

A Security-Aware PQC Benchmarking Framework on Dual-Core Xtensa Silicon

^{1st} Gertrude Nabasirye
CyberMACS
Kadir Has University
Istanbul, Türkiye
gnabasirye@stu.khas.edu.tr

^{2nd} Hristina Mihajloska Trpcheska
Faculty of Computer Science and Engineering
Ss. Cyril and Methodius University
Skopje, North Macedonia
hristina.mihajloska@finki.ukim.mk

^{3rd} E. Fatih Yetkin
Dept. of Management Information Systems
Kadir Has University
Istanbul, Türkiye
fatih.yetkin@khas.edu.tr

Abstract—Deploying memory-heavy post-quantum cryptography (PQC) at the smart grid edge threatens real-time substation determinism, directly conflicting with the strict $\leq 3ms$ GOOSE tripping window of the IEC 61850 standard. To resolve this architectural tension, we propose a security-aware, hardware-in-the-loop benchmarking framework on the dual-core Xtensa LX7 (ESP32-S3) SoC running FreeRTOS. To prevent inter-core resource contention, the framework employs an asymmetric task-isolation design, pinning cryptographic operations to Core 1 while leaving Core 0 responsive to line-rate grid interrupts. Additionally, we define a programmatic heap-auditing methodology to model how `heap_4.c` maps key-state allocations across physical DRAM fragments while preserving the primary 128 kB contiguous system heap. Finally, we formulate an analytical latency model of the external Octal-SPI PSRAM interface, mathematically characterizing the CPU stall overhead (C_{stall}) induced when side-channel masking forces secret polynomial shares to spill over the external bus during non-sequential Cooley-Tukey NTT strides. This framework provides a rigorous methodology to characterize micro-architectural trade-offs, enabling utility engineers to plan post-quantum migration pathways without violating real-time grid constraints.

Index Terms—PQC, Smart Grid Security, ESP32, Xtensa, ESP-IDF

I. INTRODUCTION

The modernization of the electrical transmission and distribution grid has driven the deployment of digital technologies at the physical edge, such as Internet of Things (IoT) sensors and remote terminal units (RTUs). Operating over decades, these nodes must manage high-stakes telemetry under the strict real-time deadlines of the IEC 61850 substation standard, most notably the $\leq 3ms$ GOOSE fast-tripping latency window. However, current classical public-key cryptography (RSA and ECC) securing these channels is vulnerable to quantum attacks. This vulnerability introduces the threat of Harvest Now, Decrypt Later (HNDL) attacks, requiring immediate migration to NIST-standardized lattice-based algorithms like ML-KEM and ML-DSA [1], [2]. While mathematically secure, compiling these memory-intensive, polynomial-based primitives on resource-constrained 32-bit embedded platforms creates severe micro-architectural friction that directly conflicts with substation real-time determinism.

To deploy PQC at the grid edge, three fundamental system bottlenecks have to be addressed: real-time task starvation,

dynamic memory fragmentation, the PSRAM latency cliff. Heavy NTT-based PQC computations on a single core are starving from the required real-time processing in the networking stacks. The large PQC key and state space in particular causes severe memory fragmentation on the shared internal SRAM of such small microcontrollers which are running a hard multitasking RTOS [3]. Masking and other side-channel countermeasures for PQC increase the memory that is required for the key state (e.g., for NTT-based arithmetic in PQC) significantly, so that the state variables can no longer be stored in the internal SRAM of the processor. External Pseudo-Static RAM (PSRAM) has to be used instead, which causes severe CPU stall time due to the non-sequential memory access (strides) of the external memory [4].

To address these challenges, this paper proposes a security-aware, hardware-in-the-loop benchmarking framework designed for dual-core Xtensa LX7 (ESP32-S3) SoCs running FreeRTOS. Our primary contributions are three-fold:

- 1) We profile dynamic heap behaviors under FreeRTOS (`heap_4.c`) to demonstrate how key-state allocations preserve the primary contiguous 128 kB DRAM block.
- 2) We derive a cycle-accurate analytical model of the external Octal-SPI PSRAM interface to isolate the CPU stall penalty (C_{stall}) during non-sequential Cooley-Tukey NTT strides.
- 3) We propose an asymmetric task-affinity model, designed so that pinning PQC math to Core 1 preserves Core 0's responsiveness to meet the sub-4 ms GOOSE tripping window, and define a multicast-storm scenario for its evaluation.

