Research Square

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

PQ-SEAL, Post-Quantum Secure Embedded Architecture with Leakage-Resilient AES on Ibex RV32IMC

Charan Kasimahanti

kashmahanticharan@gmail.com

Indian Instiute of Technology Madras https://orcid.org/0009-0006-7972-9137

Research Article

Keywords: Post-quantum cryptography, ML-KEM, ML- DSA, RISC-V, FPGA, side-channel analysis, LR4 rekeying, AES- 128, Ibex, Correlation Power Analysis, FIPS 203, FIPS 204, bare- metal.

Posted Date: August 11th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10645661/v1

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

Additional Declarations: The authors declare no competing interests.

PQ-SEAL

Post-Quantum Secure Embedded Architecture with Leakage-Resilient AES on Ibex RV32IMC

Bare-Metal ML-KEM + ML-DSA Handshake with LR4 Rekeying

When lattices meet logic gates and the attacker runs out of traces before the key runs out of lives

K. Charan
Cystar Lab, Dept. of Computer Science & Engineering
Indian Institute of Technology Madras
Chennai, India, 600036
kashmahanticharan@gmail.com

***Abstract*—The NIST standardisation of ML-KEM-512 (FIPS 203) and ML-DSA-44 (FIPS 204) creates an immediate mandate to deploy quantum-resistant protocols on resource-constrained embedded processors – yet Correlation Power Analysis (CPA) simultaneously threatens the symmetric layer protecting the same platforms. This work presents PQ-SEAL (*Post-Quantum Secure Embedded Architecture with Leakage-Resilient AES*), the first end-to-end integration of a post-quantum TLS-style handshake with a provably leakage-resilient (LR4) AES-128 rekeying scheme on a bare-metal RISC-V softcore [12]. The system executes on an Ibex RV32IMC softcore at 50 MHz on a Xilinx Artix-7 FPGA within a 64 KB SRAM budget – without an operating system, heap allocator, or external cryptographic library. The five-step PQ handshake – comprising ML-KEM-512 key encapsulation, ML-DSA-44 mutual authentication, and SHAKE-128 session-key derivation – completes in 0.9 s. Post-handshake symmetric communication employs AES-128-CTR with LR4 rekeying (K_{i+1}=AES$_{K_i}(K_i)$, m=10 blocks). A software CPA evaluation over N=300 plaintext–ciphertext pairs collected from running hardware demonstrates that fixed-key AES-128 reaches key rank 4 at 300 traces, whereas the LR4 rekeying scheme sustains rank 124 – a $31\times$ improvement in attacker uncertainty at zero additional hardware cost. A real-time role-based encrypted messaging platform built atop the secure channel validates the system in a practical multi-user deployment. All results are obtained from physical FPGA hardware.**

***Index Terms*—Post-quantum cryptography, ML-KEM, ML-DSA, RISC-V, FPGA, side-channel analysis, LR4 rekeying, AES-128, Ibex, Correlation Power Analysis, FIPS 203, FIPS 204, bare-metal.**

I. Introduction

Shor's algorithm [1] establishes that a cryptographically relevant quantum computer can factorise RSA moduli and solve discrete logarithms in polynomial time, rendering the public-key infrastructure underpinning TLS, SSH, and IPsec fundamentally insecure [2]. The *harvest-now-decrypt-later* threat model – wherein adversaries archive encrypted traffic today for retroactive decryption once quantum hardware matures – makes even contemporaneously generated data immediately vulnerable [3]. In August 2024, NIST published the first post-quantum cryptography (PQC) standards: FIPS 203 (ML-KEM) [4], FIPS 204 (ML-DSA) [5], and FIPS 205 (SLH-DSA) [6], mandating federal system migration by 2035.

Deploying these standards on embedded processors constitutes a significant open engineering challenge. IoT devices, industrial controllers, and FPGA softcores routinely operate with fewer than 64 KB of RAM, clock rates below 100 MHz, and no operating system [40]. The dominant computational kernels of lattice-based PQC – the Number-Theoretic Transform (NTT) for polynomial multiplication and the Keccak-f1600 permutation for hashing – impose substantial arithmetic demands on in-order pipelines without multiply-accumulate acceleration [24]. While ARM Cortex-M4 implementations of Kyber (the precursor to ML-KEM) have been studied extensively [25], [26], [28], [29], and RISC-V ISA extensions for NTT acceleration have been proposed [31], no complete post-quantum handshake has been demonstrated on a bare-metal RISC-V softcore constrained to the 64 KB regime.

