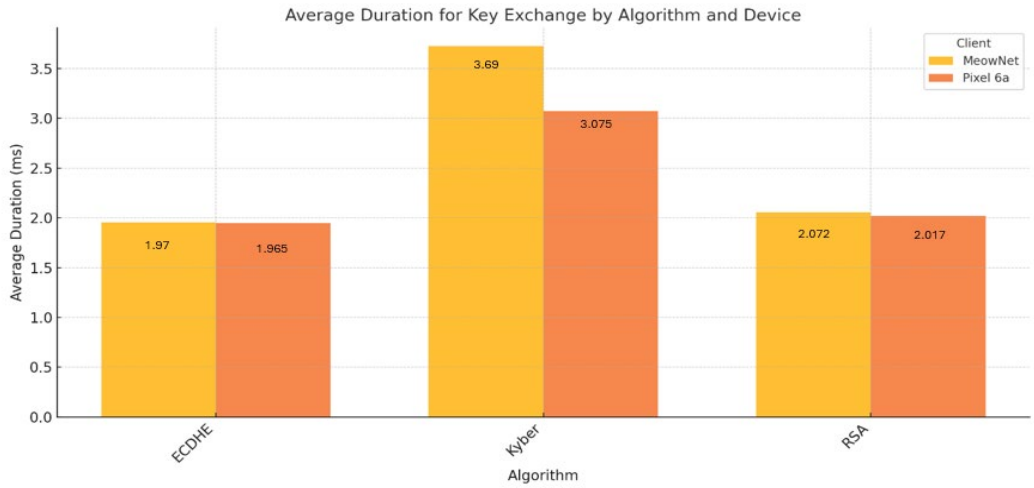


Figure 20. Average Duration for Key Exchange by Algorithm and Device.

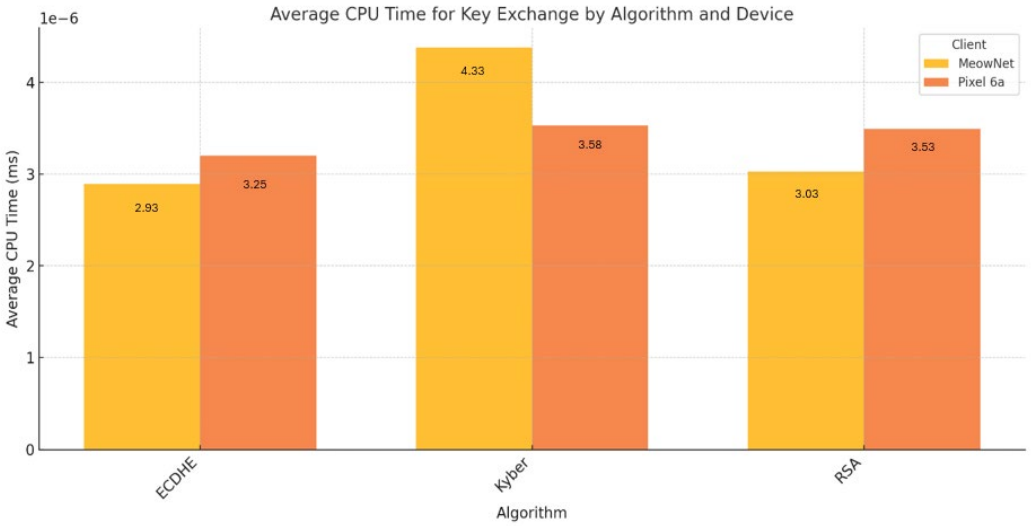


Figure 21. Average CPU Time for Key Exchange by Algorithm and Device.

