IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN XXXX-XXXX, Vol. 0, No. 0, pp. 1–7. DOI:XXXXXXXX

ML-DSA Digital Signatures in Resource-Constrained MQTT Environments

Jiahao Xiang[1] and Lang Li[1]

Hengyang Normal University, College of Computer Science and Technology, Hengyang, China

Abstract. The imminent threat of large-scale quantum computers necessitates the migration of Internet of Things (IoT) systems to post-quantum cryptographic standards. While NIST has standardized ML-DSA (Module-Lattice-Based Digital Signature Algorithm) for digital signatures, the practical deployment of post-quantum authentication in resource-constrained IoT environments remains unexplored. This research evaluates ML-DSA integration within MQTT-based IoT systems through comprehensive performance analysis on ARM Cortex-M4 microcontrollers. Our methodology encompasses signature generation and verification benchmarking, memory utilization analysis, and protocol overhead assessment under realistic IoT constraints. We analyze the performance implications of ML-DSA deployment compared to classical signature schemes, examining computational overhead, memory requirements, and verification latency on resource-constrained devices. These findings will reveal fundamental trade-offs between post-quantum security and IoT performance requirements, providing critical insights for practical deployment strategies in resource-limited environments.

Keywords: Post-Quantum Cryptography · ML-DSA · MQTT Protocol · IoT Security · Resource-Constrained Devices

1 Introduction

The emergence of quantum computing poses an existential threat to current cryptographic infrastructures, necessitating systematic migration to post-quantum cryptographic standards across all computing domains. The National Institute of Standards and Technology (NIST) has formalized ML-DSA (Module-Lattice-Based Digital Signature Algorithm) within FIPS 204 [NIS24], establishing this CRYSTALS-Dilithium-based scheme as the primary standard for post-quantum digital signatures.

While post-quantum standardization represents significant theoretical progress, practical deployment encounters severe constraints in resource-limited environments. Internet of Things (IoT) systems exemplify these challenges, where computational, memory, and energy limitations fundamentally constrain cryptographic implementation choices. The MQTT protocol, widely adopted for IoT messaging due to its lightweight characteristics, becomes particularly problematic when post-quantum signatures impose prohibitive performance overhead on resource-constrained devices. This disparity creates a critical gap between standardization achievements and practical deployment feasibility in IoT environments.

This research systematically addresses the challenge of ML-DSA integration within MQTT-based IoT systems through comprehensive performance analysis on ARM Cortex-M4 microcontrollers. Our investigation encompasses three critical dimensions: signature operation benchmarking, memory utilization analysis, and protocol overhead assessment under realistic IoT constraints. Through comparative analysis with classical signature schemes, we quantify computational overhead, memory requirements, and verification latency to establish fundamental trade-offs between post-quantum security guarantees and IoT performance requirements.

The remainder of this paper is organized as follows: Section 2 presents background information on post-quantum cryptography and related work in IoT deployments. Section 3 provides an overview of the ML-DSA algorithm and its implementation considerations. Section 4 describes our implementation architecture for MQTT-based IoT systems. Section 5 details our experimental methodology for performance evaluation on ARM Cortex-M4 microcontrollers. Section 6 presents and analyzes our experimental results, examining the trade-offs between security and performance. Finally, Section 7 concludes with implications for practical deployment and future research directions.

2 Related Work and Motivation

The transition from theoretical post-quantum standardization to practical deployment has revealed fundamental implementation challenges that extend beyond algorithmic considerations. While conventional network protocols have undergone extensive analysis for post-quantum migration [KSD20, SKD20], IoT-specific communication protocols remain inadequately addressed, creating deployment barriers in resource-constrained environments.

2.1 Post-Quantum Signature Performance in Embedded Systems

Empirical evaluations consistently demonstrate that post-quantum signature schemes impose substantial computational overhead on ARM Cortex-M microcontrollers commonly deployed in IoT devices [BZB+22, Mar24]. These performance implications manifest across multiple operational dimensions.

Signature size analysis reveals substantial increases for post-quantum schemes. ML-DSA signatures range from 2,420 bytes (Level 1) to 4,595 bytes (Level 5) compared to 64 bytes for ECDSA, representing 30-70× size increases. These expanded signature sizes, combined with elevated computational demands, frequently exceed the processing capabilities of resource-constrained devices.

