The Kyber encryption algorithm and its implementation in .NET

Danylo S. Zolotopupov, Lesia V. Bulatetska and Vitalii V. Bulatetskyi

Lesya Ukrainka Volyn National University, 13 Voli Ave., Lutsk, 43025, Ukraine

Abstract

This paper examines modern cryptographic algorithms and their application for ensuring secure message exchange. Special attention is paid to the Kyber algorithm as one of the promising post-quantum cryptographic methods that guarantees reliable protection of information from quantum computer attacks. The Kyber algorithm is based on the hardness of the Module-Learning With Errors (MLWE) problem, which belongs to the mathematical class of lattice-based problems. The primary objective of this study is the practical implementation of the post-quantum Kyber algorithm through the development of a custom cryptographic library on the .NET platform. To achieve this, the theoretical foundations of Kyber were analyzed and implemented in code. The implementation successfully reproduces the full key exchange cycle, comprising key generation, encapsulation, and decapsulation. Following functional verification, performance benchmarks were conducted across various cryptographic parameter sets, covering both single- and multi-threaded scenarios on the .NET platform. The algorithm's viability was validated on diverse target platforms, including mobile devices, using a cross-platform .NET MAUI application. Prospects for further research include the development of practical applications, particularly the creation of a program for secure message exchange. The use of Kyber in modern technologies will contribute to ensuring the long-term protection of information in the context of developing quantum computing and growing cybersecurity threats.

Keywords

Kyber, post-quantum cryptography, encryption, key exchange, data security, .NET

1. Introduction

Encryption is the process of transforming data, which allows hiding information and ensuring its confidentiality. In modern information systems that exchange data over open communication channels, two types of encryption algorithms are typically used: symmetric and asymmetric.

Symmetric encryption is used to protect the content of messages. Its feature is the necessity of a shared secret key, which is possessed by both parties of the exchange. To guarantee security, this key must remain inaccessible to third parties.

Asymmetric encryption, in turn, ensures the secure exchange of a symmetric algorithm's shared key over open channels. It is based on the use of two interconnected keys: a public and a private one. The public key is used to encrypt the message, while it can only be decrypted with the help of the private key. Thanks to the mathematical properties of the algorithm, obtaining the private key from the public key is considered practically impossible.

The growth of computational power, particularly the development of quantum computers, threatens most traditional encryption algorithms. In 2016, the National Institute of Standards and Technology of the USA (NIST) announced a competition for the development of encryption algorithms resistant to quantum computing attacks [1]. The competition aimed to create standards for publicly available algorithms for digital signatures, public-key encryption, and key exchange, capable of protecting confidential information even after the emergence of sufficiently powerful quantum computers. In the English-speaking world, the emergence of computers powerful enough to break popular asymmetric encryption algorithms is called "Q-day". Despite the fact that modern quantum computers are considered

CS&SE@SW 2025: 8th Workshop for Young Scientists in Computer Science & Software Engineering, November 21, 2025, Kryvyi Rih, Ukraine
https://doi.org/10.55056/cssesw2025/paper31
Zolotopupov.Danylo2022@vnu.edu.ua,bloodmagv@gmail.com (D. S. Zolotopupov); Bulatetska.Lesya@vnu.edu.ua (L. V. Bulatetska); bulatetsky.vitaly@vnu.edu.ua (V. V. Bulatetskyi)
0009-0005-5347-1070 (D. S. Zolotopupov); 0000-0002-7202-826X (L. V. Bulatetska); 0000-0002-9883-4550 (V. V. Bulatetskyi)

very far from being able to solve real-world problems, malicious actors and various state agencies are already accumulating encrypted information today with the goal of decrypting it in the future when they have such a technological capability; therefore, it is necessary to defend against quantum attacks today. In August 2024, NIST published FIPS 203 [2], a document that describes in detail the implementation of the Kyber algorithm, including certain optimizations.

Post-quantum cryptographic algorithms, especially Kyber, require careful analysis and adaptation for reliable use in modern software platforms. To study Kyber's performance, reliability, and behavior within the .NET environment, it was necessary to create a custom implementation that fully follows the NIST specification [2].

