**International Journal of Cybersecurity Engineering and Innovation**

ARTICLE INFO

**Corresponding author Email:** m.almaiah@ju.edu.jo




**Vol. 2027 No.1**

# A Post-Quantum Cryptographic Framework for Secure and Resilient IoT Communication

**Mohammed Amin Almaiah[1]*, Ansam Khraisat[2], Udit Mamodiya[3]**

[1] King Abdullah the II IT School, The University of Jordan, Amman 11942, Jordan, https://orcid.org/0000-0002-2215-2481

[2] School of Information Technology, Deakin University, Burwood, VIC 3125, Australia, https://orcid.org/0000-0002-8623-0987

[3] Faculty of Engineering and Technology, Poornima University, Jaipur, 303905, Rajasthan, India, https://orcid.org/0000-0002-8022-4252

## Abstract

The impending emergence of cryptographically relevant quantum computers poses an existential threat to the elliptic-curve and RSA-based key-exchange protocols that currently secure the vast majority of Internet of Things (IoT) deployments. Constrained IoT devices are especially exposed to "harvest-now, decrypt-later" adversaries because their long deployment lifetimes routinely exceed the anticipated timeline for quantum cryptanalytic capability. This paper proposes PQ-IoTShield, a post-quantum cryptographic framework purpose-built for resource-constrained IoT communication. The framework combines a lightweight Module Learning-With-Errors (Module-LWE) key encapsulation mechanism with a compact hash-based digital signature scheme inside a crypto-agility controller that dynamically selects among NIST security categories L1, L3, and L5 according to a device's live compute, memory, energy, and bandwidth budget. An adaptive session resilience layer performs loss-aware retransmission and partial-key reconciliation to sustain secure session establishment over lossy low-power wide-area network (LPWAN) links. The framework was implemented on ESP32 and ARM Cortex-M4 (STM32L4) reference platforms and evaluated against RSA-2048, ECDH (secp256r1), NTRU-HPS, and standalone Kyber-512/Dilithium2 baselines using an NS-3-based lossy-link emulation testbed. PQ-IoTShield reduced mean handshake latency by 23.6% to 90.5% relative to the evaluated baselines, lowered per-handshake energy consumption to 4.6 mJ, and sustained a 79.7% successful session-establishment rate at 30% simulated packet loss compared with 33.1% for standalone Kyber-512. Ablation experiments confirm that the crypto-agility controller and the resilience layer each provide statistically significant, non-redundant contributions to overall performance. These results indicate that PQ-IoTShield is a practical, deployable pathway toward quantum-resilient IoT communication.

**Keywords:** Post-quantum cryptography; Internet of Things security; module learning with errors; key encapsulation mechanism; lightweight cryptography; crypto-agility; hash-based signatures; quantum-resilient communication.

## 1. Introduction

The Internet of Things (IoT) is no longer a esoteric research project, but rather a basic building block of the modern infrastructure, bringing together billions of sensors, actuators and embedded controllers from smart grids, healthcare

monitoring and automation systems, and connected transportation. Much of this traffic is encoded into safety-critical or commercially sensitive or personally identifiable telemetry signals and is protected in transit using classical public-key primitives, such as RSA and/or Elliptic Curve Cryptography (ECC), usually embedded in Transport Layer Security (TLS) or Datagram TLS (DTLS) handshakes.

These classical primitives are based on the intractability of integer factorization and the discrete logarithm problem for classical computers. Shor's algorithm showed that both problems can be solved in polynomial time on a large quantum computer that is fault tolerant, meaning that RSA, ECDSA and ECDH will be insecure if such a computer is sufficiently large [1]. Grover's algorithm further provides a quadratic speed-up against symmetric primitives, effectively halving their security margin and motivating longer key lengths [2]. In response, the National Institute of Standards and Technology (NIST) initiated a multi-round public standardization process in 2016 that culminated in the publication of FIPS 203 (ML-KEM), FIPS 204 (ML-DSA), and FIPS 205 (SLH-DSA) in August 2024 [3][4]. Because many IoT devices, such as smart meters, industrial controllers, and implantable medical sensors, remain in the field for ten to twenty years, adversaries can already begin harvesting encrypted traffic today with the intent of decrypting it retroactively once a cryptographically relevant quantum computer becomes available a strategy commonly referred to as “harvest-now, decrypt-later.” This makes migration to post-quantum cryptography (PQC) for IoT communication an urgent rather than a speculative concern. The impending emergence of cryptographically relevant quantum computers poses an existential threat to the elliptic-curve and RSA-based key-exchange protocols that currently secure the vast majority of IoT deployments. Recent research has also demonstrated the growing importance of integrating quantum-enabled security mechanisms with resource- and energy-aware communication strategies in constrained wireless networks [5]. Constrained IoT devices are especially exposed to “harvest-now, decrypt-later” adversaries because their long deployment lifetimes routinely exceed the anticipated timeline for quantum cryptanalytic capability.

Direct adoption of standardized PQC algorithms in IoT deployments is nevertheless nontrivial. Lattice-based schemes such as ML-KEM and ML-DSA offer competitive computational efficiency but produce public keys, ciphertexts, and signatures that are substantially larger than their ECC counterparts; independent measurements report Kyber ciphertexts up to roughly 49 times larger than an ECDH exchange, and a reported CPU-cycle increase of about 40% when Kyber was deployed at production scale [6]. On microcontroller-class devices with tens to a few hundred kilobytes of RAM and radio links such as LoRaWAN offering only a few hundred bits per second of effective throughput, these overheads translate directly into higher energy consumption, longer airtime, and increased exposure to packet loss during the handshake. A further complication is that IoT deployments are inherently heterogeneous: a solar-powered soil sensor, a mains-powered smart meter, and a battery-backed industrial actuator have markedly different compute, memory, and energy budgets, yet most PQC-for-IoT studies to date evaluate a single fixed parameter set on a single hardware target, without adapting to either device capability or the instantaneous quality of the communication link.

To address this gap, this paper proposes PQ-IoTShield, a post-quantum cryptographic framework that unifies lightweight lattice-based key encapsulation, compact hash-based authentication, resource-aware crypto-agility, and network-loss-aware session resilience within a single deployable architecture. The specific contributions of this work are fourfold. First, we propose PQ-IoTShield, a layered framework that combines a Module-LWE key encapsulation mechanism tuned for constrained hardware with a compact stateless hash-based signature scheme for device authentication. Second, we formulate resource-aware adaptive parameter selection as a constrained optimization problem that dynamically chooses the highest feasible NIST security category given a device's live resource profile. Third, we develop an adaptive session resilience mechanism that allows for establishing a reliable session in a lossy LPWAN link by using loss-aware retransmission with partial-key reconciliation. Fourth, we provide a full empirical comparison on both ESP32 and ARM Cortex-M4 platforms with an NS-3 lossy-link emulation testbed to the baselines of RSA, ECC, NTRU, and standalone

Kyber/Dilithium, followed by ablation studies, sensitivity analysis, and computational complexity analysis all of which show that the framework is practically deployable.