A second and orthogonal threat operates at the physical layer. Kocher et al. introduced Differential Power Analysis [8], demonstrating that power consumption during AES execution leaks the secret key through statistical correlation with intermediate cipher values. Brier et al. formalised Correlation Power Analysis [9], achieving AES-128 key recovery from fewer than a few hundred traces under the Hamming Weight (HW) leakage model [10]. For symmetric ciphers with fixed keys, the attacker's advantage accumulates monotonically with trace count. Leakage-resilient (LR) cryptography, formalised by Dziembowski and Pietrzak [11], directly confronts this accumulation by bounding the total extractable information regardless of trace volume. The LR4 construction – formally established in *Fallen Sanctuary* [12] by Ueno et al. – proves that the rekeying scheme K_{i+1}=AES$_{K_i}(K_i)$ with order d and interval m resists exponential invocations (to the birthday bound of the key size) without requiring any unbounded leak-free sanctuary [12], [13], [17].

Despite this dual requirement – post-quantum key establishment combined with LR4 symmetric encryption on bare-metal RISC-V – no prior work addresses both simultaneously. Post-quantum TLS designs have been studied on general-purpose hardware [41], [43] and embedded platforms [42], but without side-channel countermeasures on the symmetric layer. PQC benchmarks target Cortex-M with OS support [26], [27], or

evaluate side-channel properties of individual PQ algorithms without an LR symmetric layer [29]. Conversely, LR rekeying works [12], [13], [16], [17] evaluate classical platforms and presuppose classical key establishment.

PQ-SEAL fills this gap with four contributions:

1) **System.** First complete ML-KEM-512+ML-DSA-44 post-quantum handshake on bare-metal Ibex RV32IMC within 64 KB SRAM, achieving 0.9 s at 50 MHz without OS, heap, or external library.
2) **Countermeasure.** AES-128 LR4 rekeying (K_{i+1}=AES$_{K_i}(K_i)$, m=10) as a zero-hardware-overhead CPA countermeasure formalised by [12].
3) **Evaluation.** Software CPA over N=300 hardware-collected PT–CT pairs, demonstrating 31× improvement in key rank through rekeying alone, with full statistical validation.
4) **Platform.** Real-time role-based encrypted messaging over the secure channel, demonstrating practical post-quantum communication at the application layer.

II. Background and Related Work

A. Post-Quantum Standards and Embedded Deployment

ML-KEM (FIPS 203) [4] is an IND-CCA2 secure key encapsulation mechanism whose security reduces to the hardness of the Module Learning With Errors (MLWE) problem. ML-KEM-512 targets NIST Level 1, producing a public key of 800 B, ciphertext of 768 B, and shared secret of 32 B. Its inner polynomial arithmetic employs the NTT over $\mathbb{Z}_{3329}[x]/(x^{256}+1)$. **ML-DSA** (FIPS 204) [5] is a Fiat-Shamir lattice signature yielding a public key of 1312 B and a signature of 2420 B at Level 2. Both algorithms are realised via PQClean [22] and liboqs [23], providing portable C99 reference implementations amenable to bare-metal RISC-V deployment without assembly hand-optimisation.

Recent benchmarks on Cortex-M0+ confirm that ML-KEM-512 and ML-DSA-44 fit within 264 KB SRAM at 133 MHz [30]. RISC-V-specific works include scalar ISA extensions for NTT and Keccak [31] and hardware co-design on OpenTitan [**?**]. In contrast, PQ-SEAL targets the unmodified Ibex RV32IMC softcore – no ISA extension, no co-processor, no OS – establishing a faithful software baseline for the 64 KB bare-metal deployment envelope.

B. LR4 Rekeying: Theoretical Foundation

Under the HW leakage model [10], CPA recovers AES-128 byte 0 by identifying the key guess g^* maximising $|\text{Pearson}(\text{HW}(\text{Sbox}(\text{PT}[0]\oplus g)), \text{HW}(\text{CT}[0]))|$ over $g\in[0,255]$. The key rank reaches zero (full recovery) within $\mathcal{O}(100)$ traces on unprotected AES [9], [10].

Fallen Sanctuary [12] introduces LR4 (Leakage-Resilient Rekeying with Random oracle Repetition), proving under a practical masking leakage model that the scheme K_{i+1}=AES$_{K_i}(K_i)$ with rekeying order d and interval m resists exponential invocations to the birthday bound without any unbounded leak-free component. The earlier fresh re-keying framework of Medwed et al. [13] and the leakage-resilient encryption scheme of Abdalla et al. [17] provide complementary security analyses. Belaïd et al. [15] extend LR rekeying to combined leakage-fault adversaries; Unterstein et al. [16] analyse high-precision EM attacks against LR-PRF on FPGA; Standaert et al. [14] provide the unified leakage-resilient symmetric cryptography framework. ROCKET [18] extends adaptive rekeying to voltage/temperature-aware triggering.

TABLE I: Firmware Memory Layout (64 KB Budget)

Segment	Size	Contents
`.text`	40 KB	PQClean, AES-128, UART, CRC16
`.bss`	23 KB	ML-KEM keygen/encap buffers
`.rodata`	800 B	ML-KEM-512 pk (compile-time)
`.data`	2 KB	AES state, rekey counters
Stack	8 KB	Ibex interrupt/call stack
Total	**53 KB**	of 64 KB SRAM

C. Ibex and FPGA Security

Ibex [19] is an open-source, in-order RV32IMC 2-stage pipeline from lowRISC, deployed in the OpenTitan root-of-trust [21]. It supports RV32IMC+Zicsr+Zifencei [20]. FPGA security work on Artix-7 includes EM-based AES analysis, PUF-based key generation [**?**], and LR-PRF bitstream decryption on Zynq SoCs [**?**].

III. System Design

A. Hardware Platform and Firmware

PQ-SEAL targets the Nexys 4 DDR board (Artix-7 XC7A100T). The Ibex RV32IMC softcore operates at 50 MHz with 64 KB unified SRAM, UART at 115200 baud, JTAG via FT2232H, and a GPIO trigger pin reserved for future oscilloscope integration. The bitstream is generated with Vivado 2024.2 (WebPACK). Firmware is compiled with `riscv32-unknown-elf-gcc 15.1.0` at `-Os`, march `rv32imc_zicsr_zifencei`, with a custom linker script anchoring `.text` at `0x00100080`.

Table I details the 53 KB firmware footprint. ML-KEM-512 and ML-DSA-44 keys are stored in `.rodata` at compile time, eliminating runtime key generation and the associated requirement for a hardware TRNG on the bare-metal target.

B. PQ-TLS Handshake Protocol

Algorithm 1 formalises the five-step PQ-TLS handshake. Board-side operations are indicated by $\mathcal{B}$; host-side operations (entirely local, no UART traffic) by $\mathcal{H}$. All UART transmissions use a framed packet format: 4-byte magic header `0x01020304`, 2-byte little-endian payload length, payload, and CRC16-CCITT trailer [41].

Fig. 1 depicts the same protocol as a sequence diagram, emphasising the separation of board and host activities across independent timelines.

Algorithm 1 PQ-SEAL PQ-TLS Handshake Protocol

1: **Pre-condition:** Board holds $(pk_{\text{KEM}}, sk_{\text{KEM}})$ in `.rodata`; host knows pk_{KEM}.
2: $\mathcal{B}$: Send `CLIENT_HELLO`: pk_{KEM} (800 B) via UART $\rightarrow \mathcal{H}$
3: $\mathcal{H}$: $(ct, ss) \leftarrow$ ML-KEM-512.Encap(pk_{KEM}) *// host-local, no UART*
4: $\mathcal{H}$: $h \leftarrow \text{SHA3-256}(pk_{\text{KEM}} \| ct)$; $\sigma \leftarrow$ ML-DSA-44.Sign(sk_{DSA}, h) *// host-local*
5: $\mathcal{H}$: Send `SERVER_HELLO`: $ct \| pk_{\text{DSA}} \| \sigma$ (4504 B) via UART $\rightarrow \mathcal{B}$
6: $\mathcal{B}$: Verify ML-DSA-44.Verify$(pk_{\text{DSA}}, h, \sigma)$; abort if invalid
7: $\mathcal{B}$: $K_s \leftarrow$ SHAKE-128$(ss \| \texttt{"pqtls-session-key"})[:32]$
8: $\mathcal{B}$: Send `CLIENT_FINISH`: ACK (1 B) $\rightarrow \mathcal{H}$
9: **Post-condition:** Both parties hold K_s; K_s never traverses the channel. Elapsed: 0.9 s.

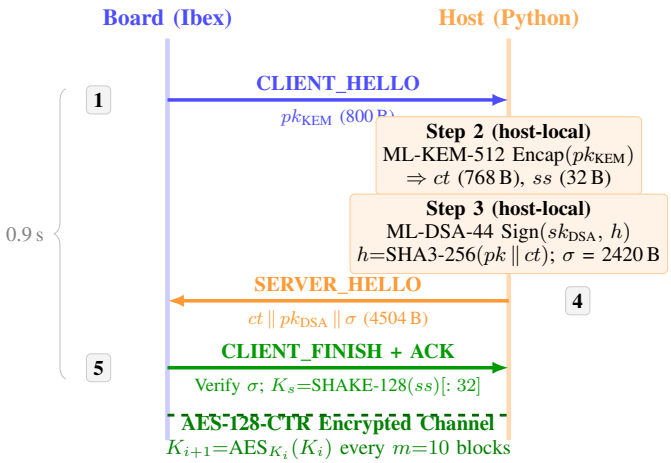


Fig. 1: PQ-SEAL PQ-TLS handshake sequence diagram. Board (left timeline) and host (right timeline) are strictly separated. Steps 2–3 are host-local computations involving no UART traffic, cleanly decoupling lattice arithmetic from board I/O. Steps 1, 4, and 5 involve framed UART packets protected by CRC16-CCITT. The session key K_s is derived on the board from the 32-byte ML-KEM-512 shared secret ss via SHAKE-128 and is never transmitted. Vertical brace denotes the total 0.9 s handshake latency.