2. Literature review

2.1. The AES symmetric encryption algorithm

The AES (Advanced Encryption Standard) algorithm is a standard for symmetric block encryption, approved by the National Institute of Standards and Technology of the USA in 2001 [3]. The basis of AES is the Rijndael method, which uses substitution, permutation, and mathematical operations to transform data into an encrypted form [4]. The official standard FIPS PUB 197 [3] describes in detail the internal structure of AES and its advantages.

AES works with a fixed data block size (128 bits) and supports key lengths of 128, 192, or 256 bits. Encryption occurs in several rounds (10, 12, or 14 depending on the key length), each of which performs the operations SubBytes (byte substitution), ShiftRows (row shifting), MixColumns (column mixing), and AddRoundKey (round key addition) [3].

AES is widely used in modern encryption systems. Its high performance makes its application possible even on devices with limited resources. But the most important fact is that AES is a globally recognized standard for most government and commercial systems and is actively used in the banking sector. Due to its popularity, it is constantly under the scrutiny of leading experts, hackers, and scientists from all countries, yet it is still considered reliable. The main limitation of AES is the need for a secure key exchange between parties before encryption can begin. This is due to the symmetric nature of the algorithm, where the same key is used for encryption and decryption. If this key is compromised, an attacker will gain access to all the encrypted information.

Most research confirms the high cryptographic strength of AES [4]. Biryukov et al. [5], Chen et al. [6] noted that AES-128, AES-192, and AES-256 remain resistant to brute-force, differential cryptanalysis, and side-channel attacks. Intel, in its technical documentation [7], also emphasizes the advantages of AES and introduces six new AES-NI instructions that significantly increase the performance and security of encryption. These instructions became part of the Intel IA-32 architecture starting in January 2010.

In modern publications, great attention is paid to the resistance of AES to side-channel attacks, optimization for IoT devices, and analysis of its effectiveness in the context of quantum threats. For example, Salman et al. [8], Al-Mashhadani and Shujaa [9] investigate the implementation of AES on devices with limited resources. The analysis of AES in the context of post-quantum cryptography is also relevant. FMTAD [10] presented an improved version of the Advanced Encryption Standard (AES) that uses quantum technology to enhance protection.

Overall, AES continues to remain a reliable, efficient, and versatile symmetric encryption algorithm, suitable for both traditional and constrained environments.

2.2. The RSA asymmetric encryption algorithm

The RSA algorithm is based on the difficulty of factoring the product of two large prime numbers and uses modular arithmetic [1]. Before using the algorithm, a pair of keys is formed: public and private. The party that generates the keys chooses two large prime numbers p and q, which are kept secret,

and calculates their product $n = pq$. Next, a public exponent e is chosen, which is copri me with $(p-1)(q-1)$, and a number d is found such that $ed \equiv 1 \pmod{(p-1)(q-1)}$.

At the encryption stage, the message M is converted into a numerical form and is encrypted by calculating:

$$C = M^e \bmod n,$$

where C is the ciphertext.

Decryption occurs by calculating:

$$M = C^d \bmod n,$$

where M is the decrypted message in numerical form.

The difficulty of factoring n ensures security, because to decrypt, one needs to know p and q, and determining them is a difficult task for large n. The typical length of the public key is from 1024 to 2048 bits.

The RSA algorithm, like AES, is very popular and widespread. There is a large number of its implementations from reputable developers in all popular programming languages. It is well researched and predictable in its functioning. RSA has been actively used since the early 2000s and is studied by specialists from all over the world. As of 2024, breaking an 829-bit key has been confirmed; breaking a 512-bit key is no longer difficult for a modern home PC.

In 1994, Peter Shor proved that a quantum computer of sufficient power is capable of finding the private key (factoring a large number into its prime components) of the RSA algorithm quite quickly [11]. Experts predict that quantum computers capable of breaking RSA-2048 may appear within the next 10–20 years, although some sources indicate this possibility as early as 2027 [12].

Thus, while RSA currently remains secure, organizations should already be planning the transition to post-quantum algorithms to guarantee the long-term security of their data.