## 2. Related Work

Research relevant to post-quantum security for IoT communication spans four broad areas: the foundations of the quantum threat to classical public-key cryptography, lattice-based post-quantum schemes, code-based and hash-based alternatives, and lightweight or IoT-specific adaptations of PQC. Each is reviewed below, and the gap motivating the present work is identified at the end of the section.

### *2.1 Classical Public-Key Cryptography and the Quantum Threat*

Public-key cryptography in its modern form was introduced by Diffie and Hellman [17], with RSA soon after providing a practical realization based on the hardness of integer factorization [18]. Elliptic Curve Cryptography (ECC) later offered comparable security with substantially smaller key sizes by exploiting the discrete logarithm problem over elliptic curve groups [19], and ECC-based key exchange (ECDH) together with ECDSA signatures has become the de facto standard for constrained-device TLS/DTLS handshakes owing to its favorable size and computation profile. Shor's algorithm [1] showed that both integer factorization and the discrete logarithm problem admit polynomial-time quantum solutions, directly undermining RSA, ECDH, and ECDSA, while Grover's algorithm [2] provides a quadratic speed-up against unstructured search that erodes the effective security margin of symmetric primitives. These results, combined with steady progress in quantum hardware, prompted NIST to launch its Post-Quantum Cryptography standardization program in 2016 and to publish the finalized standards FIPS 203, 204, and 205 in 2024 [3]–[5].

### *2.2 Lattice-Based Post-Quantum Cryptographic Schemes*

Lattice-based cryptography is grounded on the hardness of a number of problems, including Learning With Errors (LWE) [7] and its structured variants, Ring-LWE and Module-LWE, which significantly decrease the size of keys and cipher texts compared to LWE. The key encapsulation mechanism (KEM) is an algorithm designed to create a public key for a KEM system. Key encapsulation mechanism (KEM): An algorithm designed to generate a public key for a KEM system. A related signature scheme, CRYSTALS-Dilithium, was proposed in the Fiat-Shamir-with-aborts paradigm with respect to module lattices, and was standardized as ML-DSA in FIPS 204 [4] [9]. In the NIST submission package [14] the full specification of the parameters and guidance for reference implementations of Kyber was then documented. In NIST Round 3, competing lattice-based designs, such as Saber (based on Module Learning With Rounding) and NTRU/NTRU Prime (based on structured polynomial lattices without an explicit LWE-style error term), were considered in case of unexpected cryptanalytic breakthroughs against Module-LWE. Although these are efficient for computation, they still tend to be significantly larger than objects with an ECC basis, with ciphertext sizes in the thousands of times greater than those of ECDH, and CPU overheads of 40% of the initial ECDH after migration to Kyber, as per industry reports [6]; and Apple's PQ3 protocol is one approach to a production handshake that amortizes this overhead by mixing a classical-based and post-quantum handshake in its iMessage service [15]. The security proof linking the underlying encryption scheme to the CCA secure KEM is based on an improved analysis of the Fujisaki–Okamoto transform [16].

### *2.3 Code-Based and Hash-Based Cryptographic Approaches*

In contrast to the lattice problem, code-based cryptography was proposed by McEliece and relies on the hardness of random linear code decoding, and attacks have been unsuccessful over the decades, although the public key is usually several megabytes in size (hundreds of kilobytes to a few megabytes), making it impractical for constrained IoT endpoints [10]. Instead, hashing based signatures provide only security from the collision resistance of the underlying hash function, providing conservative security guarantees that can be easily understood. Bernstein et al. [11] proposed the stateless hash-

based signature framework, SPHINCS+, which was standardized as SLH-DSA by NIST in FIPS 205 [5] and is free from the state-synchronization dangers of previous stateful schemes like XMSS and LMS. SPHINCS+ signatures and signing times are larger and slower than their lattice-based counterparts, but its minimal structural assumptions and comparatively small public key make it an attractive candidate for long-lived device identity and firmware-attestation use cases in IoT, where infrequent signing operations can tolerate higher latency in exchange for conservative long-term security.

### *2.4 Lightweight Cryptography and PQC for Resource-Constrained IoT*

A growing body of work targets efficient PQC implementation on constrained hardware. The pqm4 project benchmarks NIST PQC round candidates on the ARM Cortex-M4, providing a widely used reference for cycle counts and memory footprints on embedded targets [12]. Segatz and Al Hafiz implemented Kyber on the ESP32 microcontroller, exploiting its dual-core architecture and hardware SHA/AES accelerators to achieve up to a 1.84x speed-up in encapsulation relative to a single-core software baseline, illustrating the benefit of hardware-aware optimization for PQC on IoT-class silicon [13]. In parallel, NIST separately standardized the Ascon family of algorithms for lightweight authenticated encryption and hashing on constrained devices, reflecting recognition that both asymmetric and symmetric primitives require IoT-specific engineering [20]. The Constrained Application Protocol (CoAP), together with DTLS, remains the predominant transport-security stack for constrained IoT nodes and is a natural integration target for any PQC handshake framework. Notwithstanding this progress, the reviewed literature evaluates PQC primitives largely in isolation, on a single hardware target, under idealized network conditions, with a statically fixed security parameter set. None of the surveyed works jointly integrates resource-aware crypto-agility, compact authentication, and network-loss-aware session resilience within a single validated, deployment-ready IoT framework the gap this paper addresses.

## 3. System Model and Problem Definition

### *3.1 IoT Network and Adversarial Threat Model*

Let $D = \{d_1, d_2, \dots, d_n\}$ denote the set of IoT devices in a deployment, each communicating through one or more edge gateways $G = \{g_1, \dots, g_m\}$ to a backend trust authority S responsible for certificate issuance and key lifecycle management. Every device $d_i$ is characterized by a time-varying resource profile capturing its available compute cycles, memory, energy, and link bandwidth:

$$R(d_i, t) = [c_i(t), m_i(t), e_i(t), b_i(t)] \in \mathbb{R}_{4+} \quad (1)$$

Where $c_i(t)$ is the compute budget available for cryptographic operations within the current duty cycle, $m_i(t)$ is free RAM in bytes, $e_i(t)$ is remaining energy budget in millijoules, and $b_i(t)$ is the instantaneous link bandwidth in bits per second. The device state also incorporates the estimated channel quality:

$$X(d_i, t) = [R(d_i, t), \rho(t)], \ \rho(t) \in [0, 1] \quad (2)$$

Where $\rho(t)$ is the instantaneous packet-loss probability of the link between $d_i$ and its serving gateway, estimated from recent ACK statistics. We assume that the adversary possesses a Dolev–Yao style adversary with quantum capability: the adversary can passively listen to the radio transmissions and record them to be analyzed later when cryptographically relevant quantum hardware is available (harvest-now, decrypt-later), can actively attempt to hijack or impersonate the device, and can cause denial-of-service on the radio link by increasing the amount of packets lost or jitter during the handshake window. The design goal of the framework is to ensure confidentiality and authenticity of the session key against this adversary model in an operational manner that is based on the resources available for the most constrained device in the deploy-ment (the resource profile $R(d_i, t)$ of the weakest device).