Banegas et al. [BZB+22] quantified these performance implications through comprehensive benchmarking on embedded systems, establishing that CRYSTALS-Dilithium signature operations require approximately 45% additional computational cycles compared to classical ECDSA implementations on ARM Cortex-M4 processors. This computational overhead compounds with memory constraints to create deployment bottlenecks in IoT environments.

2.2 Deployment Bottlenecks in IoT Applications

Practical deployment scenarios reveal critical performance bottlenecks that challenge IoT system viability. Analysis of the SUIT (Software Update for the Internet of Things) framework demonstrates that post-quantum signature verification operations require up to 3.2 seconds on low-power microcontrollers, substantially exceeding acceptable latency constraints for real-time IoT applications.

These performance constraints are compounded by security vulnerabilities in embedded implementations. Fault injection research targeting ML-DSA and ML-KEM implementations achieved 89.5% attack success rates on ARM Cortex-M processors through electromagnetic fault injection techniques [WYQ+24]. The analyses demonstrate that Keccak-based hash functions—integral to ML-DSA randomness generation and signature computation—exhibit particular susceptibility to loop-abort faults enabling complete private key recovery, necessitating additional countermeasures that further impact performance.

2.3 Alternative Approaches and Limitations

Recent research has explored alternative authentication architectures to address these deployment challenges. Kim and Seo [KS25] demonstrate that direct application of post-quantum signatures to MQTT authentication introduces prohibitive performance overhead, prompting KEM-based authentication architectures that eliminate signature operations entirely. While their CRYSTALS-Kyber implementation achieves 4.32-second handshake completion on 8-bit AVR microcontrollers, this approach circumvents rather than resolves the fundamental challenge of post-quantum signature deployment.

Signature-based authentication mechanisms remain essential for applications requiring cryptographic non-repudiation, comprehensive audit trails, and compatibility with existing public key infrastructure frameworks. Consequently, optimization approaches targeting signature operations become critical for IoT deployment feasibility.

Current algorithmic optimization research reveals limitations in addressing fundamental resource constraints. Barrett multiplication techniques achieve 1.38-1.51× performance improvements on ARM Cortex-M3 processors and 6.37-7.27× improvements on 8-bit AVR platforms [HKS25]. However, these optimizations provide insufficient performance gains to bridge the gap between post-quantum signature requirements and IoT device capabilities, necessitating comprehensive system-level analysis.

2.4 Research Gap and Motivation

The absence of comprehensive empirical studies specifically evaluating ML-DSA performance within MQTT protocol implementations represents a critical knowledge gap in post-quantum IoT deployment. This gap becomes particularly significant given MQTT's widespread adoption in industrial IoT deployments, where signature-based authentication remains mandatory for regulatory compliance and security audit requirements.

Existing research primarily focuses on isolated cryptographic operations or alternative protocol architectures, failing to address the systematic integration challenges inherent in MQTT-based IoT systems. This research addresses these limitations through comprehensive performance analysis of ML-DSA deployment within realistic MQTT environments, providing essential insights for practical post-quantum IoT migration strategies.

3 ML-DSA Algorithm Overview

The Module-Lattice-Based Digital Signature Algorithm (ML-DSA) represents NIST's standardized post-quantum digital signature scheme, formalized in FIPS 204 [NIS24]. Derived from the CRYSTALS-Dilithium algorithm, ML-DSA employs lattice-based cryptographic constructions to provide quantum-resistant digital signature capabilities while preserving computational feasibility for existing computational architectures.

3.1 Mathematical Foundation

ML-DSA constructs digital signatures using the Fiat-Shamir With Aborts paradigm applied to module lattice problems. The algorithm's security relies on two fundamental computational assumptions: the Module Learning With Errors (MLWE) problem and a variant of the Module Short Integer Solution (MSIS) problem termed SelfTargetMSIS [NIS24]. These problems operate over polynomial rings $R_q = \mathbb{Z}_q[X]/(X^{256} + 1)$ where $q = 2^{23} - 2^{13} + 1 = 8380417$. The ring structure facilitates efficient polynomial arithmetic through Number Theoretic Transform (NTT) operations, enabling $O(n \log n)$ multiplication complexity essential for practical deployment.

