CALCULATION

Volume ***2****, Issue* ***1****, 2026, Pages:****26–33***

E-ISSN: 3062–2107

www.simadp.com/calculation

A COMPUTATIONAL BENCHMARK OF STANDARDIZED LATTICE-BASED POST-QUANTUM SIGNATURE SCHEMES

MELIKE KARATAY *

Abstract. This article provides an expanded computational evaluation of three standardized lattice-based post-quantum signature schemes—ML-DSA-44, ML-DSA-65 and Falcon-512—using reference implementations from the Open Quantum Safe (liboqs) project. Unlike prior benchmarking efforts that rely on optimized implementations or hardware-specific tuning, this work focuses on reproducible baseline performance on a conventional CPU platform. We measure key generation, signing and verification times across 5,000 iterations and analyze the effect of message length on signing cost. Results confirm that ML-DSA variants achieve significantly faster signing and key generation, whereas Falcon-512 produces substantially smaller keys and signatures. The study further highlights the computational dominance of lattice operations over hashing, explaining the weak dependence of signing cost on message size. These findings aim to support system architects evaluating the practical feasibility of standardized post-quantum signature algorithms.

Keywords: Post-quantum cryptography, Lattice-based signatures, Performance evaluation

2020 Mathematics Subject Classification: 94A60, 68P25.

1. Introduction

The anticipated emergence of large-scale quantum computers poses a fundamental threat to classical public-key cryptosystems whose security relies on the presumed hardness of integer factorization and discrete logarithms. In response, substantial global effort has been devoted to the development of post-quantum cryptography (PQC), aiming to design cryptographic primitives that remain secure against both classical and quantum adversaries. Among the candidate families explored during the NIST Post-Quantum Cryptography Standardization Project [4], lattice-based cryptography has emerged as the most prominent due to its strong worst-case to average-case reductions [1], conceptual simplicity, and highly efficient algebraic structure [3].

Digital signature schemes constitute one of the central components of the PQC transition. ML-DSA, formerly known as CRYSTALS–Dilithium, was selected by NIST as the primary

Received: 2025.10.17 *Revised: 2025.12.16* *Accepted: 2025.12.22*

* Corresponding author

Melike Karatay ⋄ melike.karatay@fbu.edu.tr ⋄ https://orcid.org/0000-0001-6941-4752

post-quantum digital signature standard [4, 5]. ML-DSA relies on module-lattice constructions and the Fiat–Shamir-with-Abort transformation, achieving a favorable balance between security, determinism, and implementation simplicity. Falcon, another lattice-based signature scheme, was also advanced by NIST due to its compact keys and signatures and its mathematically elegant use of Fourier-based Gaussian sampling [6]. These schemes represent two distinct design philosophies within lattice-based cryptography, making their comparative evaluation both practically and theoretically significant.

A considerable body of benchmark literature exists for lattice-based signatures. Highly optimized implementations are included in the SUPERCOP benchmarking suite [9], the PQClean project provides portable and optimized CPU backends [10], and numerous studies evaluate performance on ARM microcontrollers [11], vectorized x86-64 architectures exploiting AVX2/AVX-512 instruction sets [12], or hardware-accelerated co-processors. While these studies offer valuable insights, they do not necessarily reflect the default, platform-neutral performance of unoptimized reference implementations. Reference implementations emphasize clarity and correctness over speed and therefore constitute a reproducible baseline, independent of platform-specific tuning and unavailable hardware features.

This motivates the present work. We perform a systematic, reproducible benchmark of ML-DSA-44, ML-DSA-65, and Falcon-512 using their unmodified reference C implementations from the Open Quantum Safe (liboqs) project. By avoiding compiler-dependent and architecture-dependent optimizations, our analysis isolates the inherent computational characteristics of each scheme. We further examine the influence of message length on signing cost and relate the results to the internal structure of the signing algorithms. In particular, previous studies show that lattice operations overwhelmingly dominate runtime, while hashing (e.g., SHAKE256) contributes only marginal overhead [8, 7]. Our results empirically confirm this behavior.