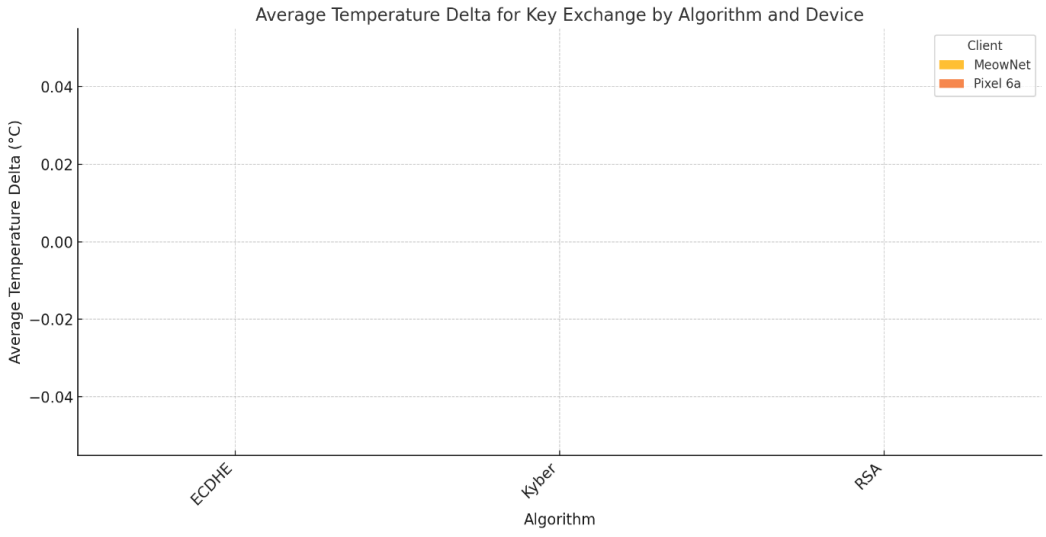


Figure 22. Average Temperature Delta for Key Exchange by Algorithm and Device.

 doi:10.20944/preprints202601.1876.v1

6. Discussions

The integration of PQC into mobile communication systems presents significant opportunities and challenges. This study provides critical insights into the performance and practical applicability of various PQC algorithms, offering a roadmap for their deployment in real-world environments. The following discussion delves into the comparative performance of these algorithms, their interaction with hardware, and their implications for Quality of Service.

The results obtained in this study align with findings reported in existing PQC performance literature. Prior benchmarking studies, such as those conducted by Lauterbach [46]. and Asecuritysite [30,47,48], also observed that Kyber exhibits relatively low-key generation and encapsulation time compared to classical RSA, particularly on constrained hardware. Similarly, other evaluations of Dilithium and Falcon consistently report faster signing performance and substantially faster verification times than SPHINCS+, reinforcing the trends observed in this experiment. Variations in absolute timing values between our results and published studies can be attributed to differences in device hardware, cryptographic library implementations, and measurement environments.

6.1. Comparison of PQC Benchmarking Works

Table 5 summarizes the related PQC benchmarking works. Our work has performed full PQC workflow implementation inside a mobile app with actual device metrics, temperature tracking, and Firebase-based message flow, and it is fully end-to-end mobile benchmarking, not desktop simulation. The other works in existing PQC performance literature are either desktop-only with no mobile OS, no thermal analysis, no mobile workflow or messaging integration, or not real devices with no temperature or mobile CPU/thermal constraints, and not an application workflow. Our work also measured more performance items than other works. Refer to Annex B for a table for more detailed comparison.

Table 5. Comparison of Related PQC Benchmarking Works.

Study / Citation	Application Domain	Devices Used	OS	PQC Algorithms Evaluated	Performance Items Measured	Libraries / APIs Used	Key Differences vs. Your Study
Our Work	Mobile communication workflow; real Android chat-style PQC message exchange	Consumer smart-phones	Android	Kyber, Dilithium, Falcon, SPHINCS+	Key generation, signing, verification, encryption, decryption, CPU-time, temperature delta	BouncyCastle PQC provider, custom Android instrumentation, Firestore logging	Full PQC workflow implementation *inside a mobile app* with real device metrics, temperature tracking, and Firebase-based message flow. Fully end-to-end mobile benchmarking, not desktop simulation.
Lauterbach [26]	General Internet data protection and secure communication over networks	PC-class platform is used.	Linux (Ubuntu)	Kyber, Dilithium	Operation duration, CPU cycles	Not Specified	Desktop-only. No mobile OS, no thermal analysis. No mobile workflow or messaging integration.
Asecuritysite [30,47,48]	General algorithm speed comparison	Desktop / laptop CPU	Linux	Kyber, BIKE, HQC, FrodoKEM, Dilithium, Falcon, SPHINCS+	Basic keygen/sign/verify timing	Not Specified	Not real devices. No temperature or mobile CPU/thermal constraints. Not an application workflow.