The core cryptographic relationship underlying ML-DSA signatures follows the lattice equation $\mathbf{t} = \mathbf{A}\mathbf{s_1} + \mathbf{s_2}$, where $\mathbf{A} \in R_q^{k \times \ell}$ represents a public matrix, and $\mathbf{s_1} \in R_q^{\ell}$, $\mathbf{s_2} \in R_q^{k}$

constitute secret vectors with coefficients bounded by parameter η. This construction extends traditional discrete logarithm-based signature schemes to the lattice setting while incorporating rejection sampling techniques to eliminate statistical bias that could leak private key information.

3.2 Algorithm Structure

ML-DSA implements three primary algorithms: key generation (Algorithm 1), signature generation (Algorithm 2), and signature verification (Algorithm 3). Each algorithm operates within the module lattice framework while accommodating practical deployment constraints through optimized parameter selections.

Algorithm 1 ML-DSA Key Generation

1: **Input:** Security parameter defining (k, ℓ, η, d)
2: **Output:** Public key pk, Private key sk
3: $\xi \leftarrow \{0,1\}^{256}$ {Generate random seed}
4: $(\rho, \rho', K) \leftarrow H(\xi \| \text{IntegerToBytes}(k,1) \| \text{IntegerToBytes}(\ell,1), 128)$
5: $\hat{\mathbf{A}} \leftarrow \text{ExpandA}(\rho)$ {Generate public matrix in NTT form}
6: $(\mathbf{s_1}, \mathbf{s_2}) \leftarrow \text{ExpandS}(\rho')$ {Sample secret vectors with $\|\mathbf{s_i}\|_\infty \leq \eta$}
7: $\mathbf{t} \leftarrow \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\mathbf{s_1})) + \mathbf{s_2}$
8: $(\mathbf{t_1}, \mathbf{t_0}) \leftarrow \text{Power2Round}(\mathbf{t})$ {Compress with $d = 13$ bits}
9: $pk \leftarrow \text{pkEncode}(\rho, \mathbf{t_1})$
10: $tr \leftarrow H(pk, 64)$
11: $sk \leftarrow \text{skEncode}(\rho, K, tr, \mathbf{s_1}, \mathbf{s_2}, \mathbf{t_0})$
12: **return** (pk, sk)

As shown in Algorithm 1, the compression technique (line 6) reduces public key size by discarding the $d = 13$ least significant bits from each coefficient, as these low-order bits can be reconstructed from signature information during verification.

Algorithm 2 demonstrates the rejection sampling mechanism (lines 9-18) that ensures signatures maintain statistical independence from private key values, with expected iteration counts of 4.25, 5.1, and 3.85 for ML-DSA-44, ML-DSA-65, and ML-DSA-87 respectively. The validity checks in lines 11-12 and 16-17 determine whether each iteration produces an acceptable signature or requires repetition.

Algorithm 3 illustrates the deterministic verification process that reconstructs the signer's commitment using the hint mechanism (line 12). The critical verification steps include commitment reconstruction (line 11), hint application (line 12), and dual validation through both norm bounds and hash consistency (line 14).

3.3 Parameter Sets and Security Analysis

NIST standardizes three ML-DSA parameter sets with distinct security-performance trade-offs. Table 1 summarizes the key parameters and resulting implementation characteristics.

These parameter selections demonstrate the fundamental trade-off between quantum security strength and implementation overhead. Signature sizes increase by factors of 30-70× relative to classical ECDSA schemes, reflecting the inherent cost of lattice-based post-quantum security.