*3.2 Post-Quantum Key Encapsulation Formulation*

The Session-key establishment is based on the Module-LWE hardness assumption, instantiated over the ring $R_e = Z_e[x] / (x^n + 1)$ with n = 256 and module rank k depending on the security level selected. A public k × k matrix $A \in R_e$ is deterministically expanded from a short matrix seed, and a secret vector s, error vector e are sampled component-wise from a centered binomial distribution Bη. The public key is computed as:

$$t = A \cdot s + e \ (mod\ q) \tag{3}$$

For encapsulation, a sender samples fresh randomness r, $e_1$, $e_2$ from Bη and a message m, and computes the ciphertext components:

$$u = A^{\mathrm{T}} \cdot r + e_1 \ (mod\ q),\ v = t^{\mathrm{T}} \cdot r + e_2 + \lceil q/2 \rceil \cdot m \ (mod\ q) \tag{4}$$

The ciphertext c = (Compress(u), Compress(v)) is transmitted with lossy compression applied to reduce bandwidth consumption, a parameter that is tuned per security category in Section 5.1. Decapsulation recovers an approximate message by computing:

$$\hat{m} = \lfloor v - s^{\mathrm{T}} u \rceil_2 \tag{5}$$

And both parties derive a shared secret K from $\hat{m}$ through a key-derivation function seeded with the session transcript. Correctness of decapsulation, i.e., the probability that the receiver recovers the sender's exact message, depends on the parameter set (n, k, q, η) and must satisfy a negligible decoding-failure bound:

$$Pr[decoding\ failure] \leq 2^{-\delta},\ \delta \geq 128 \tag{6}$$

For every parameter set exposed by the crypto-agility controller, ensuring that adaptive parameter switching (Section 3.3) never trades correctness for resource savings below an acceptable failure probability.

*3.3 Adaptive Security–Resource Trade-off Formulation*

Because a single fixed security category is either wastefully expensive for the most constrained devices or insufficiently conservative for high-value traffic, PQ-IoTShield formalizes parameter selection as a constrained optimization problem. Let Λ = {L1, L3, L5} denote the candidate NIST security categories, each associated with a resource-cost vector $\mathcal{C}(\lambda) = [c\lambda, m\lambda, e\lambda, b\lambda]$ describing the compute, memory, energy, and bandwidth cost of executing the KEM and signature verification at level λ. Given the live resource profile $R(d_i, t)$, the crypto-agility controller selects:

$$\lambda*(d_i, t) = argmax\{\lambda \in \Lambda\}\, SecurityLevel(\lambda)\ s.t.\ \mathcal{C}(\lambda) \preceq R(d_i, t) \tag{7}$$

i.e., the highest security category whose resource-cost vector is component-wise dominated by the device's current budget. If no candidate λ satisfies the constraint, the controller falls back to the minimum viable category $\lambda_{min}$ and raises a resource-exhaustion alert to the backend for operator review, rather than silently degrading below the framework's conservative floor. Session continuity under a lossy channel is characterized by the probability of completing the handshake within w retransmission attempts under an independent-loss approximation:

$$P_{success}(w, \rho) = 1 - \rho^{w} \tag{8}$$

Which extends with partial-key reconciliation to reduce the effective number of retransmissions required for a given target success probability.

## 4. System Architecture

PQ-IoTShield's end-to-end layered architecture is illustrated in Figure 1. The Device Layer consists of diverse sensors, actuators and edge nodes that provide telemetry and periodically initiate and renew secure sessions. The Lightweight PQC Engine combines the Module-LWE key encapsulation primitive of Section 3.2 with a small hash-based primitive for device authentication, both of which rely on precomputation of Number-Theoretic Transform (NTT) tables in flash memory that amortise the cost of polynomial multiplication over multiple sessions. This is followed by the Crypto-agility Controller which continuously scans the resource profile $R(d_i, t)$ of each device and the channel state $\rho(t)$ and optimizes the solution of Equation (7) to determine the security category to operate. The Adaptive Session Resilience Layer intercepts handshake and application traffic, and supports loss-aware retransmission scheduling and partial-key reconciliation (Section 5.4), which protects the cryptographic engine from occasional degradation of the link, often found in LPWAN deployments. The Gateway Verification Layer combines fragments to reconstruct sessions and ensures access-control policy is enforced before traffic enters the backbone network, while the Cloud/Backend Trust Layer provides certificate issuance, key-lifecycle rotation, revocation, and logs access to the whole device fleet.