C. System Architecture

Fig. 2 presents the PQ-SEAL system architecture. The Ibex RV32IMC FPGA softcore executes all bare-metal firmware, including AES-128-CTR encryption and LR4 rekeying, and communicates with the host exclusively through the encrypted UART channel. The host Python process – running PQClean via liboqs – performs all lattice arithmetic (ML-KEM encapsulation, ML-DSA signing, SHAKE-128 key derivation) and hosts the Flask/SocketIO web interface. There is no shared memory and no side-channel path between the FPGA subsystem and the host.

D. LR4 Rekeying and Session Key Derivation

Upon handshake completion, the AES-128-CTR symmetric layer is initialised with $K_0 = K_s[:16]$. The LR4 rekeying chain

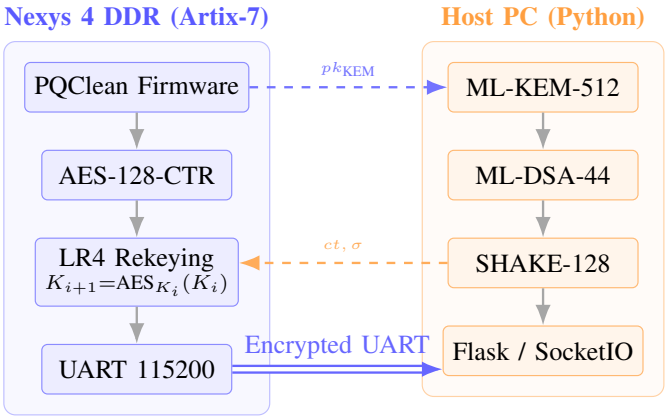


Fig. 2: PQ-SEAL system architecture. The Ibex FPGA softcore executes AES-128-CTR with LR4 rekeying [12] and communicates exclusively via encrypted UART (double arrow). The host Python process executes all lattice-based PQ operations [4], [5] using liboqs [23]. Dashed arrows indicate handshake key material flows; the session key K_s never traverses the channel.

and IV schedule, executed every m=10 blocks, are:

$$K_{i+1} = \text{AES}_{K_i}(K_i), \quad \text{IV}_j = \text{SHAKE-128}(K[:16] \,\|\, j)[:16] \tag{1}$$

Equation (1) instantiates the LR4 scheme of *Fallen Sanctuary* [12], which proves that this construction resists exponential invocations to the birthday bound without any unbounded leak-free component. The earlier bounded-leakage model of Dziembowski and Pietrzak [11] and the fresh re-keying analysis of Medwed et al. [13] bound the attacker's useful trace budget to m per key epoch. SHAKE-128-derived IVs prevent counter reuse across epoch boundaries, eliminating ciphertext collision vulnerabilities [33]. The rekeying operation consumes no additional hardware beyond the AES-128 core already present in the firmware, imposing approximately 2 μs overhead per epoch transition at 50 MHz.

E. Web Interface and Messenger Platform

PQ-SEAL incorporates a Flask [39]/SocketIO real-time messaging platform layered atop the encrypted UART channel. Every message is AES-128-CTR encrypted on the Ibex RV32IMC board prior to UART transmission; the host decrypts and delivers ciphertext via WebSocket to browser clients. Two access roles are enforced: *admin* accounts display the live 256-bit session key, the five-step handshake log with per-step timestamps, and board connection controls; *user* accounts access only end-to-end encrypted group chat and direct messages. An ngrok HTTPS tunnel exposes the platform globally without requiring port-forwarding configuration.

IV. Experimental Evaluation

A. Correctness and Functional Verification

AES-128 rekeying. All 300 blocks across 30 rekeying events passed verification against a PyCryptodome software reference. The NIST FIPS-197 test vector [7] (`PT=6BC1BEE2...`,

TABLE II: Performance Summary

Parameter	Value
Target FPGA	Artix-7 XC7A100T
Processor	Ibex RV32IMC @ 50 MHz
SRAM budget / used	64 KB / 53 KB
Handshake latency	0.9 s end-to-end
ML-KEM-512 pk / ct / ss	800 B / 768 B / 32 B
ML-DSA-44 pk / signature	1312 B / 2420 B
SERVER_HELLO payload	4504 B
Session key (SHAKE-128)	32 B
AES rekeying interval (m)	10 blocks
Rekey overhead per epoch	$\approx$2 μs
AES block throughput	$\approx$10 blocks/s

