Implementing post-quantum KEMs: Practical challenges and solutions

Pavlo Vorobets[1,†], Oleksandr Vakhula[1,†], Andrii Horpenyuk[1,†] and Nataliia Korshun[2,*,†]

[1] *Lviv Polytechnic National University, 12 Stepana Bandery str., 79013 Lviv, Ukraine*
[2] *Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudryavska str., 04053 Kyiv Ukraine*

Abstract

The paper provides an overview and analysis of the current state, problems, and prospects of post-quantum key encapsulation mechanisms. The essential cryptographic building blocks for implementing secure communication protocols are key-encapsulation mechanisms. KEMs enable two parties to securely establish a shared secret key over an insecure channel. This shared key can then be used for symmetric encryption of messages, ensuring confidentiality and integrity of the exchanged data. The National Institute of Standards and Technology (NIST) is actively working on standardizing post-quantum cryptography including KEMs. After the third round of the NIST PQC Standardization Process, NