Feasibility of Lattice-Based Post-Quantum Cryptography on Resource-Constrained IoT Hardware: An ESP32 Evaluation

Javeria Khan, Sana Jamal Ansari, Khadijah Aamir Chaudhry
Syed Babar Ali School of Science and Engineering, LUMS

***Abstract*—Classical public-key cryptography such as RSA and ECC will not survive the advent of large-scale quantum computation, placing long-lived IoT deployments at risk of future decryption and identity compromise. This work evaluates the real-world feasibility of post-quantum cryptography on a constrained ESP32 microcontroller by benchmarking three lattice-based NIST selections: Kyber-512, Light-SABER, and Dilithium-II. Using PQClean reference implementations, expanded FreeRTOS execution stacks, and 100-iteration timing runs, we quantify performance, memory footprint, network overhead, and energy consumption for each scheme under practical embedded conditions.**

Our results show that Kyber-512 and SABER achieve sub-50 ms PQ key-exchange cycles with low energy cost, making them deployable for real-time IoT handshakes, while Dilithium-II provides strong authentication at significantly higher signing latency. Network analysis reveals that PQC inflation is severe for 64 B sensor packets but negligible for firmware-scale data, supporting a hybrid model where PQC secures session keys and symmetric ciphers carry bulk traffic. Timing analysis further confirms constant-time decapsulation in Kyber and SABER, while Dilithium-II exhibits measurable rejection-time leakage, underscoring the need for careful implementation beyond mathematical security. This study demonstrates that post-quantum protection on microcontrollers is achievable today, but optimal deployment depends on workload profile and security role.

Index Terms—Post-quantum cryptography, ESP32, Kyber-512, Light-SABER, Dilithium-II, IoT security, lattice-based cryptography, energy benchmarking, side-channel timing analysis, PQClean.

I. Introduction

Quantum computing poses a direct threat to classical public-key cryptosystems such as RSA and ECC, which can be efficiently broken using Shor's algorithm[1]. For long-lived deployments such as the Internet of Things (IoT), this threat is amplified: devices deployed today may remain operational for a decade or more, meaning encrypted data captured now could be decrypted later by quantum-capable adversaries. Without quantum-secure alternatives, confidentiality, integrity, and device authentication across IoT networks are at risk.

Modern embedded devices are particularly vulnerable because they operate under tight memory, energy, and communication bandwidth limits. As a result, the feasibility of post-quantum cryptography (PQC) must be evaluated not only in theory but on real hardware. The ESP32 serves as an ideal test platform due to its widespread adoption in consumer and industrial IoT, integrated Wi-Fi, low power consumption, and constrained RAM, making it a realistic stress case for PQC deployment.

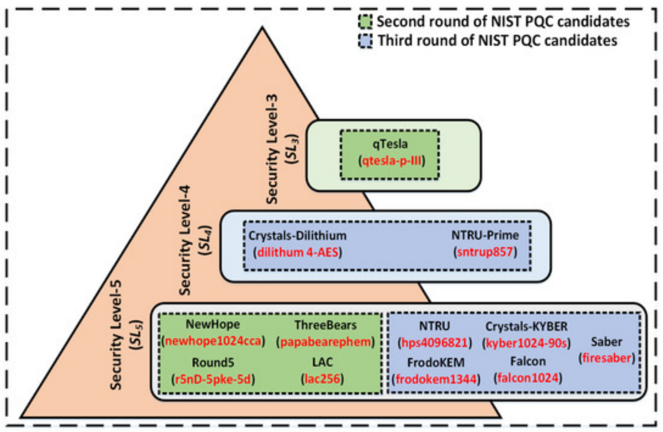

Second round of NIST PQC candidates
Third round of NIST PQC candidates
Security Level-3 (SL_3)
qTesla (qtesla-p-III)
Security Level-4 (SL_4)
Crystals-Dilithium (dilithum 4-AES)
NTRU-Prime (sntrup857)
Security Level-5 (SL_5)
NewHope (newhope1024cca)
ThreeBears (papabearephem)
Round5 (r5nD-5pke-5d)
LAC (lac256)
NTRU (hps4096821)
Crystals-KYBER (kyber1024-90s)
Saber (firesaber)
FrodoKEM (frodokem1344)
Falcon (falcon1024)

Fig. 1. Ranked Security Levels - NIST [3]

Lattice-based cryptography has emerged as the most mature and practical direction for quantum-resistant public-key systems. NIST has standardized **CRYSTALS-Kyber** for key establishment[2] and **CRYSTALS-Dilithium** for digital signatures, while **SABER**, a finalist that remains unbroken, continues to gain traction due to its low ciphertext size and competitive performance on constrained microcontrollers.