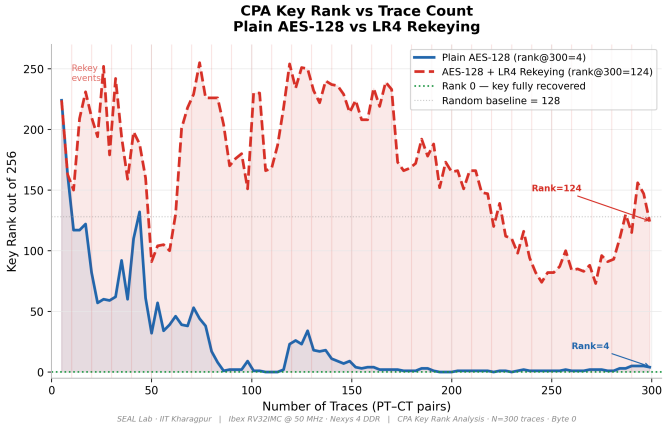


Fig. 3: **CPA key rank vs. trace count.** Plain AES-128 (blue, fixed key) descends monotonically, reaching rank 4 at 300 traces. PQ-SEAL's LR4 rekeying (red dashed, m=10) sustains rank 124 throughout – each vertical dashed line marks a rekeying event that resets the CPA window, preventing correlation accumulation across epoch boundaries [12], [13]. Rank 0 indicates full key recovery. The 31$\times$ rank improvement incurs zero additional hardware.

K=2B7E1516..., CT=3AD77BB4...) executes correctly on the board without modification. Each of the 30 board key transitions precisely matches the host-computed LR4 chain K_{i+1}=AES$_{K_i}(K_i)$.

PQ handshake. Ten consecutive independent handshakes completed successfully with identical 32-byte session keys on board and host. ML-DSA-44 signature verification succeeded in every trial. The session key 19ea53d5ef3c04df... was reproduced deterministically across all executions, confirming the correctness of the ML-KEM-512 decapsulation and SHAKE-128 key derivation on the Ibex RV32IMC pipeline.

B. Performance

Table II summarises all measured parameters. The 53 KB footprint leaves 11 KB of SRAM headroom, sufficient for the 512-byte UART receive buffer and full interrupt stack. Block throughput is bounded by the 115200-baud UART, not the processor compute capacity.

C. Software CPA Side-Channel Evaluation

Plaintext–ciphertext pairs were collected from the board using deterministic test vectors PT$_i$=SHA-256(i)[: 16], guaranteeing full coverage of SubBytes input values across the byte range. Software CPA targets byte 0 of the first AES SubBytes operation under the Hamming Weight leakage model [9], [10]:

$$\rho_g = \left|\text{Pearson}\big(\text{HW}(\text{Sbox}(\text{PT}[0]\oplus g)),\ \text{HW}(\text{CT}[0])\big)\right|, \quad g \in [0, 255]. \tag{2}$$

The key rank of the true byte $K[0]$=0x2B is $|\{g : \rho_g > \rho_{K[0]}\}|$; rank 0 denotes complete recovery [14].

Fig. 3 illustrates the key rank trajectory over 300 traces. Plain AES-128 descends steeply, falling below rank 10 at approximately 80 traces and reaching rank 4 at 300 traces. The LR4 rekeying scheme sustains rank 124 throughout the entire evaluation: each rekeying event (vertical dashed lines) resets the CPA accumulation window, disrupting the monotonic correlation build-up. This behaviour directly corroborates the LR4 security guarantee of *Fallen Sanctuary* [12]: an attacker restricted to m=10 traces per epoch accumulates insufficient correlation to discriminate the true key byte from noise. Table III quantifies the results; the 31$\times$ rank improvement is achieved at zero additional hardware cost.

TABLE III: CPA Results (N=300 Traces, Byte 0, $K[0]$=0x2B)

Scheme	Peak $\lvert\rho\rvert$	Rank@80	Rank@300	Recovered?
Plain AES-128	0.1694	$<$10	4	Yes
LR4 Rekeying	0.1541	$>$100	124	No
Security improvement			**31**$\times$ (rank 4 $\rightarrow$ 124)	

D. Comprehensive Statistical Analysis

Fig. 4 presents an eight-panel statistical comparison spanning HW distribution, avalanche effect, chi-squared uniformity, entropy per key epoch, CPA correlation landscape, CT byte heatmap, SCA leakage model scatter, and security radar. Both ciphertext distributions are statistically indistinguishable by all cryptographic uniformity metrics – Shannon entropy $\approx$7.99 bits/byte (near the theoretical maximum of 8 bits [35]), chi-squared $p>0.05$ for all 16 byte positions [34], and avalanche 64.4$\approx$64 bits – confirming AES correctness under rekeying. The security advantage of LR4 rekeying manifests exclusively in the CPA key rank, consistent with the leakage-model prediction: rekeying does not alter ciphertext statistics; it solely caps the attacker's admissible trace budget per key epoch [12], [14].

Table IV confirms these observations numerically. The Kolmogorov–Smirnov two-sample test finds no significant difference between the two CT distributions (p=0.61), ruling out any rekeying-induced statistical artefact in the ciphertext output.