3.4 Computational Complexity Analysis

ML-DSA implementation requires quantified resource analysis for deployment feasibility assessment in constrained environments. Algorithm 1 exhibits $O(k \cdot \ell \cdot n \log n)$ complexity dominated by matrix-vector multiplication operations in line 5. Algorithm 2 complexity

Algorithm 2 ML-DSA Signature Generation

1: **Input:** Private key sk, Message M, Context ctx
2: **Output:** Signature σ or $\perp$
3: $(\rho, K, tr, \mathbf{s_1}, \mathbf{s_2}, \mathbf{t_0}) \leftarrow \text{skDecode}(sk)$
4: $\hat{\mathbf{A}} \leftarrow \text{ExpandA}(\rho)$
5: $M' \leftarrow \text{BytesToBits}(\text{IntegerToBytes}(0,1) \| \text{IntegerToBytes}(|ctx|,1) \| ctx) \| M$
6: $\mu \leftarrow H(\text{BytesToBits}(tr) \| M', 64)$
7: $rnd \leftarrow \{0,1\}^{256}$ {For hedged signing; use $\{0\}^{256}$ for deterministic}
8: $\rho'' \leftarrow H(K \| rnd \| \mu, 64)$
9: $\kappa \leftarrow 0$
10: **repeat**
11: $\quad \mathbf{y} \leftarrow \text{ExpandMask}(\rho'', \kappa)$ {Sample commitment with $\|\mathbf{y}\|_\infty \leq \gamma_1$}
12: $\quad \mathbf{w} \leftarrow \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\mathbf{y}))$
13: $\quad \mathbf{w_1} \leftarrow \text{HighBits}(\mathbf{w})$
14: $\quad \tilde{c} \leftarrow H(\mu \| \text{w1Encode}(\mathbf{w_1}), \lambda/4)$
15: $\quad c \leftarrow \text{SampleInBall}(\tilde{c})$ {Challenge with $\|c\|_0 = \tau$}
16: $\quad \mathbf{z} \leftarrow \mathbf{y} + c\mathbf{s_1}$
17: $\quad \mathbf{r_0} \leftarrow \text{LowBits}(\mathbf{w} - c\mathbf{s_2})$
18: $\quad$ **if** $\|\mathbf{z}\|_\infty \geq \gamma_1 - \beta$ OR $\|\mathbf{r_0}\|_\infty \geq \gamma_2 - \beta$ **then**
19: $\quad\quad \kappa \leftarrow \kappa + \ell$
20: $\quad\quad$ **continue**
21: $\quad$ **end if**
22: $\quad \mathbf{h} \leftarrow \text{MakeHint}(-c\mathbf{t_0}, \mathbf{w} - c\mathbf{s_2} + c\mathbf{t_0})$
23: $\quad$ **if** $\|c\mathbf{t_0}\|_\infty \geq \gamma_2$ OR $\|\mathbf{h}\|_0 > \omega$ **then**
24: $\quad\quad \kappa \leftarrow \kappa + \ell$
25: $\quad\quad$ **continue**
26: $\quad$ **end if**
27: $\quad \sigma \leftarrow \text{sigEncode}(\tilde{c}, \mathbf{z}, \mathbf{h})$
28: $\quad$ **return** σ
29: **until** false

varies with rejection sampling iterations, requiring $O(E[\text{iterations}] \cdot \ell \cdot n \log n)$ expected operations where $E[\text{iterations}]$ ranges from 3.85 to 5.1 across parameter sets. Algorithm 3 maintains deterministic $O((k+\ell) \cdot n \log n)$ complexity through the commitment reconstruction process in line 11.

Memory requirements encompass both static storage and dynamic working memory. Static storage includes private keys (2,560-4,896 bytes), public keys (1,312-2,592 bytes), and precomputed NTT constants (approximately 1KB). Dynamic memory requirements include polynomial vector storage ($k \cdot \ell \cdot 256 \cdot 4$ bytes for intermediate calculations) and matrix buffers for NTT operations.

3.5 Implementation Variants and Security Trade-offs

ML-DSA provides deterministic and hedged signing variants that present distinct security-performance trade-offs for resource-constrained deployments. Deterministic signing eliminates entropy requirements during signature generation, reducing hardware complexity but introducing vulnerability to side-channel and fault injection attacks. Hedged signing incorporates fresh randomness during each signature operation, providing enhanced resistance to implementation attacks while requiring secure random number generation capabilities that may be unavailable in minimal IoT implementations.