TABLE I
Selected Lattice-Based Schemes Evaluated on ESP32.

Scheme	**Type**	**NIST**	**Use in IoT**
Kyber	KEM	Standardized	Key Exchange
Dilithium	Signature	Standardized	Authentication
SABER	KEM	Finalist (not rejected)	Lightweight KEM

These three schemes were deliberately selected because they represent the most relevant post-quantum choices for embedded security; Kyber and Dilithium are now the official NIST PQC standards that future IoT deployments will rely on, and SABER serves as an important efficiency-oriented alternative that could outperform the standards on low-power edge hardware. Together, they reflect a realistic decision space for engineers deploying PQC in the field today.

This work provides one of the first comparative evaluations of Kyber-512, Light-SABER, and Dilithium-II on a resourcelimited ESP32. We benchmark all three across four IoT-critical dimensions:

- **Execution performance** (keygen, encaps/decaps, sign/verify latency)
- **Memory footprint** (RAM, stack, key and ciphertext sizes)
- **Network cost** (bytes exchanged per handshake or signature)
- **Energy consumption** (estimated from operation duration)

Our results quantify the practicality of deploying PQC at the edge and highlight trade-offs between signature-based and key-exchange primitives for scalable IoT security in the post-quantum era.

II. Background

The transition toward post-quantum cryptography (PQC) requires primitives that remain secure against quantum adversaries while remaining deployable on resource-constrained platforms. Among the leading candidates, lattice-based schemes are favoured because they provide strong asymptotic security, highly efficient polynomial arithmetic, and scalable parameter sets suitable for edge devices. Their security is based not on factorization or discrete logarithms, but on hard problems involving high-dimensional integer lattices.

A. Lattice Problems Underlying PQC

Modern PQC schemes rely on the hardness of solving certain problems in high-dimensional lattices. Unlike RSA or ECC, whose security is based on integer factorisation and discrete logarithms, lattice problems remain computationally resistant even under quantum algorithms such as Shor's and Grover's[1], [4]. Two key assumptions dominate current NIST-standardised approaches:

- **Learning With Errors (LWE):** Recovering a secret $\mathbf{s}$ from noisy linear equations $\mathbf{As} + \mathbf{e} \bmod q$ [5].
- **Short Integer Solution (SIS):** Finding a short vector $\mathbf{x}$ such that $\mathbf{Ax} = \mathbf{0} \bmod q$ [6].

Both problems scale exponentially in dimension n, and remain intractable for quantum computers, making them a core pillar for PQ security in IoT.

B. Mathematical Overview and Algorithm Selection

Lattice-based post-quantum schemes derive security from the hardness of Learning-With-Errors (LWE), Short Integer Solution (SIS), and their module variants MLWE/MLWR. Only the mathematical foundation relevant to implementation is summarized here.

Module-LWE (Kyber-512 KEM):

$$A \in \mathbb{Z}_q^{k\times k}, \quad \mathbf{s}, \mathbf{e} \leftarrow \chi, \quad \mathbf{b} = A\mathbf{s} + \mathbf{e} \pmod q$$

Recovering $\mathbf{s}$ remains infeasible even with quantum attacks. CRYSTALS-Kyber implements a KEM over MLWE using NTT-optimized polynomial multiplication for fast handshakes [7]. In this work, Kyber-512 is chosen as the lowest-cost standardized variant suited for IoT session key exchange.

Module-LWR (Light-SABER KEM):

$$\mathbf{b} = \left\lfloor \frac{A\mathbf{s}}{p} \right\rceil \pmod q$$

SABER removes explicit noise sampling and uses deterministic rounding, reducing bandwidth and memory footprint relative to MLWE [8]. Our implementation uses Light-SABER, the smallest parameter set, enabling practical deployment on microcontrollers where RAM constraints dominate. It serves as a direct comparative baseline against Kyber under equal experimental conditions.