6.2. KEM Findings: Kyber on Mobile

Kyber emerged as a standout candidate for Key Encapsulation Mechanisms due to its efficient performance across different hardware. The results show that Kyber consistently achieved low key generation times, which are pivotal for real-time applications in mobile communications. To contextualize these findings, a comparison with existing literature was made. Prior studies such as [8] focus not on KEM execution time, but on network-level handshake overhead incurred when substituting classical algorithms with PQC. Their measurements show that replacing X25519 and RSA-2048 with Kyber-512 and Dilithium2 increases TLS handshake cryptographic payload sizes from tens of kilobytes to several hundred kilobytes, with some applications experiencing increases from 70.2 kB → 584.1 kB or 353.6 kB → 2943.2 kB per handshake. This highlights the bandwidth implications of PQC adoption rather than the computational cost.

 doi:10.20944/preprints202601.1876.v1

In contrast, this paper evaluates device-side computation, not network overhead. Kyber's on-device key generation times on Android, typically in the X–Y ms range depending on device, remain practical and substantially faster than classical RSA-2048 key generation operations. The difference between these studies underscores that network overhead and local computational cost measure distinct aspects of PQC feasibility. Kyber may increase TLS handshake size yet still performs efficiently when executed directly on consumer mobile hardware.

The key exchange operations in this study were facilitated by a lightweight HTTPS Flask server hosted on an AWS EC2 instance (Intel® Xeon® E5-2676 v3 @ 2.40 GHz, 1 GB RAM), which acted purely as a transport layer and did not influence algorithmic computation. This ensures that the measurements for Kyber reflect device-local cryptographic performance rather than network latency or server-side processing overhead.

Overall, the experimental results demonstrate that Kyber offers a strong balance of performance and security for mobile platforms. Despite the increased network overhead reported in the literature, Kyber's computational efficiency on Android smartphones suggests that it remains a viable and practical KEM candidate for quantum-safe mobile communication workflows.

6.3. Signature Findings: Dilithium vs Falcon vs SPHINCS+

For digital signature algorithms, Falcon and Dilithium both demonstrated strong performance in digital signature operations. Falcon's compact key and signature sizes, coupled with its optimized verification process, made it suitable for use in mobile communications. Dilithium, with its balanced approach to speed and security, proved ideal for applications requiring frequent authentication, such as secure messaging platforms. Conversely, SPHINCS+ is largely unsuitable for use in mobile communications, due to its larger signatures and slower processing times, underscoring its niche applicability in scenarios prioritizing long-term data security over operational efficiency. In addition, large PQC payloads (especially SPHINCS+ signatures) may increase network transmission latency, particularly in chatty or low-bandwidth environments.

6.4. Device and Hardware Effects

This study revealed significant performance disparities based on device capabilities. High-performance devices, such as the client, MeowNet with its Snapdragon Gen 1 processor, consistently outperformed mid-range hardware like the Pixel 6a, running on Google Tensor (5nm). This advantage was particularly evident for computationally demanding algorithms like SPHINCS+. These findings suggest that successful PQC integration may require hardware optimization or the adoption of accelerators to minimize resource strain.

Conversely, algorithms like Kyber and Dilithium maintained robust performance even on less powerful devices, demonstrating their adaptability and suitability for diverse hardware environments. This makes them ideal for deployment in mobile networks, where a wide range of devices coexists.

6.5. QoS Implications in 5G/6G Workloads

In mobile networks like 5G and 6G, maintaining high QoS (Quality of Service) is critical in mobile networks, particularly for applications such as telemedicine, autonomous vehicles, and IoT systems. Algorithms like Kyber and Dilithium align well with QoS requirements, offering low latency and efficient resource utilization. Their integration ensures that cryptographic operations do not impede network performance, even under high traffic conditions.

In contrast, SPHINCS+ poses challenges for QoS due to its extended computational times and higher energy consumption. These factors could lead to increased latency and reduced user satisfaction in latency-sensitive applications. As such, SPHINCS+ may be better suited for archival encryption or other use cases where performance is a secondary concern.

 doi:10.20944/preprints202601.1876.v1

As mobile communication networks expand, integrating PQC at scale presents challenges. One of the most significant concerns is computational overhead and latency. While Kyber and Dilithium exhibit reasonable processing times, making them viable for mobile networks, SPHINCS+ remains unsuitable due to excessive computational demands. Optimizing cryptographic libraries and leveraging hardware accelerators, such as FPGAs and GPUs, could potentially mitigate performance issues and improve overall efficiency.