The contributions of this work are:

- A reproducible, platform-neutral performance benchmark of standardized lattice-based signature schemes using only reference implementations.
- An analysis of the effect of message size on signing time, with theoretical justification rooted in the structure of lattice arithmetic and hashing.
- A comparative discussion situating the results within the broader PQC benchmarking literature, highlighting implications for system architects and protocol designers.

This expanded analysis strengthens the empirical understanding of the standardized post-quantum signature landscape and provides actionable insights for researchers and practitioners preparing for PQC deployment.

2. Preliminaries

Lattice-based cryptography provides the foundation for several post-quantum signature schemes standardized or recommended by NIST. Its security relies on well-studied reductions from worst-case to average-case, particularly those associated with the Learning With Errors (LWE) problem and its structured variants such as Module-LWE and Module-SIS [1, 3]. These assumptions allow for the construction of efficient signature algorithms whose security proofs are rooted in hardness results from lattice theory.

ML-DSA, the primary NIST-selected post-quantum signature standard [4, 5], employs module lattices to construct a Fiat–Shamir with Aborts signature system. Its design emphasizes determinism, simplicity, and implementation clarity. At a high level, ML-DSA replaces Gaussian sampling with uniformly random sampling followed by rejection sampling, making it easier to implement securely while maintaining strong theoretical guaranties. The parameter sets ML-DSA-44 and ML-DSA-65 correspond to the former Dilithium-II and Dilithium-III security levels.

Falcon, in contrast, is built upon the NTRU lattice structure and employs fast Fourier sampling to generate lattice Gaussian samples [6]. This enables Falcon to achieve significantly smaller public keys and signatures, but at the cost of more computationally expensive key generation and signing operations. The complexity arises from the need for numerically stable discrete Gaussian sampling, a problem that has been extensively analyzed in the lattice-cryptography literature [7].

Hashing also plays a crucial role in lattice-based signatures. Both ML-DSA and Falcon use variants of SHAKE (SHAKE-128 or SHAKE-256) as part of their signing procedures, either for message compression or challenge generation [2]. Since hashing cost grows linearly with message length while lattice operations dominate computational complexity, the hashing component typically contributes only a small fraction of total signing time [8]. This explains why, in practice, signing performance is largely insensitive to message length—a behavior confirmed in Section 3.

All schemes tested in this study were benchmarked using their reference implementations from the Open Quantum Safe (liboqs) project [10]. Reference implementations prioritize portability, clarity, and standard compliance rather than speed. As a result, they provide a reproducible baseline for comparing algorithms without the confounding influence of architecture-dependent optimizations such as AVX2/AVX-512 vectorization [9, 12]. This makes them particularly suitable for evaluating the intrinsic computational characteristics of standardized post-quantum signatures.

2.1. **Experimental Setup.** All experiments in this work were performed on a workstation equipped with an **Intel Core i7-1165G7** processor (11th generation, 4 cores, 8 threads, base frequency 2.8 GHz, maximum turbo frequency 4.7 GHz) with full **AVX2** support. The system was configured with **32 GB RAM** and running **Ubuntu 24.04.1 LTS (64-bit)**. The detailed specification of the processor is essential because lattice-based cryptography is highly sensitive to vector instruction sets and microarchitectural differences [12, 11]. Specifying the exact model ensures reproducibility, as the runtime of lattice multiplications, NTT operations, and Gaussian sampling can vary significantly across CPU generations.

All implementations were compiled using **GCC 14.2** with the `-O3` optimization flag enabled. GCC 14 is a mature and stable compiler branch, and provides reliable support for the C implementations of liboqs and its post-quantum algorithm modules. No architecture-specific optimizations (e.g., `-march=native`, AVX-512 extensions, or handcrafted intrinsics) were enabled in order to preserve portability and ensure that the measurements reflect the behavior of the unoptimized reference implementations.

All signature schemes in this study were obtained from the **Open Quantum Safe (liboqs) reference implementation** [10]. The library was compiled from source against the

system's **OpenSSL 3.x** installation using its default reference configuration. No vendor-specific patches, no deterministic or randomized Gaussian sampling accelerations, and no fast polynomial multiplication overrides were applied.