The empirical validation of this framework (full latency, energy, and jitter profiles across the evaluated scheduling scenarios) will be reported in a more extended publication. This paper focuses on the architectural design, analytical models, and instrumentation methodology that make that evaluation possible.

The remainder of this paper is structured as follows. Section II reviews the algebraic foundations of NIST ciphers and side-channel masking. Section III details the analytical models of memory layouts, task isolation, and PSRAM bus stalls. Section IV presents the implementation and logging of our

benchmarking framework, and Section V concludes the paper.

II. LITERATURE REVIEW

A. The NIST PQC Standards

Quantum-resistant cryptography involves a fundamental change in the mathematical foundations that form the basis of current public-key cryptography. While traditional systems such as RSA and ECC are based on the hard problems of integer factorization and discrete logarithms (which are both easily solved by a cryptographically relevant quantum computer (CRQC) using Shor's algorithm), post-quantum systems are based on the computational hardness of lattice problems. We focus on the Module Learning with Errors (M-LWE) problem which is the basis of a Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM), standardized in FIPS 203 [1], and alongside the Module Short Integer Solution (MSIS) problem, the Module-Lattice-Based Digital Signature Standard, also standardized in FIPS 204 [2].

1) *The Algebraic Structure of Module-LWE*: Both ML-KEM and ML-DSA operate over a quotient polynomial ring defined as:

$$R_q = \mathbb{Z}_q[X]/(X^n + 1) \quad (1)$$

where the polynomial degree is fixed at $n = 256$. The primal M-LWE decision problem relies on distinguishing between the distribution $(\mathbf{A}, \mathbf{t} = \mathbf{A}\mathbf{s} + \mathbf{e})$ and a completely uniform distribution over $R_q^{k \times k} \times R_q^k$, where security scales directly with module rank k . For ML-KEM, $k \in \{2, 3, 4\}$ defines NIST Security Levels 1, 3, and 5. For ML-DSA, the dimensions (k, l) are set to $(4, 4)$, $(6, 5)$, or $(8, 7)$ for ML-DSA-44, ML-DSA-65, and ML-DSA-87 respectively.

While this shared algebraic structure allows us to represent keys and operations as vectors and matrices of polynomials over R_q , their coefficient moduli q differ: ML-KEM utilizes $q = 3329$ (a 12-bit modulus), whereas ML-DSA demands $q = 8380417$ (a 23-bit modulus). This difference is the primary driver of the micro-architectural execution gap. ML-KEM's 12-bit modulus allows efficient 16-bit packed arithmetic, enabling two coefficients to fit inside a single 32-bit register [5]. In contrast, ML-DSA's 23-bit modulus demands wider 32-bit variables and complex 32×64 -bit multiplications, significantly increasing the computational footprint of digital signatures on memory-constrained platforms like the Xtensa LX7 [6].

2) *Polynomial Multiplication and the Number Theoretic Transform*: The primary computational bottleneck in R_q is polynomial multiplication. To reduce naive complexity from $O(n^2)$ to $O(n \log n)$, both schemes utilize the Number Theoretic Transform (NTT) to map polynomials to the frequency domain for point-wise multiplication, though their mathematical pipelines diverge structurally. For ML-DSA, $q = 8380417$ supports a primitive 512^{th} root of unity, enabling the reducing polynomial $X^{256} + 1$ to completely factor into 256 degree-1 polynomials [2]. This permits a complete 8-layer NTT requiring 1024 butterfly operations. In contrast, ML-KEM's modulus $q = 3329$ only contains a primitive 256^{th} root of unity,

meaning $X^{256} + 1$ only factors into 128 quadratic polynomials [1]. Consequently, ML-KEM employs an incomplete 7-layer NTT running 896 butterflies, requiring more complex base multiplication of degree-1 polynomials modulo the quadratic factors in the frequency domain.

These NTT passes are executed via Cooley-Tukey (CT) forward and Gentleman-Sande (GS) inverse butterflies. On register-constrained 32-bit microcontrollers, loading intermediate coefficients and twiddle factors repeatedly creates severe register pressure, forcing the adoption of layer-merging strategies (e.g., 3-3-1 for ML-KEM or 3-3-2 for ML-DSA) to process multiple layers simultaneously within the registers [6], [7]. Furthermore, because embedded platforms lack fast hardware division units, software-emulated modular reduction is mandatory. Implementations rely on Montgomery reduction (scaling intermediates by 2^{-32}) or Barrett reduction to approximate quotients using precomputed constants and bit-shifts. For ML-KEM's 12-bit modulus, Plantard arithmetic offers a highly optimized alternative, expanding the safe input range by $2.14 \times$ to bypass intermediate reductions and reduce stack usage by up to 28% [6].