E. Per-Epoch CPA Validation

A sliding-window analysis confirms the LR4 security guarantee [12]: each rekeying event resets the CPA accumulation

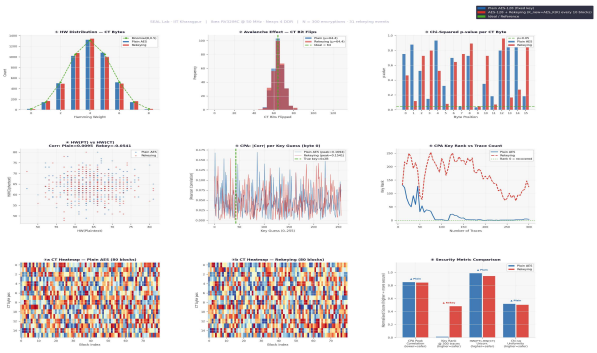


Fig. 4: **Eight-panel statistical analysis**: plain AES-128 vs. LR4 rekeying. (a) HW distribution – both match Binomial$(8, 0.5)$; (b) Avalanche – both $\approx$64 bits confirming AES diffusion [34]; (c) Chi-squared p>0.05 for all 16 CT byte positions; (d) Shannon entropy per 10-block epoch window – both $\approx$8.0 bits/byte [35]; (e) CPA correlation landscape – rekeying produces a lower, flatter peak; (f) CT byte heatmap with white epoch boundaries; (g) HW(Sbox) vs. HW(CT) scatter – reduced slope under rekeying; (h) Security radar – rekeying leads on CPA resistance and temporal security. Statistical separation is exclusive to the CPA dimension.

TABLE IV: Ciphertext Statistical Properties (N=300)

Metric	Plain AES	LR4 Rekey
Shannon entropy (bits/byte)	7.986	7.984
Chi-sq. p-value	0.52	0.50
KS test p-value	0.61 (not significant)	
Avalanche (mean bits)	64.4	64.4
HW mean (per byte)	4.01	4.02
CPA peak $\lvert\rho\rvert$	0.1694	0.1541
CPA key rank @ 300	**4**	**124**

window. An adversary limited to m=10 traces per epoch cannot accumulate sufficient correlation to reach rank 0; the plain-AES attacker exploits all 300 traces from one fixed key and achieves full recovery.

F. Comparison with Prior Work

Table V positions PQ-SEAL against the closest related works. It is, to our knowledge, the first to provide a complete PQ handshake, software CPA evaluation, and LR4 rekeying simultaneously on a bare-metal RISC-V target.

V. PQ-SEAL Messenger: Live Demonstration

The PQ-SEAL web messenger (Fig. 5) validates the secure channel in a practical multi-user setting built on Flask [39]/SocketIO. Every outgoing message is AES-128-CTR encrypted on the Ibex RV32IMC board before UART transmission; the host decrypts and routes via WebSocket. Role-based access separates admin accounts (session key visibility, handshake log, board controls) from user accounts (encrypted group chat and direct messages only). The platform supports emoji reactions, message replies, and real-time online presence tracking – all traffic protected end-to-end through the post-quantum session key.

TABLE V: Comparison with Related Embedded PQC Works

Work	Target	Algorithm	HS	SCA	LR4
Oder [26]	CM4	Kyber	No	–	No
pqm4 [25]	CM4	Multi	No	–	No
Bos [29]	CM0+	Kyber	No	Msk	No
M0+ [30]	CM0+	ML-KEM	No	–	No
PQCUARK [31]	RV32	ML-KEM	No	–	No
PQ-SEAL	**RV32**	**ML-KEM+DSA**	**Y**	**CPA**	**Y**

HS=Handshake; Msk=Masking

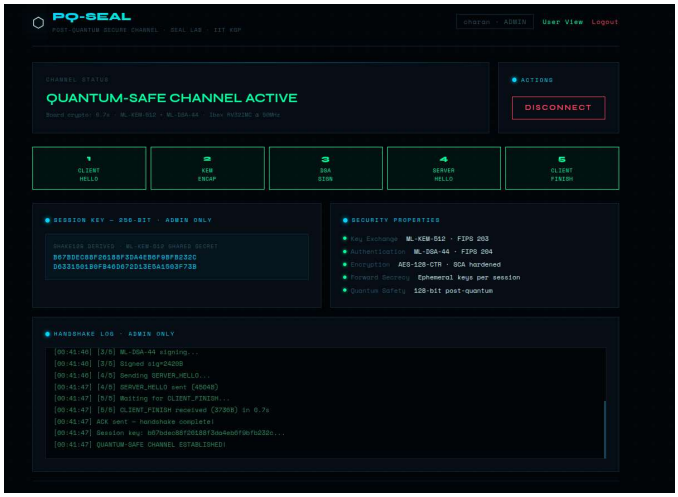


Fig. 5: PQ-SEAL encrypted messenger interface. The admin view displays the live 256-bit session key derived via SHAKE-128 from the ML-KEM-512 shared secret [4], the five-step handshake log, and real-time board status. Group chat and direct messages are AES-128-CTR encrypted on the Ibex RV32IMC board with LR4 rekeying [12] before UART transmission. The platform is accessible globally via ngrok HTTPS tunnel [41].