2.3. The Kyber asymmetric encryption algorithm

The Kyber algorithm passed all selection stages and was approved by NIST as the primary algorithm for secret key exchange, with a planned worldwide transition to it by 2030. Some encryption protocols began using it even before the official publication; for example, Signal was already using it as of December 2023.

The Kyber algorithm is based on the problems of decision module learning with errors (D-MLWE, Decision Module Learning With Errors), which can be reduced to the mathematical class of lattice-based problems. These problems are considered difficult for both quantum and classical computing systems.

The Kyber algorithm is new and, as a consequence, a relatively weakly researched algorithm. It has not yet stood the test of time. Also, this algorithm was designed for a single purpose: the secure exchange of a symmetric encryption key, which is a set of random numbers. It is not intended for exchanging other kinds of data that may contain some patterns, for example, plain text.

Cryptographically secure random numbers are an essential component of encryption algorithms. They are used to generate keys, initialization vectors (IVs), salts, and other elements that ensure the security of a system. The process of generating these numbers must meet criteria such as unpredictability and uniqueness. That is, random numbers must be unpredictable even with access to part of the source data and must not be repeated within the same session to avoid so called replay attacks.

Bos et al. [13] presents Kyber as part of the CRYSTALS suite, submitted to NIST for the standardization of post-quantum cryptographic algorithms. Cisneros Laule et al. [14] introduced RKyber, a variant that targets the Learning with Rounding (LWR) problem, simplifying computations by using deterministic errors rather than random noise. Comparative testing, conducted 1000 times, showed that RKyber provides higher performance, albeit with some reduction in security level.

An in-depth analysis of Kyber's decoding noise is presented in [15]. This work shows that it can be confined to a sphere, and the decoding problem itself is reduced to the problem of sphere packing in a hypercube, which opens the way to improving the algorithm's efficiency.

Open-source software libraries play a crucial role in the practical implementation of Kyber. The repository by Bos et al. [16] contains the official reference implementation of the Kyber key encapsulation mechanism, as well as an optimized version for x86 processors with AVX2 instruction support. Segatz and Hafiz [17] describes the implementation of Kyber on the ESP32 microcontroller, popular in IoT applications. Thanks to optimization for the dual-core architecture, an encapsulation speedup of up to 1.84 times was achieved. Seo and Kim [18] examines the implementation of the Kyber algorithm in WebAssembly, which provides better performance compared to JavaScript and allows for its portable use in various environments. Huang et al. [19] focuses on optimizing arithmetic operations by modifying the Plantard method, which allows for the efficient execution of the algorithm on 32-bit IoT platforms (ARM Cortex-M3, RISC-V) without the need for SIMD extensions.

Hardware implementations have also not been overlooked. Jati et al. [20] presents a configurable and side-channel resistant hardware implementation of the Kyber algorithm, allowing for its efficient use in hardware solutions. In [21], the first formally verified implementations of Kyber using the EasyCrypt tool are presented, ensuring a high level of confidence in the correctness of the implementation. Tran-Hoang and Nakajo [22] demonstrates a pure hardware architecture for Kyber, aiming to provide realistic and comparable evaluations on hardware platforms as part of the NIST standardization process.

3. The Kyber algorithm structure

Regarding software ecosystems, an official implementation of Kyber for the .NET environment was introduced on November 11, 2025. However, its adoption remains constrained by strict dependencies on specific platform versions and operating system libraries, complicating its integration into complex, cross-platform .NET projects.

The goal of the work was to develop a custom, fully managed implementation of the Kyber post-quantum encryption algorithm in accordance with the NIST [3] specification. Unlike the official wrapper, this solution aims to provide universal portability across all .NET-supported platforms without reliance on OS-specific cryptographic providers.

Let us consider a simplified cryptographic scheme of the Kyber algorithm [23]. For this, we review the main definitions.

$R_q = \mathbb{Z}_q[x]/(x^n + 1)$ — the ring of polynomials of degree $n - 1$ with coefficients modulo q.

S_η — the set of "small" polynomials in R_q, whose coefficients belong to the interval $[-\eta, \eta]$ (mod q).