Timing measurements were performed using `clock_gettime()` with the `CLOCK_MONOTONIC` flag. For each scheme, **5,000 iterations** of key generation, signing, and verification were executed, and the average runtime per operation was recorded. This iteration count follows common practice in PQC benchmarking literature to eliminate noise from short-lived operations [9, 11].

Benchmarking was performed using the baseline, non-optimized reference implementations to ensure reproducibility and consistency with the NIST PQC standardization documents. Timing measurements were obtained using clock_gettime() with CLOCK_MONOTONIC, and each operation (key generation, signing, verification) was averaged over 5,000 iterations.

3. Experimental Results

This section presents the empirical evaluation of ML-DSA-44, ML-DSA-65, and Falcon-512 using the reference implementations from the Open Quantum Safe (liboqs) framework. All results reflect the performance of non-optimized, platform-neutral implementations and provide a baseline for reproducible comparison, in contrast to prior studies relying on heavily optimized code paths [9, 10, 12]. Each operation was executed 5,000 times, and average runtimes were recorded.

3.1. **Memory Footprint.** Table 3.1 reports public key, secret key, and signature sizes for each scheme. These values are fixed parameters determined by the underlying lattice constructions. Falcon-512 achieves substantially smaller keys and signatures, a known consequence of its NTRU-based structure and its compact sampling techniques [6]. By comparison, ML-DSA variants feature larger public and secret keys due to their module-lattice structure and rejection-based signing design [5].

Table 3.1. Key and signature sizes (bytes)

Algorithm	Public Key	Secret Key	Signature
ML-DSA-44	1312	2460	2420
ML-DSA-65	1952	4032	3309
Falcon-512	897	1281	752

The differences observed here align with existing literature: Falcon is typically selected for size-constrained environments (IoT devices, embedded platforms), whereas ML-DSA is often preferred in high-throughput server environments where bandwidth is not the primary limitation [11, 8].

3.2. **Key Generation, Signing, and Verification Times.** Table 3.2 summarizes the average runtime (ms/op) for key generation, signing, and verification using 32-byte messages. The results indicate that ML-DSA-44 exhibits the fastest key generation and signing performance. ML-DSA-65, which targets a higher NIST security level, remains computationally efficient despite its increased parameter sizes. Falcon-512, however, presents a significantly different performance profile: **key generation is nearly two orders of magnitude slower**.

TABLE 3.2. Average runtime (ms/op) for 32-byte messages

Algorithm	KeyGen	Sign	Verify
ML-DSA-44	0.03075	0.08019	0.02803
ML-DSA-65	0.04961	0.12638	0.04743
Falcon-512	6.74992	0.22014	0.04794

This observation is consistent with prior reports documenting the high computational cost of its floating-point Gaussian sampling algorithm [6, 8]. Gaussian sampling dominates Falcon's key generation, whereas ML-DSA relies on more lightweight polynomial operations and deterministic rejection sampling.

Verification performance is relatively consistent across schemes, with ML-DSA variants achieving marginally faster verification than Falcon, in line with prior benchmark studies [9, 10].

3.3. **Effect of Message Length on Signing Time.** Table 3.3 presents the signing time as a function of message length (32 bytes, 1 KB, 16 KB). All three schemes exhibit only modest increases in signing cost as the message grows. This behavior is theoretically expected because post-quantum signature schemes adopt a **hash-then-sign** paradigm: the message is first compressed using a hash function (e.g., SHAKE256), and the resulting digest—not the full message—is incorporated into the lattice-based signing procedure [8]. As a result, the

TABLE 3.3. Average signing time (ms/op) as a function of message length

Algorithm	32 B	1 KB	16 KB
ML-DSA-44	0.08019	0.08534	0.13618
ML-DSA-65	0.12638	0.12718	0.17833
Falcon-512	0.22014	0.22285	0.26021

dominant computational cost is the internal lattice arithmetic (polynomial multiplications, NTT-based transforms, sampling steps), rather than message processing. Our results empirically confirm prior claims that hashing contributes only a negligible fraction of the total runtime [8, 7].

