Comparative Study of Lattice-Based Post-Quantum Cryptography Algorithms from NIST and Korea

Boyeon Song
Quantum Network Research Center
Korea Institute of Science and Technology Information
Daejeon, South Korea
bysong@kisti.re.kr

***Abstract*—Post-quantum cryptography (PQC) has emerged as a critical research area for securing communications in the quantum era. Among the selected algorithms, lattice-based schemes have gained particular attention due to their strong security foundations and practical efficiency.**

This paper presents a comparative analysis of lattice-based PQC algorithms from the U.S. National Institute of Standards and Technology (NIST) standardization process and the Korean PQC (KpqC) competition. The evaluation considers security levels, key and ciphertext/signature sizes, and implementation performance. The results highlight fundamental trade-offs between security, efficiency, compactness, and implementation complexity, providing insights into the strengths and limitations of current lattice-based algorithms. These findings offer practical guidance for assessing their suitability for deployment in real-world post-quantum systems.

***Index Terms*—post-quantum cryptography (PQC), lattice-based cryptography, NIST PQC standardization, Korean PQC competition (KpqC), key size, performance.**

I. Introduction

The advent of large-scale quantum computers poses a serious threat to conventional public-key cryptographic systems such as RSA, Diffie-Hellman, and elliptic curve cryptography, since these schemes can be efficiently broken by quantum algorithms. In particular, Shor's algorithm can solve the integer factorization and discrete logarithm problems in polynomial time, thereby rendering classical systems insecure once practical large-scale quantum computers become available. To address this challenge, the field of Post-Quantum Cryptography (PQC) has emerged, focusing on algorithms that remain secure against both classical and quantum adversaries.

In 2016, the U.S. National Institute of Standards and Technology (NIST) launched the PQC standardization project with the goal of selecting and standardizing quantum-resistant Public-Key Encryptions (PKEs), Key Encapsulation Mechanisms (KEMs), and Digital Signature (DS) schemes. Subsequently, in 2021, Korea's National Security Research Institute and National Intelligence Service established the Korean post-quantum Cryptography (KpqC) research group. To strengthen national security in the quantum era, the KpqC competition was initiated to develop Korean standard PQC schemes.

Among the various PQC families, lattice-based cryptography has attracted significant attention due to its strong security foundations and relatively efficient performance across diverse platforms. In this paper, we present a comparative analysis of lattice-based PQC algorithms from both NIST and KpqC. Our study examines their security levels, key and ciphertext/signature sizes, and implementation performance. Through this comparison, we identify the strengths and limitations of each approach, as well as their potential for adoption in real-world systems.

The remainder of this paper is organized as follows. Section II reviews the algorithms selected in the NIST PQC standardization process. Section III describes the final algorithms of the KpqC competition. Section IV compares lattice-based KEM and DS schemes in terms of security levels, parameter sizes and performance. Finally, Section V concludes the paper.

II. NIST PQC Algorithms

In 2016, NIST initiated a public PQC standardization process aimed at selecting quantum-resistant public-key cryptographic algorithms against quntum threats. A total of 82 candidate algorithms were submitted for consideration. After three rounds of rigorous evaluation, NIST announced the first set of algorithms selected for standardization on July 5, 2022: CRYSTALS-Kyber for PKE/KEM and CRYSTALS-Dilithium, FALCON, and SPHINCS$^+$ for DSs. More recently, on March 11, 2025, NIST announced HQC as a fourth-round selec-

This research was supported by Korea Institute of Science and Technology Information (KISTI).(No.K25L5M2C2-01)

tion, making it the second PKE/KEM algorithm to be standardized.