R_q^k — the set of column vectors with k elements belonging to R_q.

S_η^k — the set of column vectors with k elements belonging to S_η.

The plaintext space is defined as $\{0,1\}^n$, where any text message $m \in \{0,1\}^n$ is associated with a polynomial in R_q whose coefficients are 0 or 1 (figure 1).[23] For example, if $n = 5$ and $m = 10110$, then $m \leftrightarrow m(x) = 1 + x^2 + x^3$.

$\lfloor x \rfloor$ is the greatest integer less than or equal to x, and $\lceil x \rceil$ is the smallest integer greater than or equal to x.

For example: $\lfloor 3.7 \rfloor = 3$, $\lfloor -3.7 \rfloor = -4$, $\lceil 3.7 \rceil = 4$, $\lceil -3.7 \rceil = -3$.

The notation $\lfloor x \rceil$ denotes the integer nearest to x according to the rules of algebraic rounding, that is: $\lfloor 3.7 \rceil = 4$, $\lfloor -3.7 \rceil = -4$, $\lfloor 3.5 \rceil = 4$, $\lfloor -3.5 \rceil = -3$, $\lfloor -3.1 \rceil = -3$. [23]

3.1. Rounding

Let q be an odd prime number, and $x \in [0, q-1]$. Also, let $x' = x \bmod q$, so that

$$x' \in \left[-\frac{q-1}{2}, \frac{q-1}{2} \right].$$

Then

$$\mathrm{Round}_q(x) = \begin{cases} 0, & \text{if } -\frac{q}{4} < x' < \frac{q}{4}, \\ 1, & \text{otherwise.} \end{cases}$$

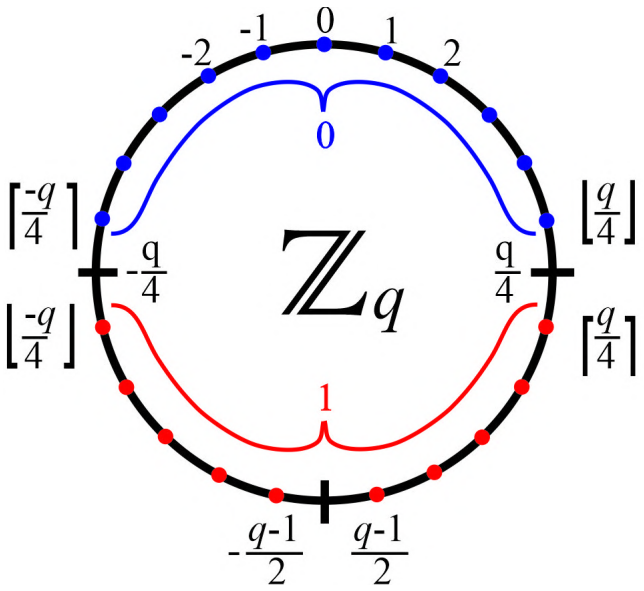

Figure 1: Encoding bits in $\mathbb{Z}_q$ [23].

The Round_q operation can be extended to polynomials in R_q by applying it to each coefficient of the polynomial.[23]

3.2. Domain parameters and key generation

For concreteness we consider ML-KEM-1024: $q = 3329$, $n = 256$, $k = 4$, $\eta_1 = 2$, $\eta_2 = 2$, $d_u = 11$, $d_v = 5$. To generate a key pair, User1 performs the following steps:

1. Select (randomly) $A \in R_q^{k \times k}$, $\quad s \in S_{\eta_1}^k$, $\quad$ and $\quad e \in S_{\eta_2}^k$.
2. Compute $t = As + e$.
3. User1's encryption (public) key is (A, t); their decryption (private) key is s.

To encrypt a message $m \in \{0,1\}^n$ for User1, User2 performs the following steps:

1. Obtain User1's authentic public key (A, t).
2. Select (randomly) $r \in S_{\eta_1}^k$, $\quad e_1 \in S_{\eta_2}^k$, $\quad$ and $\quad e_2 \in S_{\eta_2}^k$.
3. Compute $u = A^T r + e_1, \quad v = t^T r + e_2 + \lceil \frac{q}{2} \rfloor m$.
4. Output the ciphertext $c = (u, v) \in R_q^k \times R_q$.