6.6. Computational Overheads and Thermal Considerations

Another critical aspect of PQC adoption is the transition from classical cryptographic systems. This transition has to be incremental, ensuring backward compatibility with existing systems. Hybrid cryptographic approaches, such as Hybrid-TLS implementations combining Kyber with ECDHE and Dilithium with ECDSA, can facilitate a smoother transition and allow legacy systems to gradually adapt to quantum-resistant protocols without complete overhauls.

Beyond individual device implementations, the scalability of PQC requires broader network-level integration. While current implementations focus on device-to-device communication, real-world deployment necessitates a more comprehensive approach. 5G and emerging 6G core networks must integrate quantum-resistant algorithms to ensure that the backbone of mobile communications remains secure. This includes adapting existing security protocols to accommodate PQC within the entire communication stack, from key exchange mechanisms to encrypted data transmission.

Lastly, based on the findings, this study recommends the use of Kyber as the Key Exchange Mechanism for key exchange operations for mobile communications. Dilithium and Falcon can be implemented as the Digital Signature algorithm for mobile communications.

While this study provides valuable insights, it is not without limitations. The experiments were conducted in controlled environments, which may not fully capture the complexities of real-world network conditions. Additionally, further research is needed to explore the impact of emerging hardware accelerators and optimize PQC algorithms for broader deployment.

7. Future Work

The integration of Post-Quantum Cryptography (PQC) into mobile communication networks represents a critical step toward quantum-safe security. However, this study highlights several areas that warrant further exploration to ensure the practical deployment and optimization of PQC solutions.

Besides a trustworthy CA to issued credentials for PQC as a part of future work, future research could focus on optimizing PQC algorithms for mobile devices with limited computational resources. Lightweight implementations of resource-intensive algorithms like SPHINCS+ could significantly enhance their applicability in real-world scenarios. Additionally, fine-tuning algorithms to reduce power consumption and improve processing speeds will be critical for mobile and IoT devices.

Exploring the role of hardware accelerators, such as FPGAs or GPUs, in enhancing the efficiency of PQC algorithms offers a promising avenue. Hardware optimization for Kyber and Dilithium could enable real-time encryption in latency-sensitive applications like video conferencing and autonomous vehicles.

During the transition to quantum-safe cryptographic standards, hybrid systems that combine classical and PQC algorithms will play a crucial role. Future research should develop frameworks that ensure seamless interoperability between these systems, particularly in protocols like TLS.

The resilience of PQC algorithms must be continually evaluated against emerging threats. Advancements in quantum computing and cryptanalysis techniques may necessitate further refinement of algorithms like Kyber, Dilithium, and Falcon to maintain their security guarantees.

Collaboration with standardization bodies such as NIST will be essential for refining PQC standards and addressing regulatory challenges. Engaging with international organizations can accelerate the global adoption of quantum-safe cryptography, ensuring interoperability and compliance.

 doi:10.20944/preprints202601.1876.v1

8. Challenges

This section outlines the key challenges encountered in our project so far, which also pave the way for areas of future work. Addressing these challenges is crucial for the successful implementation and further development of our project.

8.1. Absence of Certificate Authority

One of the primary challenges we have faced is the absence of a CA in our implementation. The absence of a trusted CA in this study required the use of self-signed certificates, which lack the trustworthiness of CA-issued credentials. This approach, while suitable for testing, should not be implemented in production environments.

8.2. Challenges for Mobile Application Development and PQC Integration

The development of the Android application has posed its set of challenges. Creating an application that effectively incorporates PQC algorithms requires a deep understanding of both mobile application development and advanced cryptographic techniques. Ensuring that the app is user-friendly, secure, and performs efficiently, while integrating the complex functionalities of PQC, requires careful attention to detail and expertise in both fields.

8.3. Message Routing and Payload Limits

Another challenge is the handling of the message routing and persistence of the sent data from one client to another client. Managing the transmission and storage of encrypted data in mobile networks presents logistical challenges. Services like Firebase, while efficient, impose payload size limitations that complicate the delivery of encrypted messages. A hybrid approach, storing encrypted data in a secure database and transmitting message identifiers via Firebase, provides a scalable solution.