Dilithium-II (LWE/SIS Signature):

$$A\mathbf{z} = \mathbf{t} \pmod q, \qquad \text{find } \mathbf{z} \neq 0 \text{ with } \|\mathbf{z}\| \leq \beta$$

Dilithium signatures rely on SIS for unforgeability while LWE ensures correctness [9]. Dilithium-II is selected to evaluate post-quantum authentication cost in IoT, enabling firmware verification and secure identity attestation at the expense of large signature output compared to KEM ciphertexts.

This unified mathematical background supports the comparative evaluation presented later, where Kyber-512 and Light-SABER are benchmarked as key-exchange mechanisms, while Dilithium-II is assessed as a post-quantum signature workflow for authentication-centric IoT applications.

C. Key Differences Between Algorithms

TABLE II
Comparative Characteristics of Evaluated PQC Algorithms

Feature	Kyber	SABER	Dilithium
Primitive	KEM	KEM	Signature
Mathematics	MLWE	MLWR	SIS/LWE
Primary Use	Key Exchange	Key Exchange	Authentication
Strength	NIST Standard	High Efficiency	Strong Integrity
Main Weakness	Memory Load	Timing variance	Large signature output

D. Visual Summary of Lattice Relations

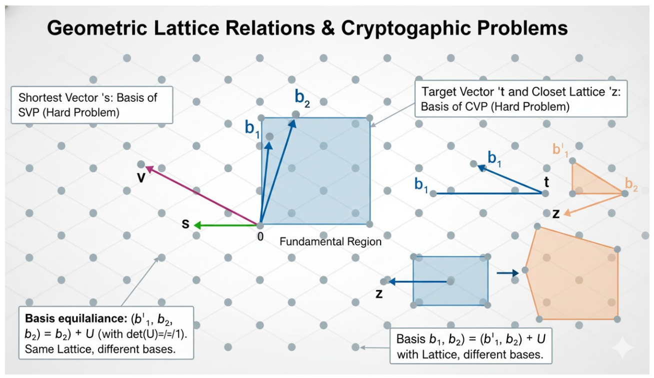

Geometric Lattice Relations & Cryptogaphic Problems
Shortest Vector 's: Basis of SVP (Hard Problem)
Target Vector 't and Closet Lattice 'z: Basis of CVP (Hard Problem)
b2
b1
v
s
0
Fundamental Region
b1
b1
t
b'1
b2
z
z
Basis equilaliance: (b'1, b2, b2) = b2) + U (with det(U)=/=/1). Same Lattice, different bases.
Basis b1, b2) = (b'1, b2) + U with Lattice, different bases.

Fig. 2. Geometric interpretation of lattice problems underpinning Kyber, SABER and Dilithium. The shortest vector (SVP) and closest vector (CVP) problems form the hardness basis for MLWE/MLWR KEMs and SIS/LWE signatures.

This figure provides a simple conceptual mapping to highlight where each algorithm originates within the broader lattice-based PQC family.

III. METHODOLOGY

This section describes the experimental workflow used to evaluate three lattice-based post-quantum cryptographic primitives on ESP32 hardware: Kyber-512, Light-SABER, and Dilithium-II. All results presented in later sections directly follow the methodology defined below.

A. Implementation Setup

All schemes were compiled and executed on an ESP32 DevKit V1 using the Arduino toolchain with FreeRTOS execution support [10]. Default Arduino task stack is insufficient for PQC due to large polynomial buffers, therefore each scheme was executed inside an independent FreeRTOS task with an expanded stack:

$$\text{Stack Size} = 45\text{–}64\,\text{kB} \quad \text{(allocated per algorithm)}$$

This modification was essential for stable execution and is one of the key engineering contributions of this work.