To decrypt the ciphertext c, User1 computes $m = \mathrm{Round}_q(v - s^T u)$,. User1 uses their private key s. [23]

3.3. Implementation of a cryptographic library for Kyber in .NET

Figure 2 shows the structure of the Cryptography project, which contains the implementation code for the Kyber algorithm and supporting structures/algorithms. It is a shared project, meaning the project's code exists as text and cannot be compiled or executed independently. Other projects can reference it, and during their build process, this code is included and compiled.

The ArrayExtensions class is used to define extension methods for the byte[] (byte array) type. The Kyber algorithm involves concatenating byte arrays with individual bytes or other byte arrays. The ModuloRing class represents an integer in the algebraic ring of residues modulo $\mathbb{Z}_m$, defined by its value and the modulus m. The PolyRing class represents the coefficients of a polynomial of degree n, where each coefficient belongs to $\mathbb{Z}_m$. It is defined by an array of coefficient values of size n and

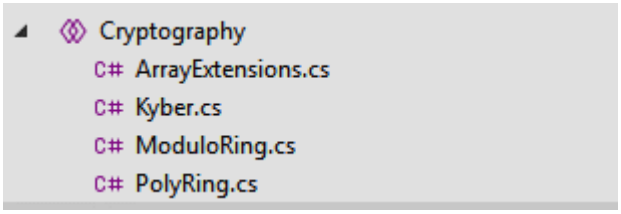

Figure 2: Cryptography project structure.

the modulus m. Both classes overload operators for assignment, addition, multiplication, and equality checking.

The ModuloRing and PolyRing classes help prevent errors in the Kyber algorithm implementation, where modular arithmetic is extensively used. They also contribute in code's overall readability and maintainability.

The Kyber class implements the encryption algorithm, consisting of three stages: key generation, encapsulation, and decapsulation. According to the FIPS 203 standard, the algorithm supports three security levels, each defining a set of parameters. When creating an instance of the class, the security level is passed to the constructor and is the only internal state of the object. All public methods of the class are pure functions, they depend solely on input data and do not modify the object's internal state. Such approach ensures thread safety, predictable execution and ease of use for the developers utilizing this library.

The Kyber class includes a nested Consts class defining algorithm-level constants and an abstract EncryptionParams class with three derived classes: `ML_KEM_512`, `ML_KEM_768`, and `ML_KEM_1024`. These classes contain five properties: k, Eta1, Eta2, Du, Dv – as specified in the algorithm's specification [2], determining the encryption's security level. These parameters are stored in the Kyber class's single property, Params, of type EncryptionParams.

The FIPS 203 standard incorporates several algorithmic optimizations designed to enhance computational efficiency. To accelerate polynomial multiplication, the implementation leverages the Number-Theoretic Transform (NTT). Specifically, two immutable arrays, zetaPowBitRev7 and zetaPow2Bitrev7P1, are populated within the static constructor to cache 128 precomputed integers essential for both forward and inverse NTT operations. Furthermore, ciphertext efficiency is optimized through compression, achieved by discarding the low-order bits of polynomial coefficients. Although this process introduces additional noise during the decapsulation stage, it remains strictly within the algorithm's error tolerance limits. Finally, to minimize bandwidth usage, the transmission of the substantial Matrix A is replaced by a 256-bit seed, allowing both parties to deterministically regenerate the matrix locally. The main public methods of the Kyber class (figure 3) and the cryptographic scheme of the algorithm (figure 4) are discussed below.

Suppose User1 (Alice) and User2 (Bib) aim to establish a shared key for symmetric encryption using the Kyber algorithm. User1 initiates the process by calling the KeyGen() method, generating two keys: an encapsulation key and a decapsulation key. The encapsulation key is sent to User2 via an open channel, while User1 retains the decapsulation key. Upon receiving the encapsulation key, User2 calls the Encapsulate() method, passing the key as input. This method returns a generated shared key and a ciphertext, a data set containing the encrypted shared key, which is sent to User1 via an open channel. Upon receiving the ciphertext, User1 calls the Decapsulate() method, providing the ciphertext and the decapsulation key generated earlier. This method returns the shared key, identical to the one generated by User2. For a code snippet demonstrating this exchange refer to figure 5.