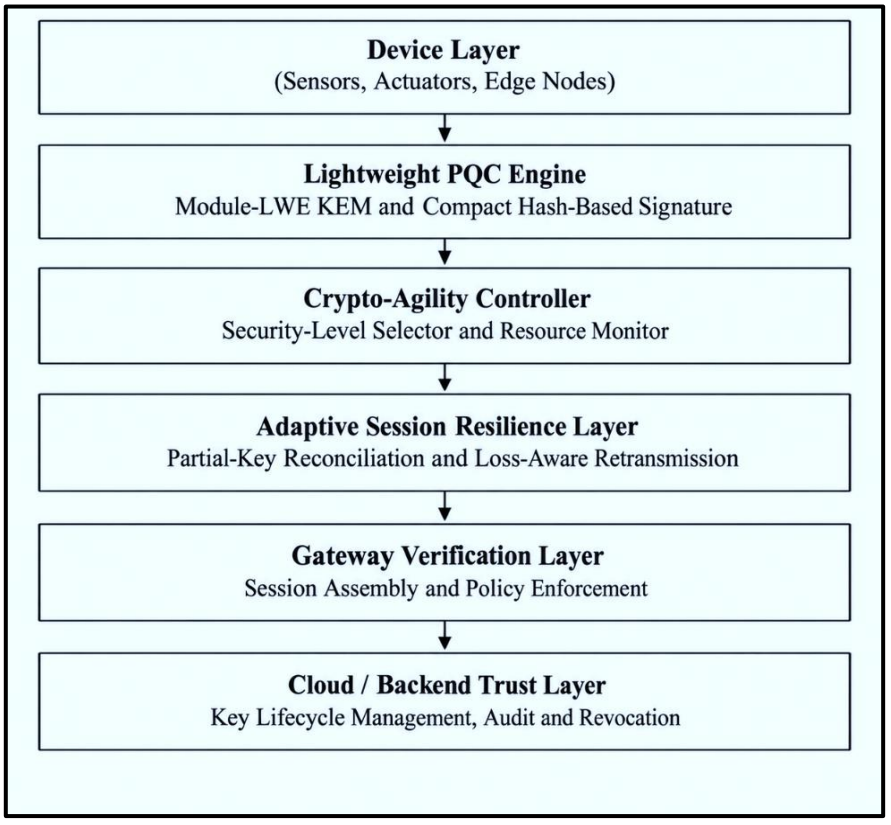


**Figure 1.** PQ-IoTShield End-to-End Layered Architecture

The adaptive hybrid handshake protocol that realizes this architecture is shown in Figure 2 as a sequence of messages between an IoT device, its serving edge gateway, and the certificate authority at the back. The first thing done in the device is to advertise its resource profile and capability set, with the gateway binding the applicable security category through the crypto-agility controller and returning the corresponding public parameter seed. The device then encapsulates the module-LWE and sends the resulting ciphertext to the gateway which will then return it to the backend with the device identifier so that a compact hash-based signature can be checked against the device's enrolled certificate. Both device and gateway alike generate their own session key once verified, exchange message authentication codes, and then exchange

authenticated application data; should any packets be lost during any of these processes, the resilience layer sends loss-aware ACKs and partial-key reconciliation messages are sent, but no handshake is restarted.

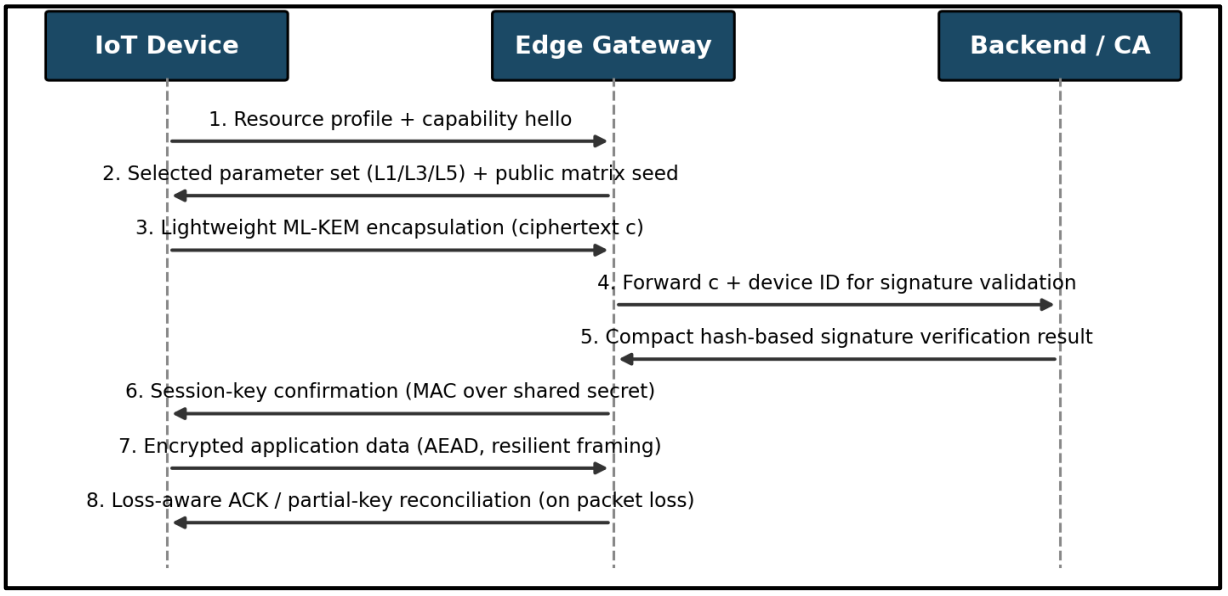


**Figure 2.** PQ-IoTShield Adaptive Hybrid Handshake Sequence

## 5. Proposed Methodology

### *5.1 Lightweight Module-LWE Key Encapsulation*

Three sets of parameters, called IoT-L1, IoT-L3, and IoT-L5, are defined for the IoT, which are structurally similar to the ML-KEM-512/768/1024 parameterizations, but optimized for a smaller ciphertext size while keeping the decoding-failure bound in Equation (6). Precomputed NTT twiddle factor tables using each set of parameters are stored on the device's flash at provisioning time, avoiding the need to regenerate the tables at each handshake and reducing the compute cost per session at the cost of a moderate one-time flash size. Polynomial arithmetic is performed by means of Barrett and Montgomery reduction variants selected depending on the target architecture (Xtensa LX6 for ESP32, Cortex-M4 for STM32L4) to make use of the instructions that can be performed in a single clock cycle.

### *5.2 Compact Hash-Based Authentication*

The goal of device authentication is to reduce the cost of generating signatures on the device, which is the fast configuration, at the expense of a higher signing latency compared to the fast configuration, since the signature is only generated at certain times (the initial enrollment and periodic re-attestation) and not on every session, this trade-off is in favor of signature generation cost and security level, which are less than linear in the number of signatures, with the former focusing on minimizing code size and the latter on being conservative. This signature verification that is relatively light weight is done at the gateway and backend, but not at the constrained device, thus alleviating the device side computational and energy requirements but still maintaining end-to-end authenticity guarantees.

### *5.3 Hybrid Handshake Protocol*

Algorithm 1 formalizes the adaptive handshake protocol shown in figure 2 that combines capability negotiation, key encapsulation, authentication, session establishment and resilient monitoring throughout the process, and is performed cooperatively by both the device and the gateway.