3.4. **Comparison with Existing Benchmark Literature.** Compared with SUPERCOP and PQClean results, which often rely on AVX2-optimized or handcrafted assembly implementations [9, 10, 12], the runtimes reported here are notably slower. This discrepancy is expected because our experiments deliberately use reference implementations, which trade speed for portability and clarity.

Nevertheless, the relative ordering among schemes—ML-DSA faster than Falcon for signing and key generation, Falcon significantly smaller in memory footprint—remains consistent with the performance trends documented in earlier PQC studies [5, 6].

These results therefore provide a clean, reproducible baseline that complements the highly optimized benchmarking performed in prior work, while removing architectural dependencies that often obscure true algorithmic cost.

4. Results and Discussion

The empirical findings presented in Section 3 reveal distinct computational profiles for the three standardized lattice-based signature schemes examined in this study. The observed trends are consistent with the theoretical structure of the algorithms as well as previously published benchmark results [9, 10, 5, 6].

4.1. **Comparative Performance Interpretation.** ML-DSA-44 demonstrates the fastest performance in both key generation and signing. This is expected due to its relatively small parameter set and the efficiency of the Fiat–Shamir-with-abort paradigm, which relies primarily on structured polynomial multiplications and lightweight rejection sampling. As the security level increases, ML-DSA-65 incurs higher computational cost, but the increase remains moderate. This aligns with analytic observations that the module dimension and polynomial degree scale in a controlled manner within the ML-DSA parameter hierarchy [5].

Falcon-512, by contrast, exhibits a markedly different computational pattern. Although it achieves significantly smaller public keys, secret keys, and signatures—an asset particularly valuable in constrained environments—its key generation time is nearly two orders of magnitude slower than that of ML-DSA. This gap is well documented in the NIST PQC literature and arises from Falcon's reliance on Fourier-based Gaussian sampling, which requires numerically stable floating-point operations and careful error control [6, 7]. These operations are both computationally dense and highly sensitive to microarchitectural details, which explains the large performance disparity even on modern processors equipped with AVX2 support [12].

Verification performance is more uniform across schemes. ML-DSA variants maintain a slight advantage, but the differences remain small, consistent with prior SUPERCOP and PQClean evaluations [9, 10]. This suggests that verification cost is not a primary bottleneck for any of the standardized schemes, even in high-throughput environments such as certificate validation pipelines or TLS handshake scenarios.

4.2. **Impact of Message Length: Hashing vs. Lattice Computation.** A key observation from Table 3.3 is that signing time increases only minimally as the message length grows from 32 bytes to 16 KB. This behavior is theoretically expected: all NIST-standardized PQC signature schemes adopt a hash-then-sign construction where the raw message is first compressed into a fixed-length digest before entering the lattice-based signing procedure.

Since the cost of hashing (e.g., SHAKE256) is negligible compared with lattice operations such as NTT-based polynomial multiplication or Gaussian sampling, the total signing time is effectively dominated by the latter [8]. Our results empirically confirm this theoretical expectation and reinforce the idea that signature performance in PQC systems is largely independent of message size, simplifying performance modeling for real-world deployments.

4.3. **Comparison with Optimized Implementations.** It is important to emphasize that the results reported here differ substantially from those obtained using optimized implementations such as the AVX2-enhanced PQClean backends or the handcrafted SUPERCOP assembly routines [9, 10, 12]. In optimized environments, Falcon often closes much of the performance gap with ML-DSA, particularly in signing and verification, while still lagging

in key generation. In contrast, our reference-only measurements provide a platform-neutral baseline that isolates the inherent computational cost of each scheme independent of hardware capabilities.

This distinction highlights a broader methodological point: optimized benchmarks answer the question *"What is the maximum attainable performance on a specific architecture?"*, while reference benchmarks answer *"What is the intrinsic cost of the algorithm itself?"*. Both perspectives are essential to understanding real-world PQC deployments.