CRYSTALS-Kyber (commonly referred to as Kyber) and CRYSTALS-Dilithium (Dilithium) are two cryptographic primitives from the cryptographic suite for algebraic lattices (CRYSTALS), a package submitted to the NIST PQC standardization effort. Both algorithms rely on the hardness of the Module Learning With Errors (MLWE) problem and were ultimately selected for standardization by NIST for their strong security and excellent performance. Kyber [1] is a KEM secure against indistinguishability under adaptive chosen ciphertext attacks (IND-CCA2). Dilithium is a DS designed to be strong existential unforgeablility under chosen-message attack (SUF-CMA) [2]. On August 13, 2024, NIST published them as FIPS 203: Module-Lattice-Based Key Encapsulation Mechanism (ML-KEM) and FIPS 204: Module-Lattice-Based Digital Signature Algorithm (ML-DSA) [3], respectively. NIST expects ML-KEM and ML-DSA as two primary algorithms to work well in most use cases.

Fast-Fourier Lattice-based Compact signatures Over NTRU (FALCON) is derived from Nth degree Truncated polynomial Ring Units (NTRU) and is based on the Gentry–Peikert–Vaikuntanathan method for constructing lattice-based signature schemes, combined with a trapdoor sampler known as Fast Fourier Sampling [4]. Its security relies on the hardness of the Short Integer Solution problem over NTRU lattices. FALCON was selected by NIST for standardization in cases where Dilithium signatures are considered too large. Draft FIPS 206, derived from FALCON, is currently under development and is expected to be released in 2025.

SPHINCS$^+$ is a stateless hash-based signature scheme built from a hierarchy of few-time and multi-time hash-based signature components, including the forest of random subsets and the extended Merkle signature scheme [5]. It was selected by NIST to provide a non-lattice-based alternative to Dilithium and FALCON. On August 13, 2024, NIST published SPHINCS$^+$ as FIPS 205: Stateless Hash-Based Digital Signature Algorithm (SLH-DSA) [6].

Hamming Quasi-Cyclic (HQC) is a code-based PKE scheme. It achieves indistinguishability under chosen-plaintext attacks (IND-CPA) and can be transformed into an IND-CCA2-secure KEM using the Hofheinz, Hovelmanns, and Kiltz transform [7]. Its security is based on the hardness of the Syndrome Decoding problem for random Quasi-Cyclic codes. As a code-based construction, HQC offers an alternative to ML-KEM, which is lattice-based.

Table I provides a summary of the PQC algorithms selected for NIST standardization, as discussed above.

TABLE I: NIST PQC Standardization Algorithms

Purpose	Algorithm	Category	Selected
KEM	**CRYSTALS-Kyber**	Lattice-based	July 2022
	HQC	Code-based	Mar. 2025
DS	**CRYSTALS-Dilithium**	Lattice-based	July 2022
	FALCON	Lattice-based	
	SPHINCS$^+$	Hash-based	

III. KpqC Algorithms

In 2021, the KpqC competition has begun for the Korean standard PQC schemes. In November 2022, the first round began with 7 KEM candidates and 9 DS candidates. In December 2023, 4 KEM algorithms and 4 DS algorithms have been advanced to the second round. The final winners of the KpqC competition were announced in January 2025: SMAUG-T and NTRU+ for PKE/KEMs and HAETAE and AIMer for DSs.

SMAUG-T [8] is a lattice-based KEM constructed from the MLWE and Module Learning With Rounding (MLWR) problems. Its long-term security relies conservatively on the hardness of MLWE, while ephemeral keys are derived more efficiently using the MLWR framework [8]. Quantum security is ensured through the Fujisaki-Okamoto (FO) transform with decryption-failure handling, providing robustness against quantum adversaries while preserving practical efficiency. The scheme leverages the module structure (as in Kyber) together with sparse secrets (as in homomorphic encryption), enabling faster performance and reduced ciphertext size.

NTRU+ [9] is a recently proposed NTRU-based KEM that addresses both the security and performance limitations of the original NTRU [10]. Although NTRU was the first practical lattice-based PKE scheme defined over a polynomial ring, it exhibits several shortcomings, such as difficulties in establishing worst-case to average-case reductions under moderate modulus sizes, cumbersome message sampling distributions, and comparatively slower algorithms relative to other lattice-based constructions [9]. To mitigate these issues, NTRU+ introduces two generic transformations: ACWC_2 (Average-Case to Worst-Case) and $\overline{\mathrm{FO}}^{\perp}$, a variant of the FO transform that achieves chosen-ciphertext security without the need for re-encryption [9].