3.4. Testing of the algorithm

After implementing the Kyber encryption algorithm, the entire process of shared key exchange was reproduced: generating a public-private key pair, generating a shared key and ciphertext using the public key, and deriving the shared key from the ciphertext using the private key. At this stage, testing

Figure 3: Signatures of the Kyber class public methods.

was conducted on a single device to verify the correctness of the Kyber implementation. After the verification and debugging phase, performance measurements were conducted (table 1). Tests were performed in a release build for each set of cryptographic parameters for both single- and multi-threaded scenarios. Multi-threading was achieved via the Parallel.For method of the .NET's Standard Class Library. For a smartphone, the test was run via a MAUI application.

Table 1
Comparative performance analysis of ML-KEM (Kyber) on various hardware platforms.

Test	System configuration (avg. decaps/sec)				
	Desktop Intel Core i5-12600K	**Desktop Intel Core i9-9900K**	**HP Envy x360 15-ey0xxxx AMD Ryzen 5825U**	**Lenovo ThinkPad T450 Intel Core i5-5300U**	**Sony Xperia 1 IV Snapdragon 8 Gen 1**
ML_KEM_512. Single thread	1528	871	799	559	289
ML_KEM_512. Multi thread	8627	5727	3413	1034	671
ML_KEM_768. Single thread	856	474	403	299	133
ML_KEM_768. Multi thread	4703	3150	2017	500	243
ML_KEM_1024. Single thread	528	294	244	184	82
ML_KEM_1024. Multi thread	2893	1925	1240	334	138