4.4. **Implications for Deployment and System Design.** From an applied cryptography perspective, the results have direct implications for the design of PQC-enabled systems:

- **High-throughput environments:** ML-DSA-44 and ML-DSA-65 are strong candidates for systems that require frequent key generation or bulk signing, such as certificate authorities, distributed authentication services, and blockchain-based consensus networks.
- **Bandwidth- or memory-constrained environments:** Falcon-512 remains attractive for applications where key or signature size is critical—such as embedded IoT devices, mobile systems, or satellite communication—despite its slower key generation.
- **Hybrid cryptographic protocols:** These findings support hybrid PQC–classical deployments where Falcon may be combined with classical signatures for compact transmission, while ML-DSA may be preferred for server-side signing workloads.

Overall, the results reinforce a complementary relationship between ML-DSA and Falcon, consistent with NIST's decision to standardize both families. Rather than identifying a single "best" post-quantum signature scheme, the empirical evidence suggests a landscape in which algorithm choice is highly dependent on application requirements and resource constraints.

5. Conclusion

This work presented a reproducible performance evaluation of the standardized lattice-based post-quantum signature schemes ML-DSA-44, ML-DSA-65, and Falcon-512 using their reference implementations. The results show that ML-DSA variants provide the fastest key generation and signing performance, while Falcon-512 offers significantly smaller public keys, secret keys, and signatures at the cost of slower key generation. Verification times remain comparable across all schemes.

The experiments further indicate that signing time depends only weakly on message length, reflecting the dominance of lattice operations over hashing in overall computation. These findings highlight the practical trade-offs between speed and parameter size and demonstrate that the choice of signature scheme should be guided by the specific requirements of the target application.

Future work may focus on optimized implementations, alternative hardware platforms, and the behavior of these schemes in memory-constrained or side-channel-resistant settings.

References

[1] Regev, O. (2009). On lattices, learning with errors, random linear codes, and cryptography. Journal of the ACM (JACM), 56(6), 1-40.

[2] National Institute of Standards and Technology (NIST). (2015) FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Federal Information Processing Standards Publication 202, https://doi.org/10.6028/NIST.FIPS.202.

[3] Peikert, C. (2016). A decade of lattice cryptography. Foundations and trends in theoretical computer science, 10(4), 283-424.

[4] Alagic, G., Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T., ... & Smith-Tone, D. (2022). Status report on the third round of the NIST post-quantum cryptography standardization process.

[5] Ducas, L., Lepoint, T., Lyubashevsky, V., Schwabe, P., Seiler, G., & Stehlé, D. (2018). Crystals–dilithium: Digital signatures from module lattices.

[6] Fouque, P. A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T., Prest, T., ... & Zhang, Z. (2018). Falcon: Fast-Fourier lattice-based compact signatures over NTRU. Submission to the NIST's post-quantum cryptography standardization process, 36(5), 1-75.

[7] Prest, T. (2015). Gaussian sampling in lattice-based cryptography (Doctoral dissertation, Ecole normale supérieure-ENS PARIS).

[8] Katz, J., & Lindell, Y. (2007). Introduction to modern cryptography: principles and protocols. Chapman and hall/CRC.

[9] Bernstein, D. J., & Lange, T. (2009). Supercop: System for unified performance evaluation related to cryptographic operations and primitives.(2009).

[10] Kannwischer, M. J., Schwabe, P., Stebila, D., & Wiggers, T. (2022, June). Improving software quality in cryptography standardization projects. In 2022 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW) (pp. 19-30). IEEE.

[11] Howe, J., & Westerbaan, B. (2023, July). Benchmarking and analysing the NIST PQC lattice-based signature schemes standards on the ARM Cortex M7. In International Conference on Cryptology in Africa (pp. 442-462). Cham: Springer Nature Switzerland.

[12] Lei, D., He, D., Peng, C., Luo, M., Liu, Z., & Huang, X. (2023). Faster implementation of ideal lattice-based cryptography using avx512. ACM Transactions on Embedded Computing Systems, 22(5), 1-18.

(M. Karatay) Fenerbahçe University, İstanbul, Türkiye