SMAUG-T and NTRU+ thus provide lattice-based KEM alternatives to ML-KEM, offering complementary design strategies that address efficiency, security, and practicality in the post-quantum setting.

Hyperball bimodal module rejection signature scheme (HAETAE) [11] is a lattice-based DS scheme. Similar to the NIST-selected Dilithium, it is built on the Fiat-Shamir with Aborts paradigm, which ensures quantum security in the quantum random oracle model. To reduce signature sizes, HAETAE employs a bimodal distribution

for rejection sampling, following the approach of the Bimodal Lattice Signature Scheme (BLISS) [12], instead of the unimodal distribution used in Dilithium [11]. Furthermore, HAETAE replaces the discrete Gaussian distributions of BLISS, which are known to be vulnerable to side-channel attacks, with uniform distributions over hyperballs [11].

AIMer [13] is a DS scheme derived from a zero-knowledge proof of preimage knowledge for a designated one-way function. The scheme comprises two main components: a non-interactive zero-knowledge proof of knowledge and the one-way function called AIM. The AIM primitive is a symmetric construction that is well-suited to the multi-party computation-in-the-head paradigm and exhibits strong resistance to algebraic attacks. The security of AIMer is grounded solely in the robustness of its underlying symmetric primitives.

In summary, HAETAE provides an alternative lattice-based signature scheme that complements the NIST-selected ML-DSA and FALCON, whereas AIMer serves as an alternative to hash-based signatures, complementing the NIST-selected SLH-DSA. Table II presents a consolidated overview of the final KpqC competition algorithms.

TABLE II: Final KpqC Competition Algorithms

Purpose	Algorithm	Category	Selected
KEM	**SMAUG-T**	Lattice-based	Jan. 2025
	NTRU+	Lattice-based	
DS	**HAETE**	Lattice-based	
	AIMer	Hash-based	

IV. Comparison of Lattice-based PQCs

Lattice-based PQC schemes have emerged as the leading algorithms in the NIST PQC standardization process, offering both provable security guarantees and competitive performance. As discussed in the preceding sections, most of the finalists in both the NIST PQC standardization and the KpqC competition are lattice-based. In this section, we provide a comparative analysis of lattice-based KEMs and DSs with respect to their security and performance characteristics.

NIST classifies PQC algorithms according to the range of security strengths defined by existing NIST standards in symmetric cryptography, which are expected to provide significant resistance against quantum cryptanalysis. In its original Call for Proposals [14], NIST introduced five security categories, numbered 1 through 5. These categories are based on the security strengths of approved symmetric-key standards and are intended to establish comparable levels of quantum-resistant security. Category 1 corresponds to the minimum approved security level, while category 5 represents the maximum. Further details on these categories can be found in NIST SP 800-57, Part 1.

A. Lattice-based KEMs

A KEM is a cryptographic scheme to establish a shared secret key between two communicating parties, for under certain conditions [15]. This shared secret key can then be used for symmetric-key cryptography [16]. A KEM consists of three main algorithms: (1) a probabilistic key generation algorithm, denoted by KeyGen; (2) a probabilistic encapsulation algorithm, denoted by Encaps; and (3) a deterministic decapsulation algorithm, denoted by Decaps, together with a collection of parameter sets corresponding to different security levels [16].

ML-KEM (formerly Kyber) specifies three parameter sets [16]: ML-KEM-512 in security category 1, ML-KEM-768 in security category 3, and ML-KEM-1024 in security category 5. ML-KEM-512 (Kyber-512) corresponds to a security level roughly comparable to AES-128, ML-KEM-768 (Kyber-768) to AES-192, and ML-KEM-1024 (Kyber-1024) to AES-256 [16]. Specifically, ML-KEM-512 generates an encapsulation key (EK) of 800 bytes, a decapsulation key (DK) of 1,632 bytes, and a ciphertext (CT) of 768 bytes; ML-KEM-768 generates an EK of 1,184 bytes, a DK of 2,400 bytes, and a CT of 1,088 bytes; and ML-KEM-1024 generates an EK of 1,568 bytes, a DK of 3,168 bytes, and a CT of 1,568 bytes.