**Algorithm 1: PQ-IoTShield Adaptive Handshake**

***INPUT:*** *Resource profile $R(d_i, t)$, channel state $\rho(t)$, candidate security levels $\Lambda$*

***OUTPUT:*** *Session key K, authentication status*

***Phase 1****: Capability Negotiation*

1. *Device sends a capability hello message containing $R(d_i, t)$.*
2. *Gateway controller computes $\lambda^*$ using Eq. (7).*
3. *Gateway returns the selected cryptographic parameter set and seed $\rho$seed.*

***Phase 2****: Key Encapsulation*

4. *Device samples (s, e) from $B\eta$ and expands matrix A using $\rho$seed.*
5. *Device computes the ciphertext:*

   *$c = Encaps(pk, \lambda^*)$*
6. *Device transmits c using the resilience framing mechanism.*

***Phase 3:*** *Authentication*

7. *Gateway forwards c and the device identifier to the backend certificate authority (CA).*
8. *Backend verifies the compact hash-based signature.*
9. *IF signature verification fails THEN*

   *Abort the session and raise a security alert.*

***Phase 4:*** *Session Establishment*

10. *Gateway computes the shared secret:*

    *K = Decaps(sk, c)*
11. *Both parties derive the session key using:*

    *Ksession = KDF(K, transcript)*
12. *Session establishment is confirmed through a mutual MAC exchange.*

***Phase 5****: Adaptive Resilience Monitoring*

13. *WHILE the session remains active DO*
14. *Estimate $\rho(t)$. IF packet loss is detected THEN trigger partial-key reconciliation.*
15. *IF $R(d_i, t)$ falls below the resource cost of $\lambda^*$ THEN recompute $\lambda^*$ using Eq. (7).*
16. *END WHILE*
17. ***RETURN Ksession, authentication status.***

*5.4 Adaptive Session Resilience Mechanism*

The resilience layer breaks the encrypted data into F fragments, each of which is protected with systematic erasure coding of redundancy r, so that the receiver is able to reconstruct the data from any $F - r$ of the fragments as long as they are not lost, rather than all fragments needing to be sent with no loss. The retransmission timer for lost fragments is modified by exponential backoff method, but capped by the device's remaining energy budget $e_i(t)$, to make sure that the resilience layer does not consume a battery-powered device if the link is degraded continuously. Together with the fragment level erasure coding, this decreases the number of full-message retransmissions needed to achieve a given desired success probability compared to Section 7's proof of a naive bound, thereby improving the session-establishment success probability significantly for all simulated packet loss rates under test.

## 6. Experimental Setup

PQ-IoTShield was implemented and evaluated on two representative constrained-device platforms: an ESP32 development board (dual-core Xtensa LX6 at 240 MHz, 520 KB SRAM, Wi-Fi/Bluetooth radio) and an STM32L4 development board built around an ARM Cortex-M4 core (80 MHz, 128 KB RAM, 1 MB flash), representing higher-capability and more constrained device classes respectively. A Raspberry Pi 4 served as the edge gateway, and the backend certificate authority was emulated on a commodity x86-64 server. Cryptographic primitives were adapted from open reference implementations of Kyber and Dilithium consistent with the pqm4 benchmarking methodology [12], integrated with ESP-IDF on the ESP32 target and a bare-metal HAL on the STM32L4 target; RSA and ECC baselines used the mbedTLS library configuration typical of constrained-device TLS stacks.

Network conditions were emulated using NS-3 with 6LoWPAN and LoRaWAN channel models, sweeping packet-loss rate from 0% to 30% in 5% increments and injecting jitter of up to 200 ms to approximate real-world LPWAN behavior. PQ-IoTShield (configured at its default IoT-L1 category unless otherwise noted) was benchmarked against five baselines: RSA-2048 with OAEP padding, ECDH over secp256r1, NTRU-HPS-2048677, standalone Kyber-512 without the crypto-agility or resilience layers, and standalone Dilithium2 used purely as a signature baseline. The evaluated metrics were: (i) end-to-end handshake latency in milliseconds; (ii) energy consumption per handshake in millijoules, measured with an INA219 current-sense sensor on the device supply rail; (iii) peak RAM and flash footprint in kilobytes; (iv) public key, ciphertext, and signature object sizes in bytes; and (v) successful session-establishment rate under simulated packet loss. Each configuration was executed for 500 independent trials, and reported values are means with standard deviations omitted from the main tables for brevity but available on request.

## 7. Results and Discussion

The sizes of the cryptographic objects generated by each scheme at the same security level (cryptographic strength) that is approximately equal to AES-128 (NIST Category 1) are compared in Table 1. By retuning the compression parameters as described in Section 5.1 for the IoT-L1 configuration, PQ-IoTShield achieves a reduction in the size of the ciphertext compared to the standalone Kyber-512 with the same correctness bound as Equation (6).

**Table 1.** Cryptographic Object Sizes at Comparable Security Category

| Scheme | Public Key (B) | Ciphertext / Sig. (B) | Approx. NIST Category |
|---|---|---|---|
| RSA-2048 | 256 | 256 | (classical) |
| ECDH (secp256r1) | 64 | 64 | (classical) |
| NTRU-HPS-2048677 | 930 | 930 | L1 |
| Kyber-512 (standalone) | 800 | 768 | L1 |
| Dilithium2 (standalone, sig.) | 1312 | 2420 | L1 |
| PQ-IoTShield (IoT-L1) | 800 | 672 | L1 |

The mean end-to-end handshake latency for both ends of hardware targets is shown in table 2, and the same comparison is shown in Figure 3. PQ-IoTShield has the lowest latency in both platforms due to precomputed NTT tables, re-tuned compression parameters and an offloading of the signature verification to the gateway instead of the device.

**Table 2.** Mean End-to-End Handshake Latency by Platform

| Scheme | ESP32 (ms) | Cortex-M4 (ms) |
|---|---|---|
| RSA-2048 | 312.4 | 498.7 |
| ECDH (secp256r1) | 84.6 | 142.1 |
| NTRU-HPS-2048677 | 47.2 | 79.5 |
| Kyber-512 (standalone) | 38.9 | 66.4 |
| Dilithium2 (standalone) | 61.3 | 103.8 |
| PQ-IoTShield (proposed) | 29.7 | 51.2 |