8.4. PQC Library Maturity

Last significant challenge lies in the selection and choice of appropriate PQC libraries. With the field of PQC still in its standardization stages, the choice of libraries that supports the shortlisted algorithms are limited, and often these libraries are not fully optimized for mobile platforms. Deciding on the right PQC library involves considering factors such as compatibility with mobile platforms, the efficiency of the algorithms, the level of security provided, and the ease of integration into existing systems.

9. Conclusion

This paper has explored the critical roles of Post-Quantum Cryptography in addressing the profound challenges posed by quantum computing. The weaknesses of classical cryptographic systems, particularly algorithms like RSA and ECC, underscore the urgency of transitioning to quantum-resistant algorithms. As demonstrated in many studies and experiments, quantum advancements, such as Shor's and Grover's algorithms, present an imminent threat to the confidentiality, integrity, and authenticity of digital communications.

Through experimental evaluation, this research highlights the immense potential of PQC algorithms, particularly Kyber and Dilithium, in safeguarding mobile communication systems. These algorithms balance computational efficiency with robust security, making them viable for real-world implementation in resource-constrained environments such as 5G and 6G networks. Additionally, industry adoption by major organizations like AWS, IBM, and Cloudflare underscores the feasibility and urgency of integrating PQC into modern infrastructure.

The key findings of this study include the following points:

 doi:10.20944/preprints202601.1876.v1

1. Kyber demonstrated efficient key encapsulation mechanisms, making it a strong candidate for secure key exchange in mobile communications.

2. Dilithium and Falcon proved suitable for digital signatures, balancing performance and security, whereas SPHINCS+ exhibited higher computational costs, making it less practical for real-time applications and mobile communications.

3. Hardware efficiency plays a critical role, with high-performance devices (e.g., MeowNet with Snapdragon Gen 1) significantly outperforming mid-range devices (e.g., Pixel 6a with Google Tensor).

4. The integration of PQC in mobile networks must consider factors such as computational overhead, latency, and hybrid cryptographic approaches for a seamless transition.

While this study provides valuable insights, it is not without limitations. The experiments were conducted in controlled environments, which may not fully capture the complexities of real-world network conditions. Future research should explore large-scale implementations, hardware accelerations, and real-world PQC deployments in diverse mobile ecosystems.

By bridging theoretical advancements with practical applications, this work contributes to the ongoing efforts toward quantum-resistant cryptography. As quantum technologies continue to evolve, the adaptation of PQC is essential to ensure secure, efficient, and scalable digital communications in the quantum era.

Author Contributions: Original draft preparation: Rongjie Zhou; Supervision and paper revision: Huaqun Guo; Supervision: Francis E C Teo. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the scholarship from the Future Communications Research and Development Programme (FCP), Infocomm Media Development Authority (IMDA) and National Research Foundation Singapore (NRF), Singapore.

Conflicts of Interest: The authors declare no conflicts of interest.:

Abbreviations

The following abbreviations are used in this manuscript:

ACM	AWS Certificate Manager
AEAD	Authenticated encryption with associated data
AES	Advanced Encryption Standards
AES-GCM	Advanced Encryption Standard - Galois/Counter Mode
API	Application Programming Interface
AWS	Amazon Web Service
CA	Certificate Authority
CPU	Central Processing Unit
DHE	Diffie-Hellman Ephemeral
ECDHE	Elliptic Curve Diffie-Hellman Ephemeral
ECDSA	Elliptic Curve Digital Signature Algorithm
FCM	Firebase Cloud Messaging
GCM	Galois/Counter Mode
IMDA	Infocomm Media Development Authority
IND-CCA	Indistinguishability security under the Chosen Ciphertext Attack
IoT	Internet of Things
IV	Initialization Vector
KDF	Key Derivative Function
KEM	Key Encapsulation Mechanism
KMS	Key Management Service
NIST	National Institute of Standards and Technology
PQC	Post Quantum Cryptography
QoS	Quality of Service
RSA	Rivest-Shamir-Adleman
SHA	Secure Hash Algorithm
TLS	Transport Layer Security