SMAUG-T provides three parameter sets [8]: SMAUG-T-128 (security category 1), SMAUG-T-192 (security category 3), and SMAUG-T-256 (security category 5). SMAUG-T-128 generates an EK of 672 bytes, a DK of 176 bytes, and a CT of 672 bytes; SMAUG-T-192 generates an EK of 1,088 bytes, a DK of 236 bytes, and a CT of 1,024 bytes; and SMAUG-T-256 generates an EK of 1,792 bytes, a DK of 218 bytes, and a CT of 1,472 bytes.

NTRU+KEM provides four parameter sets [9]: NTRU+KEM576 (not considered here), NTRU+KEM768 (security category 1), NTRU+KEM864 (security category 3), and NTRU+KEM1152 (security category 5). NTRU+KEM768 generates an EK of 1,152 bytes, a DK of 2,336 bytes, and a CT of 1,152 bytes; NTRU+KEM864 generates an EK of 1,296 bytes, a DK of 2,624 bytes, and a CT of 1,296 bytes; and NTRU+KEM1152 generates an EK of 1,728 bytes, a DK of 3,488 bytes, and a CT of 1,728 bytes.

Table III compares the security categories (Cat.) and the sizes of the EK, DK, and CT for the lattice-based KEMs from NIST and KpqC, namely ML-KEM, SMAUG-T, and NTRU+KEM. Figure 1 illustrates these results in a bar graph, showing the EK, DK, and CT sizes for the three schemes.

For ML-KEM, the key and ciphertext sizes scale with the security level, ranging from 800/1,632/768 bytes at category 1 to 1,568/3,168/1,568 bytes at category 5, reflecting the scheme's consistent balance be-

TABLE III: Sizes (in bytes) of Keys and Ciphertexts of Lattice-based KEMs

Source	Scheme	Cat.	EK	DK	CT
NIST	**ML-KEM-512**	1	800	1632	768
	ML-KEM-768	3	1184	2400	1088
	ML-KEM-1024	5	1568	3168	1568
KpqC	**SMAUG-T-128**	1	672	176	672
	SMAUG-T-192	3	1088	236	1024
	SMAUG-T-256	5	1792	218	1472
	NTRU+KEM768	1	1152	2336	1152
	NTRU+KEM864	3	1296	2624	1296
	NTRU+KEM1152	5	1728	3488	1728

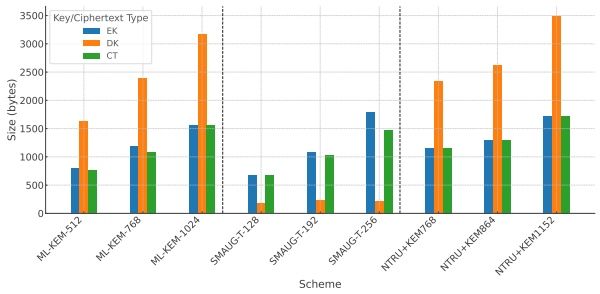

Fig. 1: Size comparison of keys and ciphertexts for lattice-based KEMs

tween compactness and security. SMAUG-T achieves notably smaller decapsulation key sizes compared to ML-KEM and NTRU+KEM, particularly at higher security categories, where its DK size remains under 300 bytes. This design significantly reduces memory requirements for secret key storage. NTRU+KEM, by contrast, provides slightly larger EK, DK and CT sizes compared to ML-KEM.

