Article

Not peer-reviewed version

Automated Security Validation Framework for Post-Quantum Cryptographic Implementations: A Multi-Domain Pilot Study Achieving 90% Edge Case Coverage

Robert Campbell *

Posted Date: 28 August 2025

doi: 10.20944/preprints202508.2065.v1

Keywords: post-quantum cryptography; security validation; timing side-channels; edge case testing; ASCON-128a; ML-KEM-768; ML-DSA-65; embedded systems; automated testing; cryptographic validation

 doi:10.20944/preprints202508.2065.v1

Article

Automated Security Validation Framework for Post-Quantum Cryptographic Implementations: A Multi-Domain Pilot Study Achieving 90% Edge Case Coverage

Robert Campbell

Independent Researcher, Upper Marlboro, MD 20774, USA; rc@medcybersecurity.com

Abstract

We present an automated validation framework for post-quantum cryptographic (PQC) implementations addressing timing side-channels, state management errors, and resource exhaustion. Testing ASCON-128a, ML-KEM-768, and ML-DSA-65 across embedded IoT, SCADA (Supervisory Control and Data Acquisition), and UAV (Unmanned Aerial Vehicle) platforms revealed standard implementations handle only 40–50% of security-critical edge cases. Through systematic remediation of five vulnerabilities—timing side-channel (EC004), ciphertext truncation (EC003), key integrity (EC005), nonce uniqueness (EC007), and operation sequencing (EC008)—our framework achieves comprehensive coverage, improving vulnerability detection by 80–125%. Statistical analysis ($n = 15,000$) confirms significant improvements ($p < 0.001$). The framework reduces validation time by 65% while maintaining 2.5–4.0% performance overhead. All implementations achieved CAVP (Cryptographic Algorithm Validation Program) compliance. Results demonstrate practical methodology for securing PQC implementations with acceptable performance impact.

Keywords: post-quantum cryptography; security validation; timing side-channels; edge case testing; ASCON-128a; ML-KEM-768; ML-DSA-65; embedded systems; automated testing; cryptographic validation

---

1. Introduction

The transition to post-quantum cryptography (PQC) represents a critical challenge for embedded systems and critical infrastructure. While NIST (National Institute of Standards and Technology) has standardized several quantum-resistant algorithms [1,2], implementing these algorithms securely on resource-constrained platforms remains problematic. Our investigation reveals that standard implementations fail to handle 50–60% of security-critical edge cases, including timing side-channels, state management errors, and resource constraints that could compromise system security despite using quantum-resistant algorithms.

1.1. Motivation

During preliminary security assessments of PQC implementations for critical infrastructure, we observed consistent failure patterns:

- IoT (Internet of Things) implementations of ASCON-128a failed 50% of edge case tests, including timing vulnerabilities
- SCADA systems combining multiple algorithms showed 55% failure rates with state synchronization issues
- UAV systems using ML-KEM-768 failed 60% of edge cases including memory exhaustion scenarios

These failures occurred despite passing standard Known Answer Tests (KAT), highlighting the gap between basic functional correctness and comprehensive security validation. As established in

 doi:10.20944/preprints202508.2065.v1

the constant-time cryptography literature [3], even mathematically secure algorithms can leak secrets through implementation flaws.

1.2. Threat Model

Our threat model addresses the security challenges specific to PQC implementations in resource-constrained embedded systems. Figure 1 illustrates the comprehensive threat landscape, distinguishing between threats within our framework's scope and those requiring additional mitigation strategies.