 doi:10.20944/preprints202601.1876.v1

References

1. IBM, "How SSL and TLS provide identification authentication confidentiality and integrity", IBM MQ 7.5, 30th April 2018. Retrieved October 12, 2023 from https://www.ibm.com/docs/en/ibm-mq/7.5?topic=ssl-how-tls-provide-authentication-confidentiality-integrity.
2. V. Chamola, A. Jolfaei, V. Chanana, P. Parashari, and V. Hassija, "Information security in the post quantum era for 5G and beyond networks: Threats to existing cryptography, and post-quantum cryptography," Computer Communications, 2021, 176, pp. 99–118. https://doi.org/10.1016/j.comcom.2021.05.019
3. G. Amponis et al., "Towards Securing Next-Generation Networks: Attacking 5G Core/RAN Testbed," 2022 Panhellenic Conference on Electronics & Telecommunications (PACET), Tripolis, Greece, 2022, pp. 1-4, doi: 10.1109/PACET56979.2022.9976365.
4. H. Yang et al., "Data-Driven Network Slicing From Core to RAN for 5G Broadcasting Services," in IEEE Transactions on Broadcasting, vol. 67, no. 1, pp. 23-32, March 2021, doi: 10.1109/TBC.2020.3031742.
5. M. B. Yassein, S. Aljawarneh, E. Qawasmeh, W. Mardini and Y. Khamayseh, "Comprehensive study of symmetric key and asymmetric key encryption algorithms," 2017 International Conference on Engineering and Technology (ICET), Antalya, Turkey, 2017, pp. 1-7, doi: 10.1109/ICEngTechnol.2017.8308215.
6. M. Kim, Y. Shin and T. Shon, "MitM Tool Analysis for TLS Forensics," 2021 International Conference on Platform Technology and Service (PlatCon), Jeju, Korea, Republic of, 2021, pp. 1-4, doi: 10.1109/PlatCon53246.2021.9680752.
7. I. Kotuliak, P. Rybár and P. Trúchly, "Performance comparison of IPsec and TLS based VPN technologies," the 9th International Conference on Emerging eLearning Technologies and Applications (ICETA), Stara Lesna, Slovakia, 2011, pp. 217-221, 2011, doi: 10.1109/ICETA.2011.6112567
8. D. Mankowski, T. Wiggers and V. Moonsamy, "TLS → Post-Quantum TLS: Inspecting the TLS landscape for PQC adoption on Android," 2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), Delft, Netherlands, 2023, pp. 526-538, doi: 10.1109/EuroSPW59978.2023.00065
9. J. Arora, K. Saluja, S. Gupta, S. Sharma and G. Kaur, "Handling Secret Key Compromise by Deriving Multiple Asymmetric Keys based on Diffie-Hellman Algorithm," 2023 8th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 2023, pp. 492-498, doi: 10.1109/ICCES57224.2023.10192607
10. V. Bhatia and K. R. Ramkumar, "An Efficient Quantum Computing technique for cracking RSA using Shor's Algorithm," IEEE 5th International Conference on Computing Communication and Automation (ICCCA), Greater Noida, India, 2020, pp. 89-94, doi: 10.1109/ICCCA49541.2020.9250806.
11. Kudelski Security Team. Quantum Attack Resource Estimate: Using Shor's Algorithm to Break RSA vs DH/DSA VS ECC. August 24, 2021. Retrieved December 12, 2025 from https://kudelskisecurity.com/research/quantum-attack-resource-estimate-using-shors-algorithm-to-break-rsa-vs-dh-dsa-vs-ecc.
12. Palaganti Venkata Nagendra Sai Sandeep. The Quadratic Leap: How Grover's Algorithm is Redefining Core Problems in Computer Science and Engineering. International Journal of Innovative Science and Research Technology. Volume 10, Issue 7, July– 2025. https://doi.org/10.38124/ijisrt/25jul1660.
13. GSM Association, "The Mobile Economy" Retrieved October 12, 2023 from https://www.gsma.com/mobileeconomy/
14. NIST, "PQC Standardization Process: Announcing Four Candidates to be Standardized, Plus Fourth Round Candidates" Retreived October 17, 2023 from https://csrc.nist.gov/news/2022/pqc-candidates-to-be-standardized-and-round-4.
15. M. Mehic et al., "Quantum Cryptography in 5G Networks: A Comprehensive Overview," in IEEE Communications Surveys & Tutorials, vol. 26, no. 1, pp. 302-346, Firstquarter 2024, doi: 10.1109/COMST.2023.3309051
16. NIST, "NIST Releases First 3 Finalized Post-Quantum Encryption Standards", Retrieved August 13, 2024. https://www.nist.gov/news-events/news/2024/08/nist-releases-first-3-finalized-post-quantum-encryption-standards
17. J. Lee, "How Quantum Computers Will Impact Bitcoin". Retrieved October 12, 2024 from https://medium.com/@kyn04138/how-quantum-computers-will-impact-bitcoin-ebac8dae2a4e.