3) *Algebraic Mechanics of ML-DSA (FIPS 204)*: Unlike the predictable, constant-time execution of ML-KEM, ML-DSA is built on the Fiat-Shamir with Aborts framework, introducing non-deterministic execution through rejection sampling. During signing, the candidate signature $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$ must fall within a strict, secret-independent hypercube to prevent private-key leakage. If $\mathbf{z}$ or the signature hint violates geometric bounds, the signer aborts, resamples the mask $\mathbf{y}$ via $\text{ExpandMask}(\rho'', \kappa)$, and repeats the heavy matrix-vector multiplications $\mathbf{w} \leftarrow \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\mathbf{y}))$. Under standard configurations (ML-DSA-65), this loop aborts an average of 5.1 times, introducing severe latency jitter and energy spikes on battery-operated nodes.

To generate a signature $(\mathbf{z}, c)$ under FIPS 204 [2], the pipeline maps as follows:

- 1) **Sample Vector $\mathbf{y}$** : Coefficients are chosen uniformly from a small range bounded by γ_1 via ExpandMask :

$$\mathbf{y} \leftarrow \text{ExpandMask}(\rho'', \kappa)$$

- 2) **Compute Matrix Product**: Compute $\mathbf{w} \leftarrow \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\mathbf{y}))$.
- 3) **Extract High Bits**: Isolate the high-order bits of w to form the commitment: $\mathbf{w}_1 \leftarrow \text{HighBits}(\mathbf{w})$.
- 4) **Generate Challenge Hash c** : Hash the message representative μ alongside the byte-encoded commitment $\mathbf{w}_1$ to produce a sparse challenge polynomial $c \in R_q$ via SampleInBall :

$$\begin{cases} \tilde{c} \leftarrow \text{H}(\mu \parallel \text{w1Encode}(\mathbf{w}_1), \lambda/4) \\ c \leftarrow \text{SampleInBall}(\tilde{c}) \end{cases} \quad (2)$$

where c has exactly τ coefficients equal to ± 1 and the rest are zero.

- 5) **Compute Candidate $\mathbf{z}$** : Let $\mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s}_1$.
- 6) **Apply Rejection Test**: The algorithm computes the low bits $\mathbf{r}_0 \leftarrow \text{LowBits}(\mathbf{w} - c\mathbf{s}_2)$ and the hint vector

$\mathbf{h} \leftarrow \text{MakeHint}(-c\mathbf{t}_0, \mathbf{w} - cs_2 + c\mathbf{t}_0)$. It will abort and restart the entire loop if any of the following bounds are violated:

- If $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$
- If $\|\mathbf{r}_0\|_\infty \geq \gamma_2 - \beta$
- If $\|c\mathbf{t}_0\|_\infty \geq \gamma_2$
- If the total number of 1's in the hint vector $\mathbf{h}$ is greater than ω .

This complex bounding structure makes post-quantum signatures significantly more resource-demanding than key encapsulation mechanisms.

B. Computational Cost of Side-Channel Resistance

Security measures for field deployments extend past mathematical protection because devices require protection against physical side-channel attacks (SCA). The ESP32 experiences power and electromagnetic (EM) emission disturbances during every data-dependent register transition which enables attackers to link secret-dependent data with leakage through Hamming weight models [8].

Masking is the most common defense, but it introduces a multiplicative slowdown. The first-order masking technique which protects lattice-based KEMs results in a $3.01\times$ performance reduction when these systems operate on ARM-based microcontrollers [4]. The implementation of third-order masking techniques leads to a $9\times$ longer execution time and it causes dynamic memory usage to exceed 30 kB [9]. The restricted memory capacity of Class-1 IoT nodes prevents them from handling this increasing amount of data which makes 1^{st} or 2^{nd} -order masking the only secure implementation option [4].

However, there is a micro-architectural cost to this memory expansion that theoretical side-channel studies routinely overlook. Hardening techniques, such as those analyzed by Bronchain and Cassiers [10], focus heavily on mathematically optimizing bitsliced Keccak-f[1600] routines or Boolean-to-arithmetic conversions. While these methods secure the algorithm in theoretical leakage models, they duplicate the active state variables and secret shares. On a memory-constrained microcontroller, this duplication causes the active footprint to exceed internal SRAM limits. When the operating system is forced to transition these variables over an external memory bus to PSRAM, the system hits a high-latency bottleneck that current physical security literature does not model.