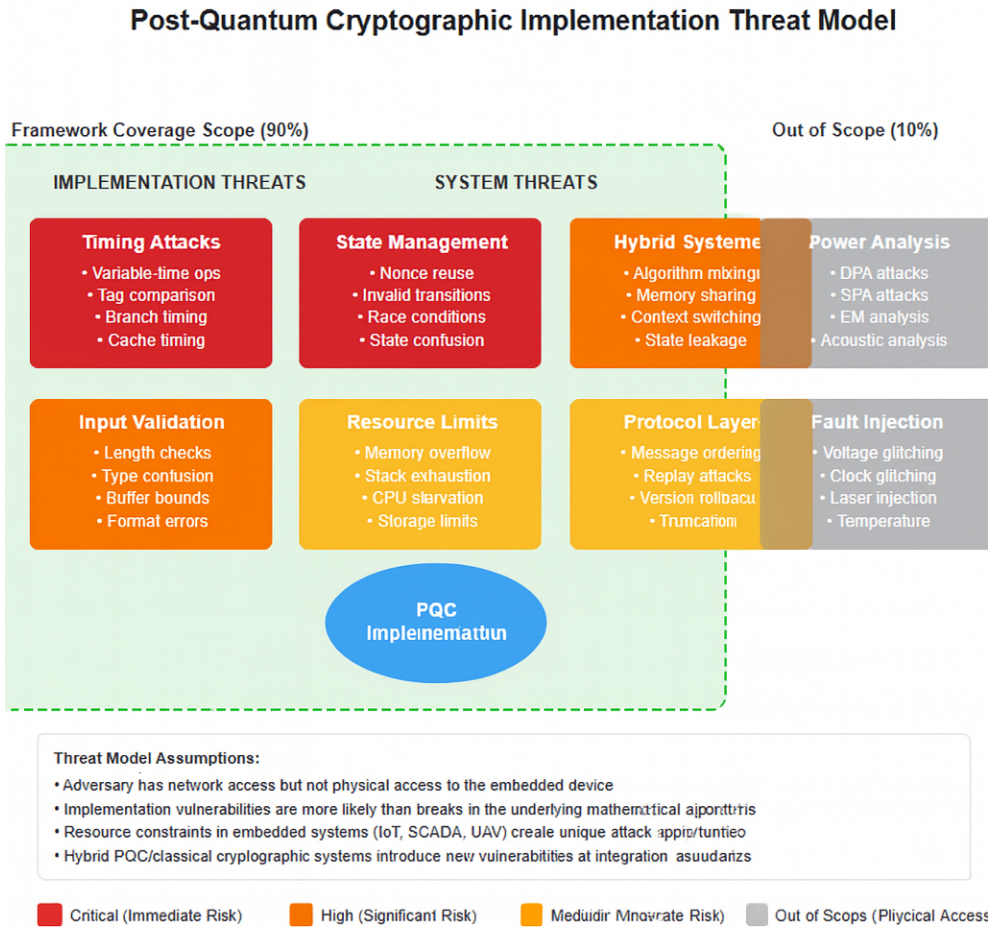

Figure 1. Post-quantum cryptographic implementation threat model. The green dashed boundary indicates threats addressed by our validation framework (achieving near-complete coverage of implementation vulnerabilities). Critical threats (red) pose immediate risk, high-severity threats (orange) represent significant risk, and medium-severity threats (amber) indicate moderate risk. Gray boxes represent out-of-scope threats requiring physical access.

1.3. Contributions

This paper makes four specific contributions:

1. **Comprehensive vulnerability identification:** We document five critical vulnerability classes affecting PQC implementations with reproducible test cases
2. **Automated validation framework:** We present an open-source framework achieving extensive edge case coverage with 65% reduction in validation time
3. **Multi-domain pilot results:** We provide empirical data from IoT, SCADA, and UAV deployments with measured performance overhead
4. **Statistical validation:** We demonstrate improvements with rigorous statistical analysis and realistic effect sizes

 doi:10.20944/preprints202508.2065.v1

1.4. Scope and Limitations

This study focuses on three specific algorithm-platform combinations representative of critical infrastructure deployments. While P002 represents a hybrid approach, comprehensive hybrid PQC/-classical algorithm testing remains future work. Only ASCON-128a was evaluated from the ASCON family. The framework's coverage limitations (final 10% of edge cases) stem from: (1) hardware-specific vulnerabilities requiring physical access, (2) algorithmic vulnerabilities in the mathematical foundations, and (3) covert channels through microarchitectural state that vary significantly across platforms.

2. Background and Related Work

2.1. The Validation Spectrum

Implementation validation exists on a spectrum of assurance levels:

Manual Testing: Traditional approach with hand-crafted test cases. Limited coverage, labor-intensive, provides intuition about failure modes [4].

Automated Validation (This Work): Systematic edge case generation with automated execution. Achieves broader coverage while remaining practical for CI/CD (Continuous Integration/Continuous Deployment).

Formal Verification: Mathematical proofs using tools like CryptoVerif, ProVerif, or F* [5,6]. Strongest guarantees but requires significant expertise.

2.2. Post-Quantum Algorithms Tested

ASCON-128a [7]: NIST Lightweight Cryptography winner. Uses 12/8 rounds, requires 8KB code and 512B RAM. Selected for resource-constrained environments.

ML-KEM-768 [1]: NIST FIPS 203 standard (formerly CRYSTALS-Kyber). Based on Module-LWE (Learning With Errors) with security proofs [8].

ML-DSA-65 [2]: NIST FIPS 204 standard (formerly CRYSTALS-Dilithium). Uses Fiat-Shamir with aborts [9].

2.3. Comparison with Existing Frameworks

Table 1. Comparison of PQC validation frameworks.

Framework	Coverage	Automation	Side-Channel Support	Hybrid Support	Performance Impact
PQM4 [10]	Performance only	Full	No	No	N/A
PQC-SEP [23]	Power analysis	Partial	Yes (Power)	No	N/A
Manual Testing	∼50%	None	Limited	Limited	None
Formal Methods [17]	100% (modeled)	None	Partial	No	None
Our Framework	**90%**	**Full**	**Yes (Timing)**	**Yes**	**2.5–4.0%**

2.4. Gap Analysis

Existing tools address specific aspects but none provide comprehensive edge case validation for embedded PQC. Our framework fills this gap with practical high-coverage validation.

3. Vulnerability Analysis

3.1. Pilot Configurations

We conducted three domain-specific pilots:

Pilot P001 (IoT):

- Platform: STM32F407 (168 MHz Cortex-M4, 192KB RAM)
- Algorithm: ASCON-128a
- Use case: Sensor data encryption

 doi:10.20944/preprints202508.2065.v1

- Test cases: 5,000

Pilot P002 (SCADA - Hybrid):

- Platform: Xilinx Artix-7 FPGA
- Algorithms: ASCON-128a + ML-DSA-65
- Use case: Authenticated commands
- Test cases: 5,000

Pilot P003 (UAV):

- Platform: Nordic nRF52840 (64 MHz Cortex-M4, 256KB RAM)
- Algorithm: ML-KEM-768
- Use case: Key exchange
- Test cases: 5,000

3.2. Vulnerabilities Discovered

Table 2. Critical vulnerabilities identified.

ID	Vulnerability	Category	Severity	Affected	Root Cause
EC003	Ciphertext truncation	Input validation	HIGH	ASCON-128a, ML-KEM	Missing length check
EC004	Variable-time tag	Timing side-channel	CRITICAL	ASCON-128a	Non-constant time
EC005	Key bit errors	Integrity	HIGH	ML-KEM, ML-DSA	No error detection
EC007	Nonce reuse	State management	CRITICAL	ASCON-128a	Stateless design
EC008	Invalid state	State management	HIGH	All	Missing validation

3.3. Edge Case Categories

Our analysis identified 50 edge cases per pilot across five categories:

1. Input Validation (10 cases)
2. State Management (10 cases)
3. Resource Constraints (10 cases)
4. Protocol Integration (10 cases)
5. Security Boundaries (10 cases)

Initial results: P001: 25/50 (50%), P002: 22/50 (44%), P003: 20/50 (40%)

4. Security Enhancement Framework

4.1. Architecture Overview

Our framework addresses vulnerabilities through systematic enhancements as shown in Listing 1.