 doi:10.20944/preprints202601.1876.v1

18. NIST, “Post-Quantum Cryptography - Post-Quantum Cryptography Standardization” Retrieved October 27, 2023 from https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization
19. NIST, “Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process”, Retrieved August 14, 2024. https://csrc.nist.gov/pubs/ir/8413/upd1/final.
20. Google, “Meet Willow, our state-of-the-art quantum chip”, Retrieved December 17, 2024 from https://blog.google/technology/research/google-willow-quantum-chip/
21. IBM, “The hardware and software for the era of quantum utility is here” Retrieved January 13, 2026 from https://www.ibm.com/quantum/blog/quantum-roadmap-2033.
22. F. Bene and A. Kiss, "Public Key Infrastructure in the Post-Quantum Era," 2023 IEEE 17th International Symposium on Applied Computational Intelligence and Informatics (SACI), Timisoara, Romania, 2023, pp. 000077-000082, doi: 10.1109/SACI58269.2023.10158562
23. PW. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer,” SIAM Journal of Computing, vol. 26, no. 5, pp.1484–509, 1997.
24. B. Yan, Z. Tan, S. Wei, H. Jiang, W. Wang, H. Wang, L. Zuo, Q. Duan, Y. Liu, W. Shi, Y. Fei, X. Meng, Y. Han, Z. Shan, J. Chen, X. Zhu, C. Zhang, F. Jin, H. Li, C. Song, Z. Wang, Z. Ma, H. Wang and G. Long, “Factoring integers with sublinear resources on a superconducting quantum processor”, arxiv:2212.12372v1, 23 Dec 2022
25. E. Zeydan, Y. Turk, B. Aksoy and S. B. Ozturk, "Recent Advances in Post-Quantum Cryptography for Networks: A Survey," the Seventh International Conference On Mobile And Secure Services (MobiSecServ), Gainesville, FL, USA, 2022, pp. 1-8, doi: 10.1109/MobiSecServ50855.2022.9727214.
26. N. Gupta, A. Jati, A. Chattopadhyay and G. Jha, "Lightweight Hardware Accelerator for Post-Quantum Digital Signature CRYSTALS-Dilithium," in IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 70, no. 8, pp. 3234-3243, Aug. 2023, doi: 10.1109/TCSI.2023.3274599.
27. P. Kampanakis, T. Hansen, A. Volanis and G. Ravago, “Post-quantum hybrid SFTP file transfers using AWS Transfer Family”, AWS Security Blog, June 3, 2023. Retrieved October 12, 2023 from https://aws.amazon.com/blogs/security/post-quantum-hybrid-sftp-file-transfers-using-aws-transfer-family/.
28. R. Zhou, H. Guo, F. E. C. Teo and S. Bakiras, "A Survey on Post-Quantum Cryptography for 5G/6G Communications," 2023 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), Singapore, 2023, pp. 1-6, doi: 10.1109/SOLI60636.2023.10425346.
29. G. R. Mounica, G. Manimaran, L. B. Jerome and P. Bhattacharjee, "Implementation of 5-Qubit approach-based Shor's Algorithm in IBM Qiskit," 2021 IEEE Pune Section International Conference (PuneCon), Pune, India, 2021, pp. 1-6, doi: 10.1109/PuneCon52575.2021.9686492.
30. B. B. OBE, “Energy Consumption of Post Quantum Cryptography: Dilithium and Kyber Beat Our Existing TLS 1.3 Performance”, ASecuritySite: When Bob Met Alice. Retrieved October 12, 2023 from https://medium.com/asecuritysite-when-bob-met-alice/energy-consumption-of-post-quantum-cryptography-dilithium-and-kyber-beat-our-existing-tls-1-3-ccadd04dd4c7.
31. G. Tasopoulos. C. Dimopoulos, A. P. Fournaris, R. K. Zhao, A. Sakzad and R. Steinfeld, “Energy Consumption Evaluation of Post-Quantum TLS 1.3 for Resource-Constrained Embedded Devices,” Cryptology ePrint Archive, Paper 2023/506.
32. B. Jarvis, “How to tune TLS for hybrid post-quantum cryptography with Kyber”, AWS Security Blog, July 5, 2022. Retrieved October 12, 2023 from https://aws.amazon.com/blogs/security/how-to-tune-tls-for-hybrid-post-quantum-cryptography-with-kyber/.
33. A. Dames, “Available on IBM z16: Future-Proof Digital Signatures with a Quantum-Safe Algorithm Selected by NIST”, IBM Blog, July 26, 2022. Retrieved October 12, 2023 from https://www.ibm.com/blog/announcement/available-on-ibm-z16-future-proof-digital-signatures-with-a-quantum-safe-algorithm-selected-by-nist/.
34. B. Westerbaan, and C. D. Rubin, “Defending against future threats: Cloudflare goes post-quantum”, The Cloudflare Blog, October 3, 2022. Retrieved October 12, 2023 from https://blog.cloudflare.com/post-quantm-for-all/.