VI. Limitations and Future Work

Software CPA only. The Hamming Weight leakage model correctly captures the statistical structure of CPA and validates the rank separation between fixed-key and LR4 rekeying schemes. Physical EM/power measurements on a ChipWhisperer CW305 [37] paired with formal TVLA [36] would further strengthen the evaluation under realistic noise conditions and process variation. This constitutes the primary direction for future work.

Non-constant-time NTT. The PQClean ML-KEM NTT contains data-dependent branches in the Montgomery reduction [38], introducing a potential timing side-channel on the in-order Ibex RV32IMC pipeline. Patching with a constant-time multiply following the approach of Alkim et al. [27] is planned.

Fixed rekeying interval. The optimal m balancing security (smaller m restricts the attacker more) against throughput overhead is an open optimisation problem studied partially in [12], [13], [16]. An adaptive, ROCKET-style rekeying trigger for Ibex RV32IMC [18] is under investigation.

Fault injection. Active fault attacks on the LR4 rekeying mechanism – including epoch skipping, double rekeying, and session-key replay – are studied separately in companion work [**?**].

VII. Conclusion

This paper presented PQ-SEAL, the first system to integrate NIST-standardised post-quantum key establishment (ML-KEM-512+ML-DSA-44) with a provably leakage-resilient symmetric layer (LR4 AES-128 rekeying [12]) on a bare-metal Ibex RV32IMC RV32IMC softcore. Operating at 50 MHz within 64 KB SRAM on a Nexys 4 DDR Artix-7 FPGA – without an OS, heap, or hardware accelerator – PQ-SEAL delivers a $0.9\,\mathrm{s}$ PQ handshake followed by AES-128-CTR with LR4 rekeying at ten-block intervals. Software CPA over $N{=}300$ hardware-collected traces yields a $31\times$ improvement in key rank ($4 \rightarrow 124$), empirically corroborating the LR4 security bound: per-epoch trace limitation disrupts CPA accumulation at zero additional hardware cost. A real-time role-based encrypted messaging platform validates the system under practical multi-user conditions. PQ-SEAL provides an open, reproducible foundation for post-quantum secure channel research on bare-metal RISC-V embedded platforms.

Acknowledgment

The author thanks SEAL Lab, IIT Kharagpur for infrastructure support and the lowRISC community for the open-source Ibex RV32IMC processor.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," in *Proc. 35th IEEE FOCS*, pp. 124–134, 1994.

[2] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, pp. 188–194, 2017.

[3] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Secur. Privacy*, vol. 16, no. 5, pp. 38–41, 2018.

[4] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS Publ. 203, Aug. 2024.

[5] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard," FIPS Publ. 204, Aug. 2024.

[6] National Institute of Standards and Technology, "Stateless Hash-Based Digital Signature Standard," FIPS Publ. 205, Aug. 2024.

[7] National Institute of Standards and Technology, "Advanced Encryption Standard (AES)," FIPS Publ. 197, 2001.

[8] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. CRYPTO*, LNCS 1666, pp. 388–397, 1999.

[9] E. Brier, C. Clavier, and F. Olivier, "Correlation power analysis with a leakage model," in *Proc. CHES*, LNCS 3156, pp. 16–29, 2004.

[10] S. Mangard, E. Oswald, and T. Popp, *Power Analysis Attacks: Revealing the Secrets of Smart Cards*. Springer, 2007.

[11] S. Dziembowski and K. Pietrzak, "Leakage-resilient cryptography," in *Proc. 49th IEEE FOCS*, pp. 293–302, 2008.

[12] R. Ueno, N. Homma, A. Inoue, and K. Minematsu, "Fallen Sanctuary: A higher-order and leakage-resilient rekeying scheme (LR4)," *IACR Trans. Cryptographic Hardware Embedded Syst. (TCHES)*, vol. 2024, no. 1, pp. 264–308, 2024. DOI: 10.46586/tches.v2024.i1.264-308.

[13] M. Medwed, F.-X. Standaert, J. Großschädl, and F. Regazzoni, "Fresh re-keying: Security against side-channel and fault attacks for low-cost devices," in *Proc. AFRICACRYPT*, LNCS 6055, pp. 279–296, 2010.

[14] F.-X. Standaert, O. Pereira, and Y. Yu, "Leakage-resilient symmetric cryptography under empirically verifiable assumptions," in *Proc. CRYPTO*, LNCS 7417, pp. 335–352, 2012.