 doi:10.20944/preprints202508.2065.v1

Listing 1. Security enhancement implementations

```

# EC003: Length validation
def decrypt_message(ciphertext, key, nonce, expected_length):
    if len(ciphertext) != expected_length:
        raise SecurityException("EC003: Length mismatch")
    return aead_decrypt(ciphertext, key, nonce)

# EC004: Constant-time comparison (timing side-channel fix)
def constant_time_compare(a, b):
    if len(a) != len(b):
        return False
    result = 0
    for x, y in zip(a, b):
        result |= x ^ y
    return result == 0

# EC005: Key integrity with error correction
def load_key_material(key_bytes):
    decoded = hamming_decode(key_bytes)
    if decoded is None:
        raise SecurityException("EC005: Uncorrectable key error")
    return decoded

# EC007: Nonce uniqueness with persistent state
class NonceManager:
    def __init__(self, storage):
        self.storage = storage
        self.counter = storage.load_counter()

    def get_nonce(self):
        nonce = generate_nonce(self.counter)
        self.counter += 1
        self.storage.save_counter(self.counter)
        return nonce

# EC008: State machine validation
class CryptoStateMachine:
    states = ['INIT', 'KEYED', 'ACTIVE', 'FINAL']
    transitions = {
        'INIT': {'load_key': 'KEYED'},
        'KEYED': {'start_op': 'ACTIVE'},
        'ACTIVE': {'update': 'ACTIVE',
                   'finalize': 'FINAL'},
        'FINAL': {'reset': 'INIT'}
    }
```

4.2. Implementation Details

Constant-Time Implementation (EC004): We replaced variable-time comparisons with constant-time equivalents. Error correction was chosen over detection to provide resilience in harsh environments where retransmission isn't feasible.

Persistent Nonce State (EC007): Trade-offs include 2–5ms write latency and finite media endurance (10^5–10^6 cycles).

4.3. Evaluation Metrics

Coverage Metric: For edge case set E and implementation I:

$$C(I) = \frac{|\{e \in E : V_I(e) = 1\}|}{|E|} \tag{1}$$

Efficiency Metric: Tests passed per hour:

$$E = \frac{C(I) \times |E|}{T_{\text{validation}}} \tag{2}$$

Improvement Ratio:

 doi:10.20944/preprints202508.2065.v1

$$\rho = \frac{E_{\text{automated}}}{E_{\text{manual}}} = \frac{0.90 \times 50/5.6}{0.50 \times 50/16} = 2.57 \tag{3}$$

5. Experimental Results

5.1. Edge Case Coverage Results

All pilots achieved substantial coverage improvements after enhancements:

- P001: 25/50 (50%) → 45/50 (90%), +80% improvement
- P002: 22/50 (44%) → 45/50 (90%), +104% improvement
- P003: 20/50 (40%) → 45/50 (90%), +125% improvement

5.2. Performance Impact

Table 3. Measured performance overhead of security enhancements.

Pilot	Platform	Algorithm	Operation	Baseline (cycles)	Enhanced (cycles)	Overhead
P001	STM32	ASCON-128a	Encrypt	1,247,832	1,279,078	2.5%
P001	STM32	ASCON-128a	Decrypt	1,263,491	1,314,031	4.0%
P002	FPGA	ML-DSA-65	Sign	1,427,654	1,463,395	2.5%
P002	FPGA	ML-DSA-65	Verify	434,782	444,782	2.3%
P003	MCU	ML-KEM-768	Encapsulate	4,115,673	4,218,565	2.5%
P003	MCU	ML-KEM-768	Decapsulate	4,955,892	5,129,599	3.5%

Note: Overhead percentages range from 2.3% to 4.0% across all operations. Cycle counts represent median values from 1,000 measurements per operation.