In [8], the performance of Kyber and SMAUG-T was evaluated on a single core of an Intel(R) Core(TM) i7-10700K processor at 3.80 GHz with 64 GB of RAM, running Debian GNU/Linux (kernel 5.4.0) and compiled with gcc version 11.4.0. The experiments adhered to the measurement methodology described in [8], with identical settings applied to both schemes.

Table IV presents benchmark results for Kyber and SMAUG-T across three parameter sets. Performance is evaluated for KeyGen (KG), Encaps (Enc), and Decaps (Dec) using both the baseline C reference implementation and an optimized implementation that leverages AVX2 vector instructions.

The reference results indicate that SMAUG-T achieves slightly lower cycle counts than Kyber for both Encaps and Decaps (*e.g.*, 100k vs. 158k for Enc, and 136k vs. 187k for Dec at security category 1). The AVX2-optimized benchmarks demonstrate substantial efficiency improvements for both schemes, reducing cycle counts by factors of 4–6. In this optimized setting, SMAUG-T offers lower Encaps costs across all parameter sets,

TABLE IV: Performance (in CPU kilocycles) of Kyber and SMAUG-T [8]

Scheme	KG	Enc	Dec	KG	Enc	Dec
	Reference			*AVX2*		
Kyber-512	128	158	187	27	39	29
Kyber-768	209	255	286	44	65	44
Kyber-1024	321	369	414	60	79	63
SMAUG-T-128	110	100	136	38	23	35
SMAUG-T-192	219	204	253	57	46	61
SMAUG-T-256	357	334	414	77	65	86

highlighting its advantage in ephemeral key operations (*e.g.*, 23k cycles for SMAUG-T-128 vs. 39k cycles for Kyber-512). In contrast, Kyber achieves lower costs for both KeyGen and Decaps (*e.g.*, 27k and 29k cycles for KG and Dec in Kyber-512 vs. 38k and 35k cycles in SMAUG-T-128).

In [9], the performance of Kyber and NTRU+KEM was implemented on a single core of an Intel(R) Core(TM) i7-8700K (Coffee Lake) processor at 3.70 GHz with 16 GB of RAM, compiled with gcc version 11.4.0. Identical settings were applied to both schemes to ensure fairness in comparison.

Table V presents the benchmark results of Kyber and NTRU+KEM under reference and AVX2-optimized implementations across different parameter sets. Execution time was measured as the average cycle count over 100,000 executions for each algorithm [9].

TABLE V: Performance (in CPU kilocycles) of Kyber and NTRU+KEM [9]

Scheme	KG	Enc	Dec	KG	Enc	Dec
	Reference			*AVX2*		
Kyber-512	116	137	158	36	39	24
Kyber-768	182	202	230	51	55	37
Kyber-1024	270	321	359	65	73	52
NTRU+KEM768	192	101	121	26	27	16
NTRU+KEM864	238	123	148	28	30	19
NTRU+KEM1152	370	162	196	41	39	26

In the reference implementation, Kyber exhibits lower KeyGen costs at equivalent security levels (*e.g.*, 116k cycles for Kyber-512 vs. 192k cycles for NTRU+KEM768), whereas NTRU+KEM achieves lower Encaps and Decaps costs across all parameter sets. For instance, NTRU+KEM768 requires only 101k cycles for Encaps compared to 137k cycles for Kyber-512, and 121k cycles for Decaps compared to 158k cycles for Kyber-512. In the optimized AVX2 setting, both schemes demonstrate substantial performance improvements, with reductions by factors of 3–6. Notably, NTRU+KEM consistently outperforms Kyber, achieving the lowest cycle counts across all operations; for example, NTRU+KEM1152 requires 41k/39k/26k cycles for KG/Enc/Dec, compared with 65k/73k/52k cycles for Kyber-1024.

B. Lattice-based DSs

A DS consists of three main algorithms: (1) a probabilistic key generation algorithm, denoted by KeyGen; (2) a probabilistic signing algorithm, denoted by Sign; and (3) a deterministic verification algorithm, denoted by Verify, together with a collection of parameter sets.