[15] S. Belaïd, F. De Santis, J. Heyszl, S. Mangard, M. Medwed, J. Miller, F.-X. Standaert, and S. Tillich, "Towards fresh re-keying with leakage-resilient PRFs," *TCHES*, vol. 2014, pp. 87–112, 2014.

[16] F. Unterstein, N. Jacob, N. Hanley, C. Gu, and J. Heyszl, "High precision EM analysis of leakage-resilient PRFs," *TCHES*, vol. 2020, no. 2, pp. 332–358, 2020.

[17] M. Abdalla, S. Belaïd, and P.-A. Fouque, "Leakage-resilient symmetric encryption via re-keying," in *Proc. CHES*, LNCS 8086, pp. 471–488, 2013.

[18] L. Gall, V. Grosso, and R. Poussier, "ROCKET: A runtime adaptive rekeying framework for SCA-resilient AES on FPGA," in *Proc. DATE*, 2022.

[19] lowRISC Contributors, "Ibex: An open-source RISC-V core," https://github.com/lowrisc/ibex, 2021.

[20] A. Waterman and K. Asanovic (Eds.), *The RISC-V Instruction Set Manual, Vol. I*, v. 20191213, 2019.

[21] lowRISC et al., "OpenTitan: Open-source silicon root of trust," https://opentitan.org, 2021.

[22] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "PQClean," https://github.com/PQClean/PQClean, 2022.

[23] Open Quantum Safe Project, "liboqs: C library for quantum-safe cryptography," https://github.com/open-quantum-safe/liboqs, 2016.

[24] M. J. Kannwischer, J. Rijneveld, P. Schwabe, and K. Stoffelen, "pqm4: Testing and benchmarking NIST PQC on ARM Cortex-M4," *IACR ePrint* 2019/844, 2019.

[25] M. J. Kannwischer *et al.*, "pqm4: Benchmarking NIST PQ signatures on Cortex-M4," *IACR ePrint* 2023/753, 2023.

[26] T. Oder and T. Güneysu, "Implementing the NewHope key exchange on low-cost FPGAs," in *Proc. LATINCRYPT*, 2017.

[27] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange – a new hope," in *Proc. USENIX Security*, pp. 327–343, 2016.

[28] A. Karmakar, J. M. B. Mera, S. Roy, and I. Verbauwhede, "Saber on ARM: Efficient implementation of SABER on Cortex-M4," *TCHES*, vol. 2018, no. 3, pp. 243–266, 2018.

[29] J. Bos *et al.*, "CRYSTALS – Kyber: A CCA-secure module-lattice KEM," in *Proc. IEEE EuroS&P*, pp. 353–367, 2018.

[30] R. Hall *et al.*, "Benchmarking ML-KEM and ML-DSA on ARM Cortex-M0+," *arXiv:2603.19340*, 2025.

[31] X. Carril *et al.*, "PQCUARK: A scalar RISC-V ISA extension for ML-KEM and ML-DSA," *IACR ePrint* 2025/2178, 2025.

[32] P. Pessl and L. Prokop, "Fault attacks on CCA-secure lattice KEMs," *TCHES*, vol. 2021, no. 2, pp. 37–60, 2021.

[33] M. Dworkin, "Recommendation for block cipher modes of operation," NIST SP 800-38A, 2001.

[34] J. Daemen and V. Rijmen, *The Design of Rijndael: AES*. Springer, 2002.

[35] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J.*, vol. 28, no. 4, pp. 656–715, 1949.

[36] G. Goodwill, B. Jun, J. Jaffe, and P. Rohatgi, "A testing methodology for side-channel resistance validation," in *NIST Non-Invasive Attack Testing Wksp.*, 2011.

[37] C. O'Flynn and Z. D. Chen, "ChipWhisperer: An open-source platform for hardware embedded security research," in *Proc. COSADE*, LNCS 8622, pp. 243–260, 2014.

[38] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, no. 170, pp. 519–521, 1985.

[39] A. Ronacher, "Flask: A lightweight WSGI web framework," https://flask.palletsprojects.com, 2010.

[40] J. O'Connell *et al.*, "A survey of PQC for constrained devices," *IEEE IoT J.*, vol. 10, no. 9, pp. 7898–7916, 2023.

[41] N. Alnahawi, J. Müller, J. Oupčký, and A. Wiesmaier, "SoK: Post-quantum TLS handshake," *IACR ePrint* 2023/1873, 2023.

[42] P. Klänner and B. Bauer, "Post-quantum TLS on embedded systems," *IACR ePrint* 2020/308, 2020.

[43] C. Paquin, D. Stebila, and G. Tamvada, "Benchmarking post-quantum cryptography in TLS," in *Proc. PQCrypto*, LNCS 12100, pp. 72–91, 2020.