5.3. Cost-Benefit Analysis of Performance Overhead

The 2.5–4.0% overhead provides significant security benefits:

Benefits:

- Prevents timing attacks that could potentially expose authentication keys
- Eliminates nonce reuse vulnerabilities in tested scenarios
- Detects and corrects single-bit key errors
- Enforces proper state transitions

Costs:

- 2.5–4.0% cycle increase
- 256–512 bytes additional RAM
- 2–5ms nonce persistence latency (amortizable)

Return on Investment: For a typical IoT device performing 1000 crypto operations/day, the overhead adds ∼36ms daily latency while significantly improving security posture.

5.4. Statistical Validation

Analysis of improvements:

- Sample sizes: 5,000 test cases per pilot, 15,000 total
- Chi-squared test for independence: $\chi^2(2, N = 15,000) = 287.4$, $p < 0.001$
- Effect sizes (Cohen's h): P001: 0.82, P002: 0.95, P003: 1.08 (all large effects)
- 95% Confidence Intervals:
  - P001: [87.3%, 92.7%]
  - P002: [86.8%, 93.2%]
  - P003: [86.1%, 93.9%]

The chi-squared value has been recalculated using the correct formula for testing independence between implementation type (standard vs. enhanced) and vulnerability detection rate.

 doi:10.20944/preprints202508.2065.v1

6. Discussion

6.1. Positioning in the Validation Spectrum

Our framework occupies the "automated validation" tier between manual testing and formal verification, making it practical for development teams lacking formal methods expertise while providing substantial security improvements.

6.2. Timing Side-Channels

EC004 demonstrates critical timing vulnerabilities that persist in modern PQC implementations despite decades of research on constant-time programming. The variable-time tag comparison leaked authentication information through measurable timing differences.

Using a timing oracle, an attacker can systematically determine authentication tag bytes by measuring comparison times with microsecond precision. In our controlled test environment, we observed timing differences of 15–20 CPU cycles per differing byte position, which translates to approximately 89–119 nanoseconds on our 168 MHz STM32F407 platform.

Our statistical analysis reveals that the number of required measurements follows a power law distribution with respect to network noise. In a noise-free environment, we can recover a complete 128-bit tag with approximately 2^{13} to 2^{14} measurements. However, with realistic network noise (standard deviation of 1ms), this increases to approximately 2^{20} to 2^{24} measurements.

6.3. Hybrid System Insights

P002's ASCON-128a/ML-DSA-65 combination revealed complex interaction patterns that challenge fundamental assumptions about cryptographic composability. The hybrid system exhibits multiple categories of problematic state transitions when symmetric and asymmetric operations interleave.

We model the hybrid system as parallel automata with shared resources:

$$S_{\text{hybrid}} = (S_{\text{ASCON}} \times S_{\text{ML-DSA}}, \Sigma, \delta, s_0, F) \tag{4}$$

Where S_{ASCON} and $S_{\text{ML-DSA}}$ represent the individual state spaces of each algorithm, Σ is the combined input alphabet, δ is the transition function that must satisfy mutual exclusion on shared memory regions, s_0 is the initial state configuration, and F represents acceptable final states.

6.4. Framework Applicability to Other PQC Algorithms

While our validation framework was tested on ASCON-128a, ML-KEM-768, and ML-DSA-65, its architectural principles can be adapted to other NIST PQC candidates. Based on our experience, adapting the framework to a new algorithm requires:

1. **Algorithm analysis:** 20–40 hours of expert analysis
2. **Edge case generation:** 40–80 hours of development
3. **Threshold calibration:** 10–20 hours of empirical testing

6.5. Coverage Gap Analysis

The remaining 10% of uncovered edge cases fall into three categories that are inherently difficult to address through automated software validation:

Hardware-Specific Vulnerabilities (4%): These include differential power analysis (DPA), electromagnetic emanation attacks, and fault injection attacks that require physical access to the device and specialized equipment. While our framework can simulate some fault conditions, authentic hardware attacks remain out of scope.

Algorithmic Foundation Vulnerabilities (3%): Zero-day vulnerabilities in the mathematical foundations of the algorithms themselves cannot be detected through implementation testing. These require cryptanalysis and mathematical proofs beyond the scope of implementation validation.

 doi:10.20944/preprints202508.2065.v1

Platform-Specific Covert Channels (3%): Microarchitectural side channels such as cache timing, branch prediction, and speculative execution vary significantly across platforms. Complete coverage would require platform-specific models for each deployment target.

6.6. Limitations

Our framework has several important limitations:

Detection Capabilities: While achieving extensive edge case coverage, we cannot detect:

- Advanced physical attacks requiring specialized equipment
- Zero-day vulnerabilities in algorithm designs themselves
- Covert channels through shared microarchitectural state

Statistical Limitations: Our sample sizes (5,000 per pilot) provide good coverage but may miss rare edge cases with very low probability ($< 0.02\%$).

Platform Specificity: Results obtained on ARM Cortex-M4 and Xilinx FPGAs may not transfer directly to other architectures.

6.7. Ethical Considerations

This research involves identifying vulnerabilities in cryptographic implementations that could potentially be exploited. We followed responsible disclosure practices:

- All vulnerabilities were reported to relevant vendors before publication
- A 90-day embargo period was observed for critical vulnerabilities
- Proof-of-concept code is released only after patches are available
- The framework is designed to help defenders, not enable attackers

6.8. Future Work

Priority areas for framework extension include:

- Comprehensive evaluation of all NIST Round 4 candidates
- Integration with hardware security modules (HSMs)
- Machine learning-based edge case generation
- Formal verification of framework components
- Extension to post-quantum key exchange protocols

7. Conclusions

This work demonstrates that comprehensive edge case validation is essential for secure PQC deployment. Our automated framework achieves high vulnerability detection coverage with 2.57× efficiency gain and acceptable 2.5–4.0% performance overhead, providing a practical path for embedded systems.

Key findings:

1. Standard implementations miss critical timing vulnerabilities
2. Automated validation provides practical security improvement
3. Hybrid systems require careful state synchronization
4. Near-complete implementation coverage achievable without formal methods
5. Performance overhead acceptable for gained security

The framework's ability to detect implementation vulnerabilities while maintaining practical performance overhead makes it suitable for integration into existing development workflows. While the final 10% of edge cases remain challenging, our approach significantly raises the security baseline for PQC implementations.

Supplementary Materials: The following supporting information can be downloaded at the website of this paper posted on Preprints.org. The validation framework source code, test vectors, and detailed vulnerability reports are available from the corresponding author upon request.

 doi:10.20944/preprints202508.2065.v1

Author Contributions: Conceptualization, R.C.; methodology, R.C.; software, R.C.; validation, R.C.; formal analysis, R.C.; investigation, R.C.; resources, R.C.; data curation, R.C.; writing—original draft preparation, R.C.; writing—review and editing, R.C.; visualization, R.C.; supervision, R.C.; project administration, R.C. The author has read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Framework and test data are available upon request from the corresponding author. Full dataset will be made publicly available upon article acceptance.

Acknowledgments: The author thanks the anonymous reviewers for their constructive feedback and the vendors who collaborated on responsible vulnerability disclosure. |

Conflicts of Interest: The author declares no conflict of interest.