ML-DSA (formerly Dilithium) provides three parameter sets in the form ML-DSA-kl, where (k, l) are the dimensions of the matrix $\mathbf{A}$, a part of the public key [3]. The parameter set ML-DSA-44 is claimed to be in security strength category 2 which generates a private key (SK) of 2,560 bytes, a public key (PK) of 1,312 bytes and a signature (Sig) of 2,420 bytes. ML-DSA-65 is claimed to be in category 3, which generates a SK of 4,032 bytes, a PK of 1,952 bytes and a Sig of 3,309 bytes. ML-DSA-87 is claimed to be in category 5 which generates a SK of 4,896 bytes, a PK of 2,592 bytes and a Sig of 4,627 bytes.

FALCON provides two parameter sets. FALCON-512 has an equivalent security to category 1, which generates a SK of 1,281 bytes, a PK of 897 bytes and a Sig of 666 bytes. FALCON-1024 has an equivalent security to category 5, gives a SK of 2,305 bytes, a PK of 1,793 bytes and a Sig of 1,280 bytes [4]. For comparison, FALCON-512 is roughly equivalent, in classical security terms, to RSA-2048, whose signatures and public keys are each 256 bytes in size [4].

HAETAE is specified with three parameter sets, each corresponding to a distinct security level. HAETAE-120 parameter set is assigned to security strength category 2 and generates a SK of 1,408 bytes, a PK of 992 bytes, and a Sig of 1,474 bytes. HAETAE-180 parameter set corresponds to security strength category 3 and generates a SK of 2,112 bytes, a PK of 1,472 bytes, and a Sig of 2,349 bytes. Finally, HAETAE-260 parameter set is associated with security strength category 5 and generates a SK of 2,752 bytes, a PK of 2,080 bytes, and a Sig of 2,948 bytes [11].

Table VI compares the security categories (Cat.) and the sizes of the SK, PK, and Sig for the lattice-based DSs, namely ML-DSA, FALCON, and HAETAE, from NIST and KpqC. Figure 2 illustrates these results in a bar graph, showing the SK, PK, and Sig sizes for the three schemes.

HAETAE achieves substantially smaller key and signature sizes than ML-DSA across all parameter sets. For instance, under the category 5 parameter set, HAETAE-260 requires 2,752 bytes, 2,080 bytes, and 2,948 bytes for the SK, PK, and Sig, respectively, whereas ML-DSA-87 requires 4,896 bytes, 2,592 bytes, and 4,627 bytes. FALCON provides the most compact signatures, with a Sig size of just 1,280 bytes at category 5, and also yields the smallest SK and PK sizes; 2,305 bytes, 1,793 bytes and 1,280 bytes, respectively, for FALCON-1024.

TABLE VI: Sizes (in bytes) of Keys and Signatures for Lattice-based DSs

Source	Scheme	Cat.	SK	PK	Sig
NIST	**ML-DSA-44**	2	2560	1312	2420
	ML-DSA-65	3	4032	1952	3309
	ML-DSA-87	5	4896	2592	4627
KpqC	**HAETAE-120**	2	1408	992	1474
	HAETAE-180	3	2112	1472	2349
	HAETAE-260	5	2752	2080	2948
NIST	**FALCON-512**	1	1281	897	666
	FALCON-1024	5	2305	1793	1280

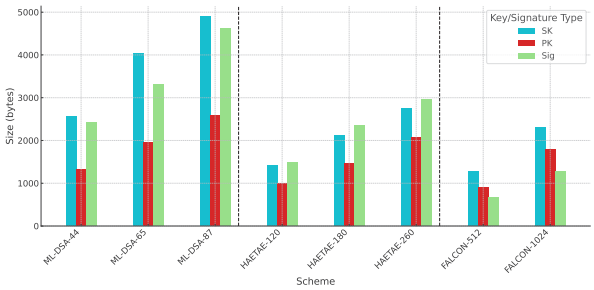

Fig. 2: Size comparison of keys and signatures for lattice-based DSs

In summary, ML-DSA incurs comparatively larger SK, PK, and Sig sizes; HAETAE offers moderate sizes across all parameter sets; and FALCON achieves the most compact overall footprint, particularly in signature size.

In [11], the performance of Dilithium, HAETAE, and FALCON were compared on a single core of an Intel(R) Core(TM) i7-10700K processor, with TurboBoost and hyperthreading disabled.

Table VII reports the average cycle counts of 1,000 executions for each parameter set of Dilithium, HAETAE, and FALCON for the reference implementations along with the AVX2-optimized implementations.

Across all parameter sets, the reference implementations incur substantially higher cycle counts than their AVX2-optimized counterparts, underscoring the efficiency gains enabled by vectorized instructions.

Among the schemes, Dilithium demonstrates balanced performance, with moderate costs for KeyGen, Sign, and Verify. In the AVX2 setting, its Verify operations remain particularly efficient. HAETAE, by contrast, exhibits significantly higher costs for both Sign and KeyGen in both reference and optimized implementations, though its Verify times remain on par with those of Dilithium. FALCON stands out for its extremely high KeyGen costs across both implementations. However, it compensates with the smallest Verify time in the reference setting and competitive Verify times under AVX2 optimization, aided by its significantly smaller signature sizes.

TABLE VII: Performance (in CPU kilocycles) of Dilithium, HAETAE, and FALCON [11]

Source	Scheme	KeyGen	Sign	Verify	KeyGen	Sign	Verify
		Reference			*AVX2*		
NIST	**Dilithium-2**	343,639	1,527,406	376,543	86,937	252,905	92,190
	Dilithium-3	630,607	2,603,237	612,852	146,688	402,012	148,883
	Dilithium-5	949,662	3,080,734	988,250	233,895	484,119	232,241
KpqC	**HAETAE-120**	1,827,567	9,458,682	376,631	733,350	1,916,063	108,890
	HAETAE-180	3,448,185	11,611,868	692,014	1,122,867	2,198,920	167,787
	HAETAE-260	4,088,383	17,229,712	896,622	1,202,911	3,013,239	207,229
NIST	**FALCON-512**	60,301,272	17,335,484	103,184	26,637,878	863,420	100,709
	FALCON-1024	178,516,059	38,009,559	224,840	78,797,658	1,740,520	228,326

V. Conclusion

In this paper, we have presented a comparative analysis of lattice-based PQC algorithms selected by NIST and the KpqC competition: ML-KEM (Kyber) from NIST, together with SMAUG-T and NTRU+KEM from KpqC for KEMs; and ML-DSA (Dilithium) and FALCON from NIST, along with HAETAE from KpqC for DSs. Our study has evaluated encapsulation and decapsulation key sizes and ciphertext sizes for KEMs, private and public key sizes and signature sizes for DSs, as well as performance benchmarks under both reference and AVX2-optimized implementations.

For KEMs, Kyber demonstrates a balanced design, offering moderate key and ciphertext sizes alongside moderate performance. SMAUG-T achieves a notable reduction in decapsulation key size and highly efficient encapsulation in the AVX2-optimized setting. NTRU+KEM, while requiring slightly larger parameters, delivers the best overall efficiency under optimization.

For DS schemes, Dilithium yields the largest key and signature sizes among the three but maintains balanced efficiency across KeyGen, Sign, and Verify. HAETAE achieves smaller keys and signatures than Dilithium, yet incurs significantly higher costs for KeyGen and Sign. FALCON produces the most compact signatures and the smallest key sizes, though this comes at the expense of substantially higher KeyGen overhead.

Overall, these results highlight the inherent trade-offs among security level, efficiency, compactness, and implementation performance in lattice-based cryptographic schemes. Such observations offer valuable guidance for evaluating their practicality and suitability for real-world deployment in the post-quantum era.

References

[1] R. Avanzi, J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, and D. Stehle, "CRYSTALS-Kyber algorithm specifications and supporting documentation," January 31, 2021, version 3.01.