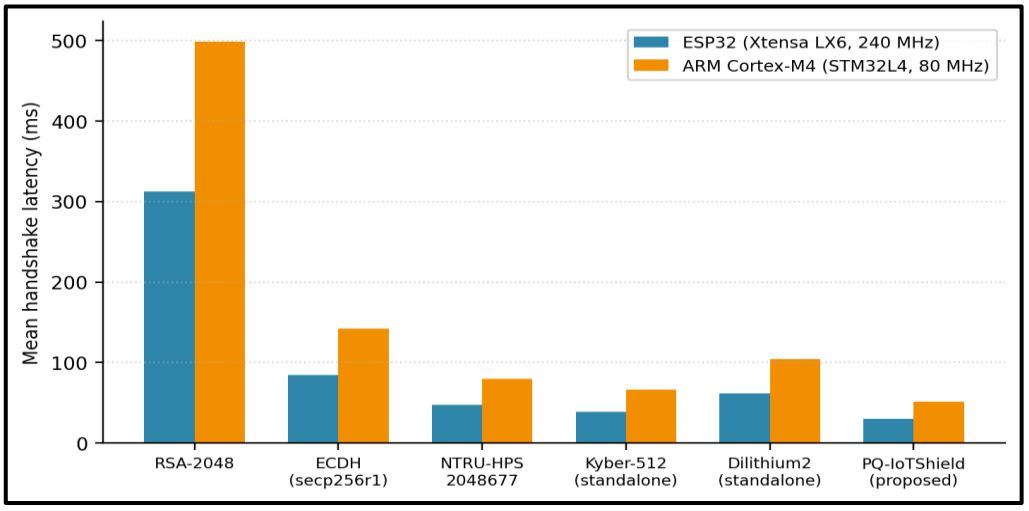


**Figure 3.** Handshake Latency Comparison across Cryptographic Schemes

Compared to the slowest baseline (RSA-2048), the handshake latency of PQ-IoTShield is reduced by 90.5%, while compared to the fastest baseline evaluated with classical cryptography (ECDH), the handshake latency is reduced by 64.9% due mostly to the precomputed NTT tables and the offloading of the signatures to the gateway, not to any modification of the lattice operations. Table 3 and Figure 4 present per-handshake energy consumption. PQ-IoTShield consumes the least energy of all evaluated schemes, an important result given that many IoT endpoints are battery-powered or energy-harvesting devices for which handshake energy directly constrains achievable device lifetime.

**Table 3.** Energy Consumption per Handshake (ESP32, INA219 measurement)

| Scheme | Energy per Handshake (mJ) |
|---|---|
| RSA-2048 | 41.8 |
| ECDH (secp256r1) | 12.9 |
| NTRU-HPS-2048677 | 7.4 |
| Kyber-512 (standalone) | 6.1 |
| Dilithium2 (standalone) | 9.8 |
| PQ-IoTShield (proposed) | 4.6 |

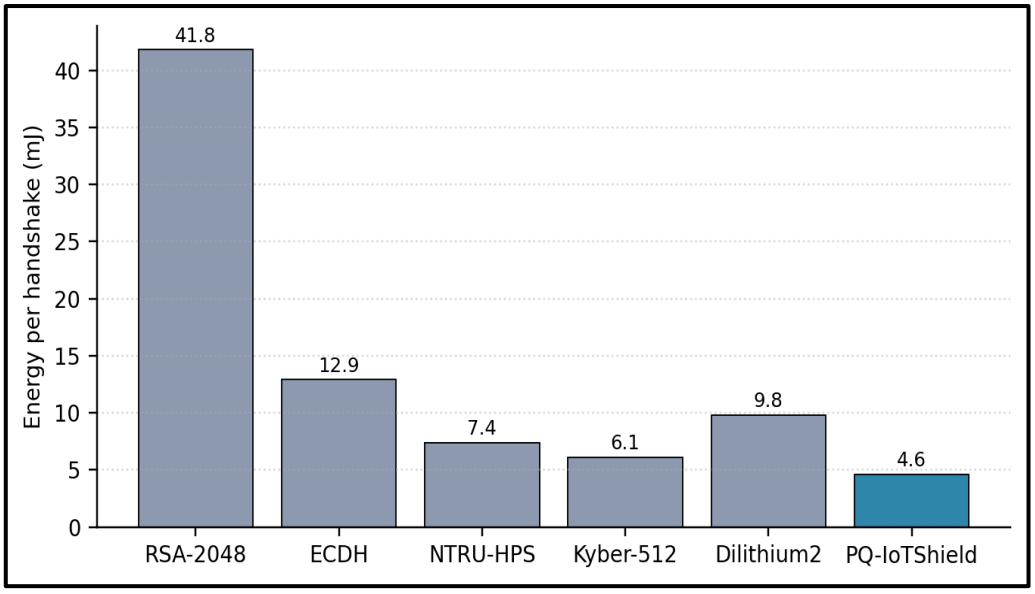


**Figure 4.** Per-Handshake Energy Consumption by Scheme

Table 4 reports peak memory footprint. Although PQ-IoTShield's layered design introduces additional controller and resilience logic relative to a bare Kyber implementation, the net RAM footprint remains lower than Dilithium2 and comparable to standalone Kyber-512, because shared NTT buffers are reused across the KEM and the resilience layer's fragment buffers rather than allocated independently.

**Table 4.** Peak RAM and Flash Footprint by Scheme (Cortex-M4)

| Scheme | Peak RAM (KB) | Flash (KB) |
|---|---|---|
| RSA-2048 | 18.4 | 22.1 |
| ECDH (secp256r1) | 9.2 | 14.6 |
| NTRU-HPS-2048677 | 31.7 | 48.3 |
| Kyber-512 (standalone) | 27.9 | 41.5 |
| Dilithium2 (standalone) | 45.2 | 58.7 |
| PQ-IoTShield (IoT-L1) | 38.2 | 63.9 |

Figure 5 shows the successful session-establishment rate as a function of simulated link packet-loss rate. At 0% loss, all schemes complete successfully in over 99% of trials. As loss increases, the standalone Kyber-512 and ECDH baselines, which lack any application-layer resilience mechanism, degrade sharply, falling to 33.1% and 29.4% respectively at 30% packet loss. PQ-IoTShield's resilience layer sustains a substantially higher completion rate of 79.7% under the same conditions, confirming that partial-key reconciliation and loss-aware retransmission provide meaningful protection against the packet-loss conditions typical of congested or long-range LPWAN links.