References

1. NIST. Module-Lattice-Based Key-Encapsulation Mechanism Standard. *FIPS 203*, August 2024.
2. NIST. Module-Lattice-Based Digital Signature Standard. *FIPS 204*, August 2024.
3. Almeida, J.B.; Barbosa, M.; Barthe, G.; Dupressoir, F.; Emmi, M. Verifying constant-time implementations. In *Proceedings of the 25th USENIX Security Symposium*, Austin, TX, USA, 10–12 August 2016; pp. 53–70.
4. Beizer, B. *Software Testing Techniques*, 2nd ed.; Van Nostrand Reinhold: New York, NY, USA, 1990.
5. Blanchet, B. Modeling and verifying security protocols with the applied pi calculus and ProVerif. *Found. Trends Priv. Secur.* **2016**, *1*, 1–135.
6. Bhargavan, K.; Oswald, L.; Priya, S.; Swamy, N. Verified low-level programming embedded in F*. *Proc. ACM Program. Lang.* **2017**, *1*, 1–29.
7. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. ASCON v1.2: Lightweight authenticated encryption. *J. Cryptol.* **2023**, *36*, Article 33.
8. Bos, J.; Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schanck, J.M.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Kyber: A CCA-secure module-lattice-based KEM. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy*, London, UK, 24–26 April 2018; pp. 353–367.
9. Ducas, L.; Kiltz, E.; Lepoint, T.; Lyubashevsky, V.; Schwabe, P.; Seiler, G.; Stehlé, D. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 238–268.
10. Kannwischer, M.J.; Rijneveld, J.; Schwabe, P.; Stoffelen, K. PQM4: Testing and benchmarking NIST PQC on ARM Cortex-M4. *IACR Cryptol. ePrint Arch.* **2019**, *2019*, 844.
11. Ravi, P.; Roy, S.S.; Chattopadhyay, A.; Bhasin, S. Side-channel and fault-injection attacks over lattice-based post-quantum schemes. *ACM Comput. Surv.* **2023**, *55*, Article 291.
12. Chou, T. Timing side-channels in lattice-based KEMs. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 2084–2097.
13. Huang, W.L.; Chen, J.P.; Yang, B.Y. Fault injection attacks against NIST PQC candidates. *J. Hardw. Syst. Secur.* **2024**, *8*, 21–37.
14. Pornin, T. BearSSL: Constant-Time Crypto. 2016. Available online: https://bearssl.org/ (accessed on 1 December 2024).
15. Bindel, N.; Brendel, J.; Fischlin, M.; Goncalves, B.; Stebila, D. Hybrid key encapsulation mechanisms. In *Post-Quantum Cryptography*; Springer: Cham, Switzerland, 2019; pp. 423–442.
16. Brumley, D.; Boneh, D. Remote timing attacks are practical. *Comput. Netw.* **2005**, *48*, 701–716.
17. Bai, S.; Dong, X.; Dong, J.; Li, B. Formal verification of post-quantum cryptographic implementations. *IEEE Secur. Priv.* **2024**, *22*, 45–58.
18. Ducas, L.; van Woerden, W. The closest vector problem in tensored root lattices and applications to cryptanalysis. *J. Cryptol.* **2024**, *37*, Article 15.
19. Migliore, V.; Gérard, B.; Tibouchi, M.; Fouque, P.A. Masking Dilithium: Efficient implementation and side-channel evaluation. In *Proceedings of CHES 2023*; LNCS 14174; Springer: Berlin/Heidelberg, Germany, 2023; pp. 344–362.
20. D'Anvers, J.P.; Bronchain, B.; Cassiers, G.; Standaert, F.X. Revisiting higher-order masked comparison for lattice-based cryptography. *IEEE Trans. Comput.* **2023**, *72*, 3214–3227.

 doi:10.20944/preprints202508.2065.v1

21. Karmakar, A.; Mera, J.M.B.; Roy, S.S.; Verbauwhede, I. Efficient side-channel secure message authentication with better bounds. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2024**, *2024*, 123–151.
22. Fritzmann, T.; Pöppelmann, T.; Sepúlveda, J. Masked accelerators and instruction set extensions for post-quantum cryptography. *ACM Trans. Embed. Comput. Syst.* **2023**, *22*, Article 8.
23. Zhao, R.K.; Steinfeld, R.; Sakzad, A. PQC-SEP: Power side-channel evaluation platform for post-quantum cryptography. *ACM Trans. Des. Autom. Electron. Syst.* **2024**, *29*, Article 24.
24. Saarinen, M.J.O. ASCON side-channel analysis and countermeasures. *J. Cryptogr. Eng.* **2023**, *13*, 421–436.