C. Comparative Literature Synthesis and Gaps

To clarify the operational boundaries of existing literature and justify our benchmarking framework, Table I synthesizes key post-quantum profiling studies.

This synthesis highlights two primary trends and a subsequent evaluation gap in current PQC literature. First, low-overhead mathematical optimizations, such as the bitslicing techniques by Bronchain & Cassiers (2022) [10] and the improved Plantard arithmetic by Huang et al. (2024) [6], are rigidly tailored for single-threaded, bare-metal execution. While highly efficient, this assumes an isolated memory map

and leaves the impact of context-switching, cache pollution, and interrupt latency from multitasking operating systems completely untested.

Second, high-level system evaluations frequently suffer from a severe high-resource platform bias. For example, the comprehensive benchmark of post-quantum TLS 1.3 handshakes by Abbasi et al. (2025) [12] defines its baseline resource-constrained edge device as a Linux-capable ARM Cortex-A53 node equipped with 1 GB of RAM. While highly valuable for demonstrating multi-threaded scalability on rich platforms, this configuration completely masks the true physical memory boundaries found in typical IoT microcontrollers. A 1 GB memory pool trivially absorbs the footprint of PQC operations, failing to reflect the severe heap fragmentation, register spilling, and algorithmic compromises required to fit these schemes onto deeply constrained devices, for example the 16 kB SRAM RISC-V boards targeted by Huang et al. (2024) [6].

Our work bridges this gap. By analyzing these trends, we identify three critical research gaps that our framework is explicitly designed to address:

- 1) **The Dynamic Heap and DRAM Allocation Gap:** Bare-metal evaluations typically assume that a cryptographic task enjoys exclusive control over SRAM, ignoring how memory-intensive post-quantum allocations navigate fragmented real-time operating system (RTOS) environments. To address this gap, our framework implements a dynamic heap-auditing methodology on the ESP32-S3. By querying the ESP-IDF heap capabilities API during an active NIST Level 5 session ($M_{\text{session}} = 6,304$ Bytes), we analyze how the underlying `heap_4.c` allocator distributes key-state variables, evaluating whether it can satisfy these high-security parameters using scattered memory fragments or if it forces a reduction in the primary 128 KB (131,072 Bytes) contiguous system heap.
- 2) **The Asymmetric Core Affinity and Multicast Storm Resilience Gap:** Multi-core benchmarks often treat symmetric processing cores as uniform computational pools, failing to reflect the real-time demands of substation automation. Specifically, the IEC 61850 standard dictates a strict ≤ 3 ms Class P2/P3 GOOSE protective tripping window where dynamic processing jitters can cause system failures. We address this gap by evaluating post-quantum cryptographic (PQC) execution across three FreeRTOS scheduling models: isolated task pinning, unpinned dynamic routing, and inter-core bus contention under a simulated substation multicast storm. This allows us to measure precise execution jitter using the coefficient of variation.
- 3) **The Physical Bus Latency of Masked Emulation Gap:** Physical side-channel resistance is a mandatory requirement for field-deployed devices, but the memory expansion of first-order masking risks forcing active variables to spill over onto an external Octal-SPI PSRAM. Existing cryptographic literature lacks empirical data on

TABLE I
STATE-OF-THE-ART PQC BENCHMARKING COMPARISON

Study	Hardware Platform	PQC Primitives	Identified Operational Gaps
Huang et al. (2024) [6]	ARM Cortex-M3, RISC-V	CRYSTALS-Kyber (ML-KEM)	Assumes absolute, exclusive control over SRAM; ignores RTOS multi-threading, context-switching noise, and communication stack co-existence.
Nagy et al. (2025) [11]	ESP32, ESP8266, Raspberry Pi 4B, Laptop	ML-KEM (512/768/1024)	Outlines binary failures due to physical RAM limitations, but does not provide dynamic, RTOS-aware stack high-water mark profiling or energy-to-solution models.
Abbasi et al. (2025) [12]	Heterogeneous (Xeon Gold Server, Laptop i5, Cortex-A53)	Kyber, NTRU, BIKE, Dilithium, Falcon	Constrained target runs full Linux (Debian 11) with 1 GB of RAM; does not evaluate deeply constrained microcontrollers, RTOS heap fragmentation, or external memory bus overheads.
Bronchain & Cassiers (2022) [10]	ARM Cortex-M4	Kyber-768, Saber	Ignores the physical impact of dynamic OS workloads or the energy/latency penalty when masked state duplication forces data across an external bus.
Proposed Work	ESP32-S3 (Xtensa Dual-Core 32-bit LX7)	ML-KEM-768, ML-DSA-44, AES-256-GCM	Profiles contiguous DRAM limits under FreeRTOS, models the physical penalty of the OPI PSRAM Cliff, measure energy and models battery effects, and evaluates asymmetric task isolation.

the resulting bus latencies and CPU stall penalties. We address this gap by developing a hardware-in-the-loop (HIL) emulation layer that applies calibrated latency scaling and memory footprint duplication to model register pressure and pipeline stalls. This enables us to isolate the physical CPU stall penalty (C_{stall}) induced when non-sequential Number Theoretic Transform (NTT) strides access external memory.

III. ANALYTICAL MODELING OF MICRO-ARCHITECTURAL GAPS

To deploy post-quantum cryptography (PQC) at the smart grid edge without breaching real-time substation constraints, we must model the micro-architectural limits of the underlying dual-core Xtensa silicon. This section outlines the analytical formulations defining three critical execution bottlenecks.

A. micro-architectural DRAM Bank Allocation Dynamics

Standard post-quantum implementations assume a single-threaded, bare-metal environment with absolute, exclusive control over SRAM. However, real-world edge nodes must run a real-time operating system (RTOS) while managing high-frequency communications via network buffers. Juggling these subsystems concurrently places immense pressure on the memory subsystem, where the physical layout of the silicon dictates allocation safety.

The ESP32-S3 contains 512 kB of total internal SRAM, but the memory available for data (DRAM) is limited to 320 kB and mapped across separate physical memory banks rather than a single continuous silicon block. The ESP-IDF memory manager utilizes an allocator based on the `heap_4.c` model, which tracks free memory as a linked list of blocks sorted by size. The largest single contiguous free block of internal 8-bit capable DRAM is capped by this physical system layout at exactly 131 072 kB [13], [14].

We model the total dynamic memory required for a post-quantum key-state session (M_{session}) as:

$$M_{\text{session}} = S_{\text{pk}} + S_{\text{sk}} + S_{\text{ct}} \quad (3)$$

where S_{pk} is the public key size, S_{sk} is the secret key size, and S_{ct} is the ciphertext payload size. For a high-security NIST Level 5 (ML-KEM-1024) session, this dynamic footprint is:

$$M_{\text{session}} = 1,568 \text{ B} + 3,168 \text{ B} + 1,568 \text{ B} = 6,304 \text{ Bytes} \quad (4)$$

Let the physical heap be represented as a set of scattered memory fragments $f_i \in F$ across secondary memory banks, and let M_{contig} represent the primary, contiguous 128 KB system block. Because the individual key-state structures are small relative to the primary block ($S_j \ll M_{\text{contig}}$), the allocator does not slice into the largest contiguous block. Instead, it satisfies these requests using the smaller pre-existing fragments:

$$\sum S_j \subset \bigcup f_i \quad \text{where} \quad f_i \cap M_{\text{contig}} = \emptyset \quad (5)$$

This allocation pattern ensures that the primary contiguous block remains completely unchanged ($M_{\text{contig}}(t_1) = M_{\text{contig}}(t_0) = 131,072 \text{ Bytes}$) even as the total free internal DRAM drops by exactly 6304 B. Formulating this bank-allocation behavior proves that smart grid gateways can execute high-security handshakes inside fast internal SRAM without fragmenting or depleting the primary system heap.

B. Multi-Core SMP Parallelization vs. Asymmetric Task Isolation

Attempting to accelerate single-threaded lattice algorithms by distributing arithmetic kernels, like the Number Theoretic Transform (NTT), across symmetric dual cores degrades performance. Microcontrollers lack hardware cache-coherence engines and multi-banked SRAM designed for parallel write access, meaning concurrent memory operations trigger severe

pipeline stalls. Furthermore, emulating 64-bit Keccak operations on 32-bit registers creates massive instruction bloating, making software synchronization overhead exceed any computational gains.

We resolve this by implementing an asymmetric task-affinity model as shown in Figure 1. We use the FreeRTOS scheduler to partition responsibilities at the operating system level:

- **Core 0 (System Core):** Handles low-latency interrupts, such as the sub-4 ms IEC 61850 GOOSE tripping loop, and the network stack.
- **Core 1 (Cryptographic Core):** Runs the heavy, high-latency post-quantum handshakes in complete isolation.

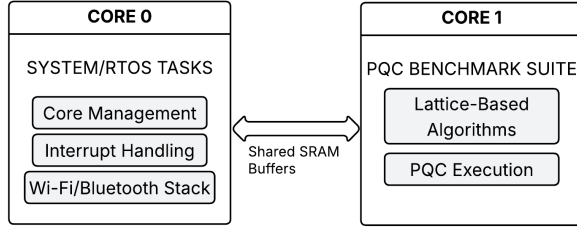


Fig. 1. Asymmetric Task-Affinity Model

This core-pinning design guarantees that heavy lattice arithmetic cannot starve high-priority real-time grid tasks.

C. PSRAM Cliff Latency Model

Side-channel countermeasures like masking double memory footprints by replicating secret polynomial states. This footprint expansion easily breaches internal SRAM, forcing state variables across an external Octal-SPI (OPI) bus, a performance bottleneck we define as the PSRAM Cliff.

We model the total post-quantum execution latency (T_{total}) as:

$$T_{\text{total}} = T_{\text{core}} + \Delta T_{\text{spill}} \quad (6)$$

where T_{core} is the ideal SRAM execution time and ΔT_{spill} is the external bus penalty. The spillover delay is driven by L1 cache-line invalidation:

$$\Delta T_{\text{spill}} = N_{\text{miss}} \times \left(\frac{C_{\text{stall}}}{f_{\text{CPU}}} \right) \quad (7)$$

where N_{miss} is the cache-miss count triggered by the non-sequential memory strides of the Cooley-Tukey NTT, f_{CPU} is the 160 MHz core clock, and C_{stall} is the cycle penalty per miss:

$$C_{\text{stall}} = \left(\frac{f_{\text{CPU}}}{f_{\text{bus}}} \times \left(N_{\text{cmd}} + N_{\text{addr}} + N_{\text{setup}} + N_{\text{dummy}} + N_{\text{data}} \right) \right) + N_{\text{overhead}} \quad (8)$$

Here, f_{bus} is the 80 MHz DDR bus, N_{cmd} is the command phase bus cycles, N_{addr} is the address phase bus cycles, N_{setup} represents hardware bus arbitration and interface setup cycles,

N_{dummy} is the address decoding latency cycles, N_{data} is the physical bus cycles needed to fetch a 32-byte cache line over an 8-bit wide OPI interface, and N_{overhead} represents internal pipeline and state machine resynchronization overhead.

IV. PROPOSED BENCHMARKING METHODOLOGY

To evaluate the operational feasibility of post-quantum cryptography at the smart grid edge, we designed and instrumented a hardware-in-the-loop benchmarking framework. The framework isolates the execution of cryptographic primitives on constrained dual-core silicon, subjects the system to controlled operational stress, and programmatically measures dynamic performance without introducing observer-effect latency distortion.

A. Experimental Testbed and Toolchain Integration

The specifications of primary execution platform are detailed in Table II:

TABLE II
EXPERIMENTAL SYSTEM SPECIFICATIONS

Component	Specification
Development Board	ESP32-S3-DevKitC-1-N16R8
Microcontroller	ESP32-S3 (Xtensa® 32-bit LX7 dual-core)
Clock Frequency	Fixed at 160 MHz
Internal Memory	512 KB SRAM
External PSRAM	8 MB Octal SPI (OPI)
External Flash	16 MB Quad SPI (QPI)

The software stack (detailed in Table III) is built upon Espressif IoT Development Framework (ESP-IDF), running a real-time operating system kernel based on FreeRTOS. We cross-compiled the Open Quantum Safe (liboqs v0.15.0) C library for the Xtensa target architecture using the `xtensa-esp32s3-elf-gcc` toolchain.

TABLE III
INTEGRATED SOFTWARE STACK SPECIFICATION

Component	Specification
Operating System SDK	ESP-IDF v5.5.1
PQC Library Core	liboqs v0.15.0
Toolchain	GCC 13.3.0
Classical Cryptographic Engine	mbedtls v3.6.4

B. Asymmetric Core Affinity and Evaluation Scenarios

To eliminate background operating system noise, we implement an asymmetric task isolation model. We isolate the primary post-quantum benchmarking thread on Core 1 using FreeRTOS:

```
xTaskCreatePinnedToCore(..., "PQC_Task",
65536, NULL, 5, &pqc_task_handle, 1);
```

This keeps Core 0 dedicated to background driver stacks and OS housekeeping, providing an isolated, low-jitter environment on Core 1.

We define three scheduling scenarios against which the framework’s execution jitter and isolation guarantees can be evaluated:

- **Scenario 1 (Pinned Isolation):** PQC execution is pinned to Core 1 with Core 0 idle, establishing an optimal, interference-free baseline.
- **Scenario 2 (Unpinned Dynamic Scheduling):** The thread constraint is removed, letting FreeRTOS dynamically distribute tasks across both cores to isolate scheduling and migration overheads.
- **Scenario 3 (Inter-Core Contention):** Core 1 runs the cryptographic tasks while Core 0 is saturated by a cache-evicting noise task, simulating a real-world substation multicast storm to measure micro-architectural bus and cache contention.

To demonstrate that the proposed instrumentation pipeline produces usable, hardware-derived telemetry, Table IV reports a representative sample of latency and dynamic stack high-water-mark measurements captured during an initial run under the unpinned scheduling configuration (Scenario 2). These figures illustrate the resolution and scope of the data the framework is designed to capture across the full primitive suite. A comparative cross-scenario evaluation including the pinned-isolation baseline and inter-core contention results is reserved for a forthcoming extended publication.

TABLE IV
REPRESENTATIVE TELEMETRY SAMPLE — UNPINNED (SCENARIO 2)

Operation	Mean Latency (ms)	Stack HWM (kB)
<i>Symmetric Bulk Encryption</i>		
AES-GCM-256 (50 B)	1.76	27.94
AES-GCM-256 (256 B)	1.52	27.94
AES-GCM-256 (1024 B)	1.92	27.94
PQC-Integrated AES	0.97	6.52
<i>Key Exchange & Encapsulation</i>		
X25519 KeyGen	125.76	27.95
X25519 SharedSecret	126.02	27.95
ML-KEM-768 Decaps	10.18	6.52
<i>Asymmetric Digital Signatures</i>		
ECDSA-P256 Sign	154.22	27.94
ECDSA-P256 Verify	297.79	27.94
ML-DSA-44 KeyGen	16.39	6.53
ML-DSA-44 Sign	50.13	6.52
ML-DSA-44 Verify	16.72	6.52

C. Dynamic Heap Auditing and Stack Diagnostics

Lattice-based algorithms risk stack overflows under standard task limits. We resolve this by executing ciphers inside an expanded 64 kB thread workspace, monitored via a dual-method auditing pipeline:

- 1) *Stack Auditing:* We query the stack safety margin at runtime using the FreeRTOS API `uxTaskGetStackHighWaterMark()`.
- 2) *Heap Diagnostics:* We profile the largest contiguous internal DRAM block using:

```
size_t free_block =
heap_caps_get_largest_free_block(
  MALLOC_CAP_INTERNAL | MALLOC_CAP_8BIT
);
```

This registers whether the `heap_4.c` allocator satisfies dynamic key states by utilizing scattered memory fragments across secondary SRAM banks, preserving the primary system-capped 128 kB contiguous block.

D. Programmatic Energy Integration Model

To bypass the analog drift and sampling limitations of external hardware shunts, we embed an analytical power-state integration model directly within our execution framework. We calculate the total energy-to-solution E (μJ) as:

$$E = V \cdot I_{\text{state}} \cdot \left(\frac{t_{\text{state}}}{1000} \right) \quad (9)$$

where $V = 3.3 \text{ V}$, t_{state} is the execution duration (μs), and I_{state} represents physical current profiles calibrated on our 160 MHz ESP32-S3:

- **Classical Current ($I_{\text{classical}}$):** 47.5 mA for software-only big-integer arithmetic.
- **Symmetric Current (I_{crypto}):** 31.1 mA for hardware-accelerated AES-256 operations.
- **Post-Quantum SIMD Current (I_{heavy}):** 65.5 mA during vector pipeline execution.

To resolve t_{state} with microsecond precision and avoid CPU register overflow wrap-around risks (which occur in 26.8 seconds), we query the 64-bit hardware Systimer directly via `esp_timer_get_time()` instead of standard 32-bit CPU cycle counters.