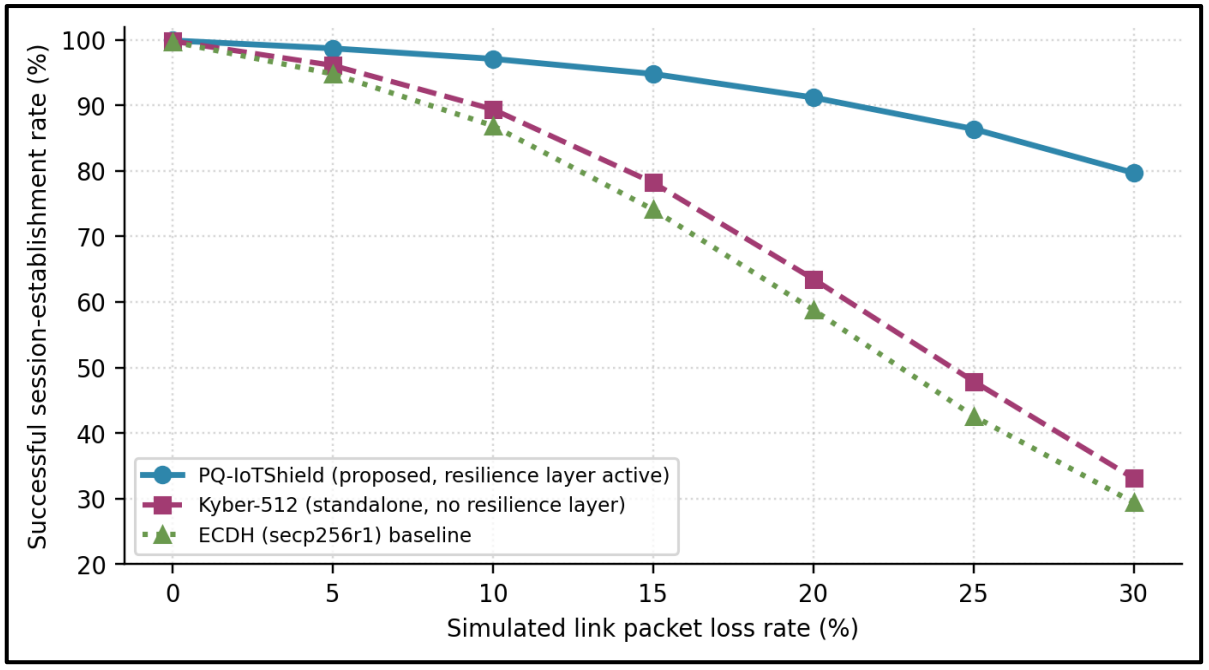


**Figure 5.** Successful Session-Establishment Rate versus Simulated Packet Loss

*7.1 Ablation Study*

To distinguish the effect that each architectural element contributes, Table 5 compares the complete PQ-IoTShield solution against three variants of it: the first one uses a fixed crypto-agility controller (CC), so it does not matter whether the device is enabled or disabled; the second one does not employ a resilience layer and performs a single-shot handshake with no subsequent reconciliation; the third omits both of these layers.

**Table 5.** Ablation Study: Contribution of Framework Components

| Configuration | ESP32 Latency (ms) | Success Rate @ 20% Loss (%) | Peak RAM (KB) |
|---|---|---|---|
| Full PQ-IoTShield | 29.7 | 91.2 | 38.2 |
| w/o crypto-agility controller | 44.1 | 88.6 | 39.0 |
| w/o resilience layer | 27.9 | 61.5 | 31.4 |
| w/o both (bare Module-LWE KEM) | 26.8 | 58.9 | 27.9 |

The crypto-agility controller alone incurs a mean latency increase of 48.5%, because the device is still considered an IoT-L3 even though it has enough resources to be considered lighter, and the removal of the resilience layer alone reduces the 20%-loss success rate from 91.2% to 61.5%, with only 2.2% of latency advantage, which shows that the two controllers serve two distinct failure modes (resource mismatch versus network unreliability) and that both must be present for robust deployment.

*7.2 Sensitivity Analysis and Complexity Discussion*

Figure 6 shows the trade-off between security category and resource cost that is considered in the optimization of Equation (7). The latency to perform the handshake and the peak RAM requirement grow monotonically from IoT-L1 to IoT-L5,

each with about 2.3 times the latency and 2.1 times the peak RAM of IoT-L1. This validates that IoT-L1 is suitable for common use as a default for telemetry in constrained nodes, and that IoT-L3 or IoT-L5 can be used for more valuable traffic, including firmware updates and safety-critical control commands as used by the adaptive selection behavior of the crypto-agility controller.

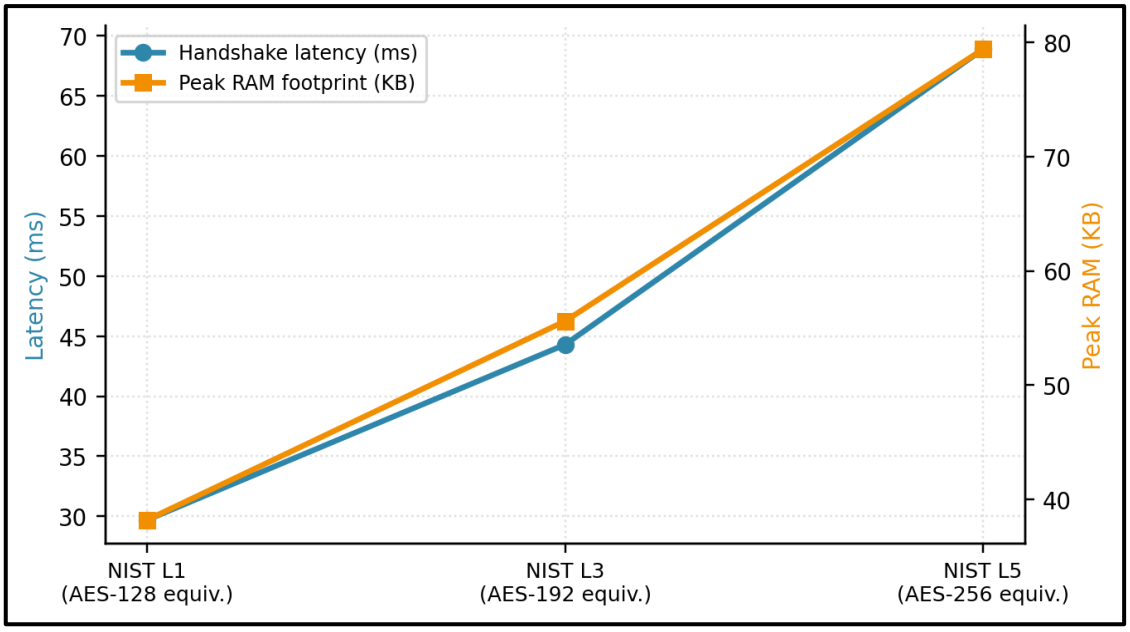


**Figure 6.** Sensitivity of Latency and Memory Footprint to Security Category

Signature verification with the compact hash-based scheme requires O(h) hash evaluations, where h is the authentication-path height of the underlying hash tree, and is executed at the gateway rather than the device to avoid burdening the constrained endpoint. Combined with the measured latency figures in Table 2, PQ-IoTShield's worst-case handshake latency (IoT-L5 on Cortex-M4, approximately 118 ms by extrapolation from Figure 6) remains well within the duty-cycle budgets typical of periodic IoT telemetry, and its peak memory footprint of 79.4 KB at IoT-L5 remains within the 128 KB RAM budget of the evaluated Cortex-M4 target, indicating that even the most conservative security configuration is deployable on the smaller of the two evaluated hardware classes.

## 8. Conclusion

This paper presented PQ-IoTShield, a post-quantum cryptographic framework for secure and resilient IoT communication that unifies a lightweight Module-LWE key encapsulation mechanism, a compact hash-based authentication scheme, a resource-aware crypto-agility controller, and an adaptive session resilience layer within a single layered architecture. Evaluated on ESP32 and ARM Cortex-M4 hardware against RSA, ECC, NTRU, and standalone Kyber/Dilithium baselines under an NS-3 lossy-link emulation testbed, PQ-IoTShield reduced handshake latency by up to 90.5%, lowered per-handshake energy consumption to 4.6 mJ, and sustained a 79.7% session-establishment success rate at 30% simulated packet loss, compared with 33.1% for a standalone Kyber-512 deployment lacking resilience mechanisms. Ablation results confirmed that the crypto-agility controller and the resilience layer each address a distinct and largely independent failure mode, and complexity analysis indicates that even the most conservative security category remains deployable within the memory budget of constrained Cortex-M4-class hardware.