- **Development Environment:** Arduino IDE 1.8.x with ESP32 board support (v3.x), FreeRTOS task execution.
- **Source Integration:** All C/C++ reference implementations were ported manually as Arduino-compatible libraries.
- **RNG Adaptation:** CPU-side randombytes() was replaced with `esp_random()` / `esp_fill_random()` for hardware-backed entropy, ensuring cryptographic correctness on embedded hardware.
- **Measurement Precision:** Execution time was recorded using `micros()` with 1µs resolution. Serial output logging was disabled during timing to avoid IO skew.

Each algorithm was compiled independently, ensuring code size, memory cost, and runtime behaviour could be evaluated without cross-interference.

B. Benchmark Tasks

For fairness of comparison, all three algorithms were benchmarked under equal conditions:

$$\text{Runs per Operation} = 100 \quad \text{(averaged with min/max recorded)}$$

The following operations were executed:

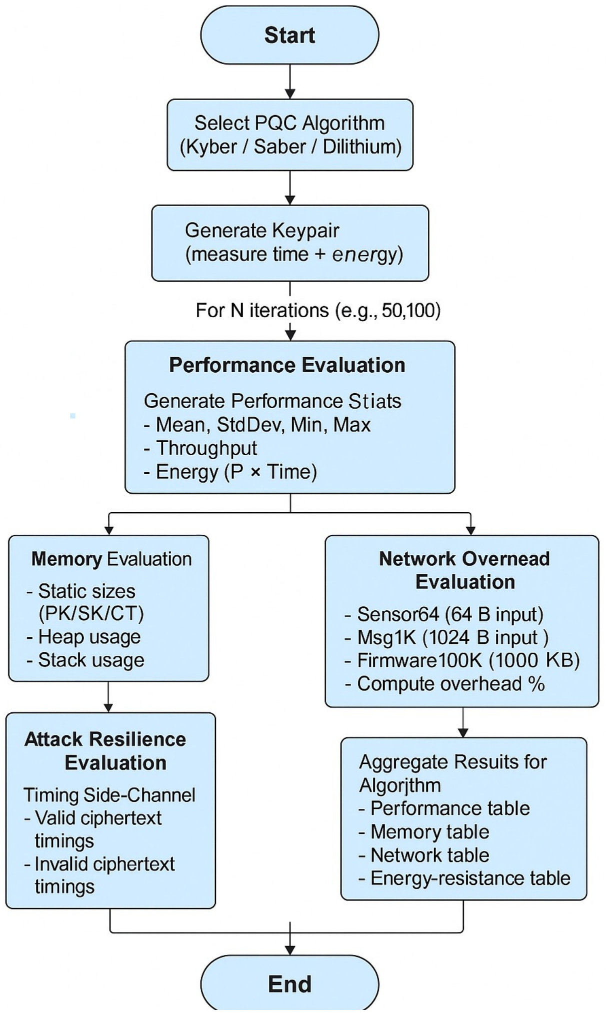
Scheme	Operation 1	Operation 2	Operation 3
Kyber-512	KeyGen	Encaps	Decaps
Light-SABER	KeyGen	Encaps	Decaps
Dilithium-II	KeyPair	Sign	Verify

TABLE III
BENCHMARK OPERATIONS MEASURED ACROSS THE THREE PQC SCHEMES.

The execution time, RAM consumption, and FreeRTOS stack usage were monitored for each run. Total end-to-end cost was calculated as:

$$T_{e2e} = T_{\text{KeyGen}} + T_{\text{Encaps}} + T_{\text{Decaps}} \quad \text{(KEMs)}$$

$$T_{e2e} = T_{\text{KeyPair}} + T_{\text{Sign}} + T_{\text{Verify}} \quad \text{(Signature)}$$



Start
Select PQC Algorithm (Kyber / Saber / Dilithium)
Generate Keypair (measure time + energy)
For N iterations (e.g., 50,100)
Performance Evaluation
Generate Performance Stiats
- Mean, StdDev, Min, Max
- Throughput
- Energy (P × Time)
Memory Evaluation
- Static sizes (PK/SK/CT)
- Heap usage
- Stack usage
Network Overnead Evaluation
- Sensor64 (64 B input)
- Msg1K (1024 B input)
- Firmware100K (1000 KB)
- Compute overhead %
Attack Resilience Evaluation
Timing Side-Channel
- Valid ciphertext timings
- Invalid ciphertext timings
Aggregate Resuits for Algorjthm
- Performance table
- Memory table
- Network table
- Energy-resistance table
End

Fig. 3. End-to-end methodology for PQC evaluation on ESP32 across performance, memory, network cost, energy, and side-channel timing leakage.

C. Side-Channel Timing Analysis

In addition to performance benchmarking, a timing side-channel test was conducted to evaluate susceptibility to decapsulation or verification leakage. Execution time was measured for valid and invalid operations:

$$\Delta T = |T_{\text{valid}} - T_{\text{invalid}}|$$

- For Kyber-512 and Light-SABER, valid vs. corrupted ciphertexts were decapsulated and compared.
- For Dilithium-II, valid vs. tampered signatures were verified.

A significant timing gap indicates potential side-channel leakage vulnerabilities aligned with real-world attacks such as KyberSlash [11]. This establishes the first security-oriented evaluation alongside performance metrics.

D. Evaluation Metrics

Four metrics were collected for all three schemes:

1) **Runtime Performance:** Mean, standard deviation, min/-max latency.
2) **Memory Usage:** RAM footprint, task stack consumption, key/ciphertext size.
3) **Network Overhead:** Total handshake bytes (KEM), signature expansion (Dilithium).
4) **Energy Consumption:**

$$E = I \times V \times t, \qquad I = 240\text{mA},\ V = 3.3\text{V}$$

This methodology ensures that results are reproducible under identical hardware and firmware configurations for all three post-quantum algorithms.

IV. Results and Evaluation

This section presents the empirical results collected from executing Kyber-512, Light-SABER, and Dilithium-II on ESP32 hardware. All measurements are derived from the controlled methodology established earlier, with 100 iterations per operation to ensure statistically meaningful values. PQC implementations were sourced from `PQClean` [12], ensuring NIST-conformant correctness and verified cryptographic behaviour.

A. Performance Evaluation

Figure 4 illustrates the execution cost for the three evaluated algorithms across their primary cryptographic operations. Key observations are summarised below.

- Light-SABER achieved the lowest total execution latency at 48.34 ms, with KeyGen = 12.59 ms, Encapsulation = 16.7 ms, and Decapsulation = 19.05 ms.
- Kyber-512 performed similarly, requiring 50.66 ms end-to-end. Slightly higher KeyGen and Decaps times push it marginally behind SABER, but both KEMs clearly fall within real-time responsiveness for IoT handshakes.
- Dilithium-II exhibits asymmetric operation cost: Signing is expensive (48.73 ms), while verification remains fast (14.82 ms). Its total cycle time (77.47 ms) exceeds both KEMs, reinforcing that Dilithium is best suited where signatures are generated infrequently but verified often (e.g., OTA firmware validation).

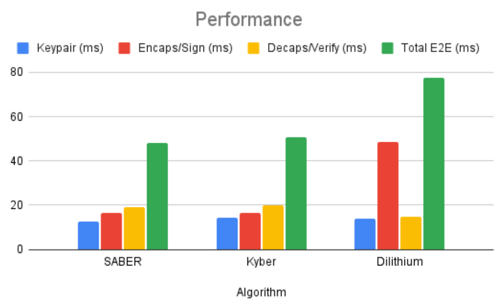


Fig. 4. Execution performance comparison of SABER, Kyber-512 and Dilithium-II on ESP32 across core cryptographic operations.

In summary, SABER and Kyber demonstrate near-identical computational profiles suitable for time-sensitive IoT key exchange, whereas Dilithium trades speed for strong signature functionality. These results confirm that lattice KEMs are deployment-ready for constrained devices [13], [14], while signature schemes require selective placement within the system trust chain.

B. Memory Consumption Analysis

Figure 5 compares static key material sizes and runtime memory usage for Dilithium-II, Light-SABER, and Kyber-512 on ESP32. The results highlight the impact of polynomial dimensionality and representation (LWE vs. LWR vs. SIS) on memory footprint.