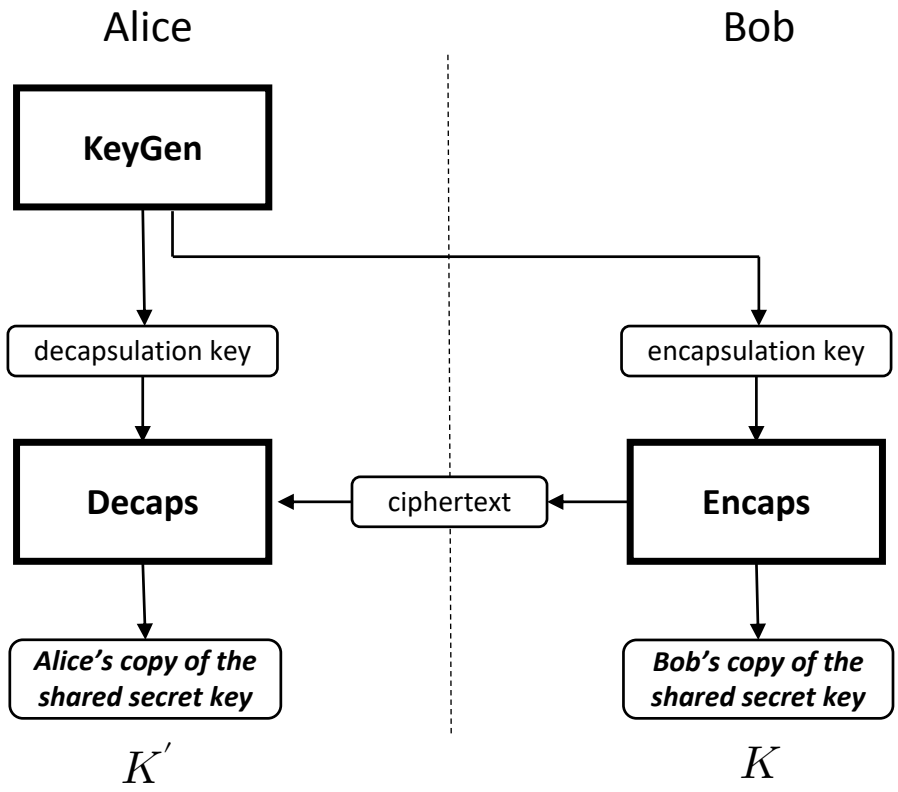

Figure 4: Cryptographic scheme of the Kyber algorithm [2].

```
// Create a Kyber instance with desired encryption level, done by both parties
Kyber kyber = new(Kyber.EncryptionLevel.ML_KEM_1024);

// Key pair generation (by User1)
(byte[] encapsKey, byte[] decapsKey) = kyber.KeyGen();

// Secret shared key encapsulation (by User2, they get encapsulation key from User1)
(byte[] secretB, byte[] cipherText) = kyber.Encapsulate(encapsKey);

// Secret shared key decapsulation (by User1, they get cipherText back from User2)
byte[] secretA = kyber.Decapsulate(decapsKey, cipherText);

// Now secretA and secretB are identical arrays of 32 bytes
```

Figure 5: Library usage example.

4. Conclusions and future research prospects

This work examined modern cryptographic algorithms and their application for secure message exchange. Within the scope of the project, an implementation of the Kyber algorithm for the .NET platform was developed as a library, following its specification. Kyber is an asymmetric algorithm used for secure key exchange, similar to classical RSA or Diffie-Hellman, but offers greater resistance to cryptanalytic attacks. Its main advantages include high efficiency, fast performance, relatively short keys, and the ability to integrate into modern encryption systems. Kyber is actively considered one of the primary candidates for future cryptographic standards, applicable in secure internet connections (TLS, VPN), cloud data encryption, and governmental and military communication systems. Due to its properties, this algorithm is among the most promising methods for information protection in the upcoming

quantum era. Future research prospects include integrating the algorithm into existing cryptographic libraries and developing practical applications, such as a secure messaging application. Additionally, it is worth investigating the algorithm's resilience to various types of attacks and its potential combination with other cryptographic methods to create hybrid security systems.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT (GPT-5.1) in order to: Grammar and spelling check. After using this tool/service, the authors reviewed and edited the content as necessary and take full responsibility for the publication's content.

References

[1] National Institute of Standards and Technology (NIST), Post-Quantum Cryptography, NIST Computer Security Resource Center (CSRC), 2025. URL: https://csrc.nist.gov/pqc-standardization.

[2] National Institute of Standards and Technology (NIST), Module-Lattice-Based Key-Encapsulation Mechanism Standard, Federal Information Processing Standards Publication 203, National Institute of Standards and Technology (NIST), 2024. doi:10.6028/NIST.FIPS.203.

[3] National Institute of Standards and Technology (NIST), Advanced Encryption Standard (AES), Federal Information Processing Standards Publication 197, National Institute of Standards and Technology (NIST), 2001. doi:10.6028/NIST.FIPS.197-upd1.

[4] A. Biryukov, D. Khovratovich, Related-Key Cryptanalysis of the Full AES-192 and AES-256, in: M. Matsui (Ed.), Advances in Cryptology – ASIACRYPT 2009, volume 591 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 1–18. doi:10.1007/978-3-642-10366-7_1.

[5] A. Biryukov, D. Khovratovich, I. Nikolić, Distinguisher and Related-Key Attack on the Full AES-256, in: S. Halevi (Ed.), Advances in Cryptology - CRYPTO 2009, volume 5677 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 231–249. doi:10.1007/978-3-642-03356-8_14.

[6] Y. Chen, Y. Wang, Y. Zhou, Differential Power Analysis of AES Implementations, IEEE Transactions on Computers 61 (2012) 678–688. doi:10.1109/TC.2011.68.

[7] S. Gueron, Intel® Advanced Encryption Standard (AES) Instructions Set, White Paper, Intel Corporation, 2010. URL: https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf.

[8] R. S. Salman, A. K. Farhan, A. Shakir, Lightweight Modifications in the Advanced Encryption Standard (AES) for IoT Applications: A Comparative Survey, in: 2022 International Conference on Computer Science and Software Engineering (CSASE), 2022, pp. 325–330. doi:10.1109/CSASE51777.2022.9759828.