V. CONCLUSION

This paper presents a security-aware, hardware-in-the-loop benchmarking framework designed to resolve the architectural tension between memory-intensive post-quantum cryptography and real-time substation determinism. By establishing a formal testing methodology on dual-core Xtensa LX7 (ESP32-S3) silicon running FreeRTOS, we lay out an approach to address key system limitations without compromising physical grid safety. Our framework introduces an asymmetric task-isolation design to protect Core 0’s low-latency IEC 61850 GOOSE loops from PQC math on Core 1, maps the heap dynamics of the `heap_4.c` allocator to preserve contiguous system DRAM, and formulates a cycle-accurate analytical model of the Octal-SPI PSRAM bus stall penalty (C_{stall}) induced during masked post-quantum executions. Future work will focus on utilizing this framework to gather extensive empirical profiles which will be used to evaluate PQC performance at the smart grid edge.

ACKNOWLEDGMENT

This work was supported partially by the European Union in the framework of ERASMUS MUNDUS, Project CyberMACS (Project #101082683) (<https://cybermacs.eu>).

REFERENCES

- [1] National Institute of Standards and Technology (US), “Module-lattice-based key-encapsulation mechanism standard,” Tech. Rep. NIST FIPS 203, National Institute of Standards and Technology (U.S.), 2024.
- [2] National Institute of Standards and Technology (US), “Module-lattice-based digital signature standard,” Tech. Rep. NIST FIPS 204, National Institute of Standards and Technology (U.S.), 2024.
- [3] P. Duong-Ngoc and H. Lee, “Configurable Mixed-Radix Number Theoretic Transform Architecture for Lattice-Based Cryptography,” vol. 10, pp. 12732–12741, 2022.
- [4] Y. Chang, Y. Yan, C. Zhu, and Y. Liu, “A High-performance Masking Design Approach for Saber against High-order Side-channel Attack,” vol. 28, no. 6, pp. 91:1–91:19.
- [5] P. Nannipieri, S. Di Matteo, L. Zulberti, F. Albicocchi, S. Saponara, and L. Fanucci, “A RISC-V Post Quantum Cryptography Instruction Set Extension for Number Theoretic Transform to Speed-Up CRYSTALS Algorithms,” vol. 9, pp. 150798–150808, 2021.
- [6] J. Huang, H. Zhao, J. Zhang, W. Dai, L. Zhou, R. C. C. Cheung, K. Koç, and D. Chen, “Yet Another Improvement of Plantard Arithmetic for Faster Kyber on Low-End 32-bit IoT Devices,” vol. 19, pp. 3800–3813, 2024.
- [7] Y. Kim, S. Yoon, and S. C. Seo, “Vectorized Implementation of Kyber and Dilithium on 32-bit Cortex-A Series,” vol. 12, pp. 104414–104428, 2024.
- [8] S. Picek, G. Perin, L. Mariot, L. Wu, and L. Batina, “SoK: Deep Learning-based Physical Side-channel Analysis,” vol. 55, no. 11, pp. 227:1–227:35.
- [9] J.-P. D’Anvers, M. Van Beirendonck, and I. Verbauwhede, “Revisiting higher-order masked comparison for lattice-based cryptography: Algorithms and bit-sliced implementations,” *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 321–332, 2022.
- [10] O. Bronchain and G. Cassiers, “Bitslicing Arithmetic/Boolean Masking Conversions for Fun and Profit: With Application to Lattice-Based KEMs,” pp. 553–588, 2022.
- [11] N. Nagy, S. Alnemer, L. M. Alshuhail, H. Alobiad, T. Almulla, F. A. Alrumaihi, N. Ghadra, M. Nagy, N. Nagy, S. Alnemer, L. M. Alshuhail, H. Alobiad, T. Almulla, F. A. Alrumaihi, N. Ghadra, and M. Nagy, “Module-Lattice-Based Key-Encapsulation Mechanism Performance Measurements,” vol. 7, no. 3.
- [12] M. Abbasi, F. Cardoso, P. Váz, J. Silva, and P. Martins, “A Practical Performance Benchmark of Post-Quantum Cryptography Across Heterogeneous Computing Environments,” vol. 9, no. 2, p. 32, 2025.
- [13] Espressif, “ESP32-S3 Technical Reference Manual,” Tech. Rep. v1.8, Espressif, 2026.
- [14] Espressif, “ESP32-S3 Datasheet,” Tech. Rep. v2.2, Espressif, 2026.