Algorithm 3 ML-DSA Signature Verification

Input: Public key pk, Message M, Signature σ, Context ctx
Output: true or false
$(\rho, \mathbf{t_1}) \leftarrow \text{pkDecode}(pk)$
$(\tilde{c}, \mathbf{z}, \mathbf{h}) \leftarrow \text{sigDecode}(\sigma)$
if $\mathbf{h} = \perp$ **then**
  return false {Invalid hint encoding}
end if
$\hat{\mathbf{A}} \leftarrow \text{ExpandA}(\rho)$
$tr \leftarrow H(pk, 64)$
$M' \leftarrow \text{BytesToBits}(\text{IntegerToBytes}(0,1) \| \text{IntegerToBytes}(|ctx|,1) \| ctx) \| M$
$\mu \leftarrow H(\text{BytesToBits}(tr) \| M', 64)$
$c \leftarrow \text{SampleInBall}(\tilde{c})$
$\mathbf{w}' \leftarrow \text{NTT}^{-1}(\hat{\mathbf{A}} \circ \text{NTT}(\mathbf{z}) - \text{NTT}(c) \circ \text{NTT}(\mathbf{t_1} \cdot 2^d))$
$\mathbf{w}'_1 \leftarrow \text{UseHint}(\mathbf{h}, \mathbf{w}')$
$\tilde{c}' \leftarrow H(\mu \| \text{w1Encode}(\mathbf{w}'_1), \lambda/4)$
return $\|\mathbf{z}\|_\infty < \gamma_1 - \beta$ AND $\tilde{c} = \tilde{c}'$

Table 1: ML-DSA Parameter Sets and Implementation Characteristics

Parameter	ML-DSA-44	ML-DSA-65	ML-DSA-87
Security Category	2 (AES-128)	3 (AES-192)	5 (AES-256)
Matrix (k, ℓ)	$(4, 4)$	$(6, 5)$	$(8, 7)$
Private Key (bytes)	2,560	4,032	4,896
Public Key (bytes)	1,312	1,952	2,592
Signature (bytes)	2,420	3,309	4,627
Expected Iterations	4.25	5.1	3.85

4 Implementation Architecture

5 Experimental Methodology

6 Results and Analysis

7 Conclusion

References

[BZB+22] Gustavo Banegas, Koen Zandberg, Emmanuel Baccelli, Adrian Herrmann, and Benjamin Smith. Quantum-resistant security for software updates on low-power networked embedded devices. In *Applied Cryptography and Network Security*, ACNS 2022. Springer, 2022. Also available at: https://eprint.iacr.org/2021/781.

[HKS25] Vincent Hwang, YoungBeom Kim, and Seog Chung Seo. Multiplying polynomials without powerful multiplication instructions. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2025(1):160–202, 2025. Extended version of Barrett Multiplication for Dilithium on Embedded Devices.

[KS25] YoungBeom Kim and Seog Chung Seo. An optimized instantiation of post-quantum MQTT protocol on 8-bit AVR sensor nodes. In *Proceedings of the 2025 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '25, pages 1–19. ACM, 2025.

[KSD20] Panos Kampanakis, Dimitrios Sikeridis, and Michael Devetsikiotis. Post-quantum authentication in tls 1.3: A performance study. In *Proceedings 2020 Network and Distributed System Security Symposium*. Internet Society, 2020.

[Mar24] Dominik Marchsreiter. Towards quantum-safe blockchain: Exploration of pqc and public-key recovery on embedded systems. Cryptology ePrint Archive, Paper 2024/1178, 2024.

[NIS24] NIST. Fips 204: Module-lattice-based digital signature standard. Federal Information Processing Standards Publication, 2024. Available at: https://csrc.nist.gov/pubs/fips/204/final.

[SKD20] Dimitrios Sikeridis, Panos Kampanakis, and Michael 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*, pages 149–156. ACM, 2020.

[WYQ+24] Yuxuan Wang, Jintong Yu, Shipei Qu, Xiaolin Zhang, Xiaowei Li, Chi Zhang, and Dawu Gu. Mind the faulty keccak: A practical fault injection attack scheme apply to all phases of ml-kem and ml-dsa. Cryptology ePrint Archive, Paper 2024/1522, 2024.