[9] M. Al-Mashhadani, M. Shujaa, IoT Security Using AES Encryption Technology based ESP32 Platform, The International Arab Journal of Information Technology (IAJIT) 19 (1922) 214–223. doi:10.34028/iajit/19/2/8.

[10] FMTAD, Quantum Threats to Encryption: RSA, AES & ECC Defense, Freemindtronic, 2024. URL: https://freemindtronic.com/quantum-computing-encryption-threats-risks/.

[11] I. Chadha, Quantum Cryptography: A Review of the Literature, NHSJS, 2025. URL: https://web.archive.org/web/20260609021823/https://nhsjs.com/2025/quantum-cryptography-a-review-of-the-literature/.

[12] D. Oliver, Post-quantum security: a future threat that needs addressing now, BIForesight, 2024. URL: https://biforesight.com/quantum/post-quantum-security-a-future-threat.

[13] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. M. Schanck, P. Schwabe, G. Seiler, D. Stehle, CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM, 2018. doi:10.1109/EuroSP.2018.00032.

[14] M. S. Cisneros Laule, J. E. Olazabal Silva, H. Nina Hanco, Lattice-Based Cryptography: Development and Analysis of a New Variant of the Crystals-Kyber Algorithm, Interfases (2024) 163–182. doi:10.26439/interfases2024.n020.7383.

[15] S. Liu, A. Sakzad, Lattice codes for CRYSTALS-Kyber, Designs, Codes and Cryptography 93 (2025) 3181–3205. doi:10.1007/S10623-025-01640-W.

[16] J. Bos, L. Ducas, E. Kiltz, T. Lepoint, V. Lyubashevsky, J. Schanck, P. Schwabe, G. Seiler, D. Stehlé, CRYSTALS - Kyber: Algorithm Specifications and Supporting Documentation, 2021. URL: https://github.com/pq-crystals/kyber.

[17] F. Segatz, M. I. A. Hafiz, Efficient Implementation of CRYSTALS-KYBER Key Encapsulation Mechanism on ESP32, CoRR abs/2503.10207 (2025). doi:10.48550/ARXIV.2503.10207. arXiv:2503.10207.

[18] S. C. Seo, H. Kim, Portable and Efficient Implementation of CRYSTALS-Kyber Based on WebAssembly, Computer Systems Science and Engineering 46 (2023) 2091–2107. doi:10.32604/csse.2023.035064.

[19] J. Huang, H. Zhao, J. Zhang, W. Dai, L. Zhou, R. C. C. Cheung, Ç. K. Koç, D. Chen, Yet Another Improvement of Plantard Arithmetic for Faster Kyber on Low-End 32-bit IoT Devices, IEEE Transactions on Information Forensics and Security 19 (2024) 3800–3813. doi:10.1109/TIFS.2024.3371369.

[20] A. Jati, N. Gupta, A. Chattopadhyay, S. K. Sanadhya, A Configurable CRYSTALS-Kyber Hardware Implementation with Side-Channel Protection, ACM Transactions on Embedded Computing Systems 23 (2024) 33. doi:10.1145/3587037.

[21] J. B. Almeida, S. Arranz Olmos, M. Barbosa, G. Barthe, F. Dupressoir, B. Grégoire, V. Laporte, J.-C. Léchenet, C. Low, T. Oliveira, H. Pacheco, M. Quaresma, P. Schwabe, P.-Y. Strub, Formally Verifying Kyber, in: L. Reyzin, D. Stebila (Eds.), Advances in Cryptology – CRYPTO 2024, volume 14921 of *Lecture Notes in Computer Science*, Springer Nature Switzerland, Cham, 2024, pp. 384–421. doi:10.1007/978-3-031-68379-4_12.

[22] K. Tran-Hoang, H. Nakajo, Kyress: a secure, scalable, and resource-efficient CRYSTALS-Kyber cryptosystem for low-cost embedded devices, The Journal of Supercomputing 82 (2026) 101. doi:10.1007/s11227-025-08178-7.

[23] A. Menezes, Kyber & Dilithium – Cryptography 101 with Alfred Menezes, 2024. URL: https://cryptography101.ca/kyber-dilithium/.