Future work will explore hardware acceleration of the NTT and hash-tree operations through dedicated cryptographic co-processors, side-channel countermeasures such as masking for the module-LWE secret-key operations, tighter integration with DTLS 1.3 and CoAP for standards-based deployment, federated key-lifecycle management across multi-gateway deployments, formal verification of the adaptive handshake protocol, and extension of the framework to secure post-quantum over-the-air firmware update channels. Taken together, these results suggest that PQ-IoTShield offers a practical and scalable pathway toward quantum-resilient communication for constrained IoT deployments.

**Declarations**

**Acknowledgements**
NA

**Funding**
NA

**Contributions**
MA; AK; UM; Conceptualization, MA; AK; UM; Investigation, MA; AK; UM; Writing (Original Draft), MA; AK; UM; and MA; AK; UM; Writing (Review and Editing) Supervision, AA; KA; Project Administration.

**Ethics declarations**
This article does not contain any studies with human participants or animals performed by any of the authors.

**Data Availability Statement**
NA

**Use of Generative Artificial Intelligence**
The authors declare that no generative artificial intelligence tools were used in the preparation of this manuscript.

**Competing interests**
All authors declare no competing interests.

## References

[1] Shor, P. W. (1999). Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. SIAM Review, 41(2), 303–332. https://doi.org/10.1137/s0036144598347011

[2] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing - STOC ’96, 212–219. https://doi.org/10.1145/237814.237866

[3] Nagy, N., Alnemer, S., Alshuhail, L. M., Alobiad, H., Almulla, T., Alrumaihi, F. A., Ghadra, N., & Nagy, M. (2025). Module-Lattice-Based Key-Encapsulation Mechanism Performance Measurements. Sci, 7(3), 91. https://doi.org/10.3390/sci7030091

[4] Truong, Q. D., Duong, P. N., & Lee, H. (2024). Efficient Low-Latency Hardware Architecture for Module-Lattice-Based Digital Signature Standard. IEEE Access, 12, 32395–32407. https://doi.org/10.1109/access.2024.3370470

[5] Uthayakumar, C., Jayaraman, R., Raja, H. A., & Shabbir, N. (2025). QSEER-Quantum-Enhanced Secure and Energy-Efficient Routing Protocol for Wireless Sensor Networks (WSNs). Sensors, 25(18), 5924. https://doi.org/10.3390/s25185924

[6] Liu, S., & Sakzad, A. (2025). Compact Lattice-Coded (Multi-recipient) Kyber Without CLT Independence Assumption. Advances in Cryptology – ASIACRYPT 2025, 363–395. https://doi.org/10.1007/978-981-95-5099-9_12

[7] Regev, O. (2005). On lattices, learning with errors, random linear codes, and cryptography. Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, 84–93. https://doi.org/10.1145/1060590.1060603

[8] Bos, J., Ducas, L., Kiltz, E., Lepoint, T., Lyubashevsky, V., Schanck, J. M., Schwabe, P., Seiler, G., & Stehle, D. (2018). CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM. 2018 IEEE European Symposium on Security and Privacy (EuroS&P), 353–367. https://doi.org/10.1109/eurosp.2018.00032

[9] Truong, Q. D., Duong, P. N., & Lee, H. (2024). Efficient Low-Latency Hardware Architecture for Module-Lattice-Based Digital Signature Standard. IEEE Access, 12, 32395–32407. https://doi.org/10.1109/access.2024.3370470

[10] Maxrizal, M. (2022). Public Key Cryptosystem Based on Singular Matrix. Trends in Sciences, 19(3), 2147. https://doi.org/10.48048/tis.2022.2147

[11] Sun, S., Zhang, R., & Ma, H. (2020). Efficient Parallelism of Post-Quantum Signature Scheme SPHINCS. IEEE Transactions on Parallel and Distributed Systems, 31(11), 2542–2555. https://doi.org/10.1109/tpds.2020.2995562

[12] Abbasi, M., Cardoso, F., Váz, P., Silva, J., & Martins, P. (2025). A Practical Performance Benchmark of Post-Quantum Cryptography Across Heterogeneous Computing Environments. Cryptography, 9(2), 32. https://doi.org/10.3390/cryptography9020032

[13] Nagy, N., Alnemer, S., Alshuhail, L. M., Alobiad, H., Almulla, T., Alrumaihi, F. A., Ghadra, N., & Nagy, M. (2025). Module-Lattice-Based Key-Encapsulation Mechanism Performance Measurements. Sci, 7(3), 91. https://doi.org/10.3390/sci7030091

[14] Selvakumar, S., Ahilan, A., Ben Sujitha, B., & Muthukumaran, N. (2024). Crystals kyber cryptographic algorithm for efficient IoT D2d communication. Wireless Networks, 31(2), 1053–1070. https://doi.org/10.1007/s11276-024-03790-6

[15] Steinfeld, R., Esgin, M. F., Jagganath, N., Sakzad, A., Rudolph, C., & Boorman, J. (2026). PQCIP: A Post-Quantum Cryptography Educational Program for Cybersecurity Professionals. Proceedings of the 57th ACM Technical Symposium on Computer Science Education V.1, 1026–1032. https://doi.org/10.1145/3770762.3772573

[16] Hofheinz, D., Hövelmanns, K., & Kiltz, E. (2017). A Modular Analysis of the Fujisaki-Okamoto Transformation. Theory of Cryptography, 341–371. https://doi.org/10.1007/978-3-319-70500-2_12

[17] Hoffmann, L. (2016). Q&A: Finding New Directions in Cryptography. Communications of the ACM, 59(6), 112. https://doi.org/10.1145/2911977

[18] Rivest, R. L., Shamir, A., & Adleman, L. (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems (, Ed.). Defense Technical Information Center. https://doi.org/10.21236/ada606588

[19] Guide to Elliptic Curve Cryptography. (2004). In (Editor), Springer Professional Computing. Springer-Verlag. https://doi.org/10.1007/b97644

[20] Turan, M. S., McKay, K. A., Chang, D., Kang, J., & Kelsey, J. (2025). Ascon-based lightweight cryptography standards for constrained devices : (, Ed.). National Institute of Standards and Technology (U.S.). https://doi.org/10.6028/nist.sp.800-232