 doi:10.20944/preprints202601.1876.v1

35. W. Evans, and B. Westerbaan, “Post-quantum crypto should be free, so we’re including it for free, forever”, The Cloudflare Blog, March 16, 2023. Retrieved October 12, 2023 from https://blog.cloudflare.com/post-quantum-crypto-should-be-free/
36. Ruben Gonzalez. Kyber - How does it work? Approachable Cryptography. September 14, 2021. Retrieved December 13, 2025 from https://cryptopedia.dev/posts/kyber/.
37. Alfred Menezes. V2: Kyber PKE and KEM. August 14, 2024. Retrieved December 13, 2025 from https://www.youtube.com/watch?v=nlGjqGdkmfI.
38. Denys Popov. CRYSTALS-Dilithium: The Digital Signature Scheme for the Post-Quantum Era. Medium. August 25, 2025. Retrieved December 13, 2025 from https://denispopovengineer.medium.com/crystals-dilithium-the-digital-signature-scheme-for-the-post-quantum-era-d8ba8f0213b9.
39. NIST. Module-Lattice-Based Digital Signature Standard. FIPS 204 Federal Information Processing Standards Publication. August 13, 2024. https://doi.org/10.6028/NIST.FIPS.204.
40. Shi Bai, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler and Damien Stehlé. CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation (Version 3.1). February 8, 2021
41. Falcon – A Post-Quantum Signature Scheme. June 28, 2019. https://pqshield.com/falcon-a-post-quantum-signature-scheme/
42. Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset. Gregor Seiler, William Whyte, Zhenfei Zhang. Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU. Specification v1.2 — 01/10/2020. https://falcon-sign.info/falcon.pdf
43. Alfred Menezes. Lecture 6. SPHINCS+ (Hash-Based Signatures). https://www.youtube.com/watch?v=Z93yiDyPUzE&t=124s
44. NIST. Stateless Hash-Based Digital Signature Standard. FIPS 205 Federal Information Processing Standards Publication. August 13, 2024. https://doi.org/10.6028/NIST.FIPS.205
45. Udara Pathum, "CRYSTALS Kyber: The Key to Post-Quantum Encryption. Identity Beyond Borders.", Retrieved August 14, 2024. https://medium.com/identity-beyond-borders/crystals-kyber-the-key-to-post-quantum-encryption-3154b305e7bd.
46. F. Lauterbach, P. Burdiak, F. Richter, and M. Voznak, “Performance analysis of post-quantum algorithms,” in Proc. 29th Telecommun. Forum (TELFOR), 2021, pp. 1–4.
47. W. J Buchanan, “PQC Key Encapsulation Mechanism (KEM) Speed Tests,” Asecuritysite.com, 2023. Retrieved October 12, 2023 from https://asecuritysite.com/pqc/pqc_kem.
48. W. J Buchanan, “PQC Digital Signature Speed Tests,” Asecuritysite.com, 2023. Retrived October 12, 2023 from https://asecuritysite.com/pqc/pqc_sig