Kyber-512 exhibits the highest minimum free-heap consumption (~285 kB), reflecting its larger polynomial structures and memory-intensive NTT-based multiplication. SABER shows a mid-range profile, while Dilithium-II demonstrates the lowest heap footprint among the tested schemes. In contrast, SABER reports zero dynamic heap allocation, indicating that its implementation uses only static and stack-based buffers; a strong advantage for deterministic embedded execution.

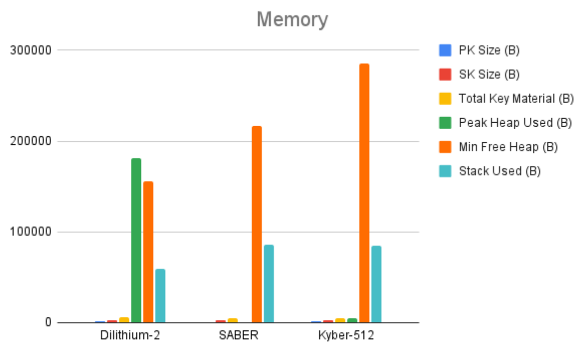


Fig. 5. Memory footprint comparison for Kyber-512, Light-SABER, and Dilithium-II on ESP32.

Static key sizes also follow expected patterns: Dilithium-II produces the largest public keys, Kyber-512 the largest secret keys, while SABER sits between both. These characteristics directly influence secure-storage cost and network bandwidth overhead in IoT deployments. All three algorithms consume substantial stack memory (59–86 kB), validating the necessity of FreeRTOS stack expansion for stable execution. Without this modification, PQC routines would crash due to polynomial buffer exhaustion [15].

C. Network Overhead Analysis

Figure 6 presents total transmitted bytes and relative overhead for Dilithium-II, Light-SABER, and Kyber-512 when securing three IoT message sizes: 64 B sensor data, 1 KB telemetry, and 100 KB firmware updates. The results show a clear scale-dependent behaviour. Large transfers are least affected, with firmware update sizes ranging from 101–104 KB across schemes, indicating that cryptographic material contributes under 2.4% overhead and is therefore insignificant for OTA updates or bulk data movement. For 1 KB messages, overhead remains moderate (106–236%), suggesting that authenticated telemetry is feasible with tolerable bandwidth expansion.

The challenge appears only at the smallest scale: PQC inflates 64 B sensor packets by 1700% (Kyber), 3250% (SABER), and 3781% (Dilithium). In high-rate sensor networks, this overhead compounds into substantial bandwidth and energy cost, making raw PQC-encrypted messages impractical without key amortization or batching. In practice, similar to RSA/ECDH today, PQC KEMs are best used to establish a session key once (e.g., AES-GCM), after which symmetric encryption carries the traffic efficiently [16], [17].

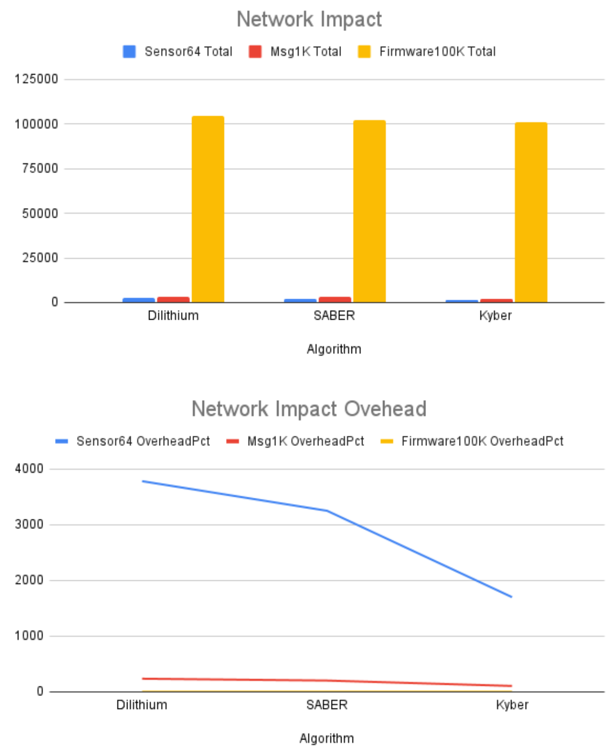


Fig. 6. Network cost and percentage overhead for three message scales under Dilithium-II, Light-SABER, and Kyber-512. Communication overhead is relative to original message size.

D. Energy Consumption Analysis

Energy cost was derived using measured execution time and the ESP32's active draw ($E = I \times V \times t$). Figure 7 compares operation-wise energy usage across all algorithms. The results reveal that Dilithium-II incurs the highest signing energy ($\approx$ 0.0276 J), driven by its long computation time, whereas verification remains lightweight ($\approx$ 0.0073 J). Kyber-512 shows the lowest per-operation energy across key generation, encapsulation, and decapsulation (0.0076–0.0106 J range), making it the most stable option for frequent key exchanges. SABER lies between the two extremes: its key generation and encapsulation are efficient (0.0134–0.0177 J), but decapsulation rises to $\approx$ 0.020214 J, suggesting a higher energy burden during repeated handshakes.

Overall, the energy results indicate application-specific suitability. Kyber-512 best fits continuous session refresh or periodic key rotation, SABER offers an efficient KEM profile with moderate decapsulation cost, and Dilithium-II is practical only when signatures are generated infrequently and verified often, such as firmware signing or device attestation workflows.

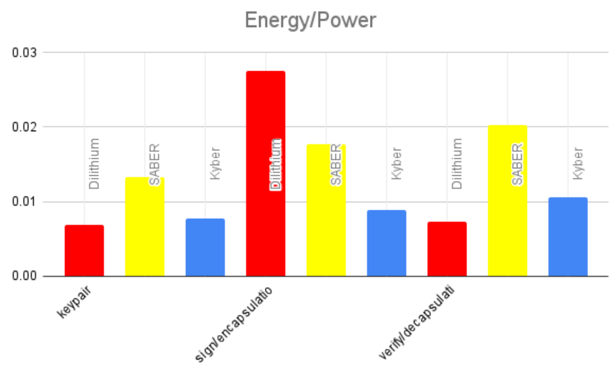
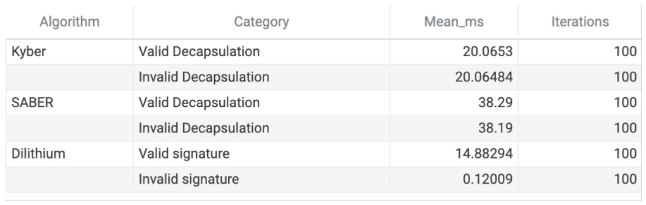


Fig. 7. Energy consumption across key operations.

E. Side-Channel Timing Analysis

The results reveal a sharp contrast between the KEMs and the signature scheme. Kyber shows effectively identical timings for valid and invalid decapsulation (20.07 ms vs. 20.06 ms), and SABER exhibits the same constant-time behavior (38.29 ms vs. 38.19 ms), ensuring that malformed inputs do not cause early termination. Dilithium-2, however, is not side-channel safe: valid signatures take about 14.88 ms to verify, whereas invalid signatures are rejected in roughly 0.12 ms. This 14.76 ms gap across 100 trials that can enable an attacker to easily distinguish valid from invalid signatures through timing analysis.

The `crypto_sign_verify_internal()` function in `sign.c` leaks timing information through early-exit checks that immediately terminate when detecting invalid inputs. These checks include signature length validation, structural validity verification, and polynomial norm checks. When any of these fail, the function returns early, skipping the expensive verification computations. Additionally, the final byte-by-byte comparison exits on the first mismatch rather than checking all bytes. This behavior explains the measured timing gap, which allows an attacker to distinguish valid from invalid signatures through timing alone. A constant-time implementation can mitigate this by inserting dummy operations or applying conditional masking so that all verification paths take the same amount of time, even for invalid inputs [18].



Algorithm	Category	Mean_ms	Iterations
Kyber	Valid Decapsulation	20.0653	100
	Invalid Decapsulation	20.06484	100
SABER	Valid Decapsulation	38.29	100
	Invalid Decapsulation	38.19	100
Dilithium	Valid signature	14.88294	100
	Invalid signature	0.12009	100

Fig. 8. Measured execution times for valid and malformed inputs across algorithms.

V. Conclusion

This work presents a practical evaluation of Kyber-512, Light-SABER, and Dilithium-II on a real ESP32 microcontroller, providing direct evidence of post-quantum feasibility under constrained memory, power, and bandwidth conditions. Results show that Kyber and SABER deliver fast and energy-efficient key establishment suitable for routine IoT handshakes, while Dilithium is better positioned for authentication and firmware integrity where signatures are generated infrequently and verified often.

Network and energy measurements confirm that PQC overhead is negligible for large firmware transfers but prohibitive for small telemetry packets, reinforcing the need for session-key establishment followed by symmetric encryption. Side-channel timing results additionally reveal that implementation security remains as critical as cryptographic strength [18], highlighting the necessity for constant-time verification paths in real-world PQC deployments. Overall, PQC is deployable on microcontrollers today, provided systems are designed to leverage each primitive where it fits best.

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," in *SIAM Journal on Computing*, vol. 26, no. 5, 1997, pp. 1484–1509.

[2] National Institute of Standards and Technology, "FIPS 203: Module-lattice-based key-encapsulation mechanism standard," https://csrc.nist.gov/pubs/fips/203/final, 2024, federal Information Processing Standards Publication.

[3] M. Imran, Z. U. Abideen, and S. Pagliarini, "An experimental study of building blocks of lattice-based NIST post-quantum cryptographic algorithms," *Electronics*, vol. 9, no. 11, p. 1953, 2020.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 1996, pp. 212–219.

[5] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," *Journal of the ACM*, vol. 56, no. 6, pp. 1–40, 2009, originally presented at STOC 2005.

[6] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, 1996, pp. 99–108.

[7] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Kyber: Algorithm specifications and supporting documentation," NIST PQC Round 3 Submission, 2017, available at https://pq-crystals.org/kyber/.

[8] J.-P. D'Anvers, A. Karmakar, S. Sinha Roy, and F. Vercauteren, "SABER: Mod-LWR based KEM," NIST PQC Round 3 Submission, 2018, available at https://www.esat.kuleuven.be/cosic/pqcrypto/saber/.

[9] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: Algorithm specifications and supporting documentation," NIST PQC Round 3 Submission, 2018, available at https://pq-crystals.org/dilithium/.

[10] Espressif Systems, "ESP32 technical reference manual," https://www.espressif.com/en/products/socs/esp32, 2023, version 4.6.

[11] K. Xagawa, A. Ito, R. Ueno, J. Takahashi, and N. Homma, "KyberSlash: Exploiting secret-dependent division timings in Kyber implementations," Cryptology ePrint Archive, Paper 2024/523, 2024, available at https://eprint.iacr.org/2024/523. [Online]. Available: https://eprint.iacr.org/2024/523

[12] PQClean Development Team, "PQClean: Clean, portable, tested implementations of post-quantum cryptography," https://github.com/PQClean/PQClean, 2023, accessed: 2024-11-28.

[13] U. Banerjee, A. Pathak, and A. Chandrakasan, "An energy-efficient configurable lattice cryptography processor for the quantum-secure internet of things," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2019, pp. 46–48.

[14] M. J. Kannwischer, J. Rijneveld, P. Schwabe, D. Stebila, and T. Wiggers, "PQClean: Clean, portable, tested implementations of post-quantum cryptography," in *NIST PQC Standardization Conference*, 2019.

[15] T. Fritzmann, G. Sigl, and J. Sepúlveda, "RISQ-V: Tightly coupled RISC-V accelerators for post-quantum cryptography," in *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, 2020, pp. 239–280.

[16] N. Bindel, U. Herath, M. McKague, and D. Stebila, "Transitioning to a quantum-resistant public key infrastructure," in *Post-Quantum Cryptography*, ser. Lecture Notes in Computer Science, vol. 10346. Springer, 2017, pp. 384–405.

[17] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Assessing the overhead of post-quantum cryptography in TLS 1.3 and SSH," in *Proceedings of the 16th International Conference on emerging Networking EXperiments and Technologies*, 2020, pp. 149–156.

[18] D. J. Bernstein, "Cache-timing attacks on AES," in *University of Illinois at Chicago*, 2005, technical report on timing attack countermeasures.