[2] L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehlé, "CRYSTALS-Dilithium: Algorithm specifications and supporting documentation," February 8, 2021, version 3.1.

[3] "Module-Lattice-Based Key-Encapsulation Mechanism Standard," National Institute of Standards and Technology, Tech. Rep. Federal Information Processing Standards (FIPS) 204, August 13, 2024.

[4] P.-A. Fouque, J. Hoffstein, P. Kirchner, V. Lyubashevsky, T. Pornin, T. Prest, T. Ricosset, G. Seiler, W. Whyte, and Z. Zhang, "FALCON: Fast-Fourier lattice-based compact signatures over NTRU specification," October 1, 2020, v.1.2.

[5] J.-P. Aumasson, D. J. Bernstein, W. Beullens, C. Dobraunig, M. Eichlseder, S. Fluhrer, S.-L. Gazdag, A. Hulsing, P. Kampanakis, S. Kolbl, T. Lange, M. M. Lauridsen, F. Mendel, R. Niederhagen, C. Rechberger, J. Rijneveld, P. Schwabe, and B. Westerbaan, "SPHINCS+," June 10, 2022, v.3.1.

[6] "Module-Lattice-Based Key-Encapsulation Mechanism Standard," National Institute of Standards and Technology, Tech. Rep. Federal Information Processing Standards (FIPS) 205, August 13, 2024.

[7] P. Gaborit, C. Aguilar-Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, E. Persichetti, G. Zemorand, J. Bos, A. Dion, J. Lacan, J.-M. Robert, P. Veron, P. L. Barreto, S. Ghosh, S. Gueron, T. Guneysu, R. Misoczki, J. Richter-Brokmann, N. Sendrier, J.-P. Tillich, and V. Vasseur, "Hamming Quasi-Cyclic (HQC)," February 19, 2025, fourth round version.

[8] J. H. Cheon, H. Choe, J. Choi, D. Hong, J. Hong, C.-G. Jung, H. Kang, J. Lee, S. Lim, A. Park, S. Park, H. Seong, , and J. Shin, "SMAUG-T: the key exchange algorithm based on Module-LWE and Module-LWR," February 23, 2024, version 3.0.

[9] J. Kim and J. H. Park, "NTRU+: Compact construction of NTRU using simple encoding method," October 10, 2024, version 2.2.1.

[10] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, J. P. Buhler, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 267–288.

[11] J. H. Cheon, H. Choe, J. Devevey, T. Güneysu, D. Hong, M. Krausz, G. Land, M. Möller, D. Stehlé, and M. Yi, "HAETAE: Shorter lattice-based Fiat-Shamir signatures," https://hmchoe0528.github.io/assets/manuscripts/HAETAE_spec_24.07.04_v3.0.pdf, July 4, 2024, version 3.0.

[12] L. Ducas, A. Durmus, T. Lepoint, and V. Lyubashevsky, "Lattice signatures and bimodal Gaussians," in *Advances in Cryptology – CRYPTO 2013*, ser. LNCS, R. Canetti and J. Garay, Eds. Berlin, Heidelberg: Springer, 2013, pp. 40–56.

[13] S. Kim, J. Ha, M. Son, B. Lee, D. Moon, J. Lee, S. Lee, J. Kwon, J. Cho, H. Yoon, and J. Lee, "The AIMer signature scheme," https://aimer-signature.org/docs/AIMer-specification-v2.1.pdf, July 12, 2024, version 2.1.

[14] National Institute of Standards and Technology, "Submission requirements and evaluation criteria for the post-quantum cryptography standardization process," 2016.

[15] "Recommendations for key-encapsulation mechanisms," National Institute of Standards and Technology, Tech. Rep. NIST Special Publication (SP) 800-227, 2024.

[16] "Module-Lattice-Based Key-Encapsulation Mechanism Standard," National Institute of Standards and Technology, Tech. Rep. Federal Information Processing Standards (FIPS) 203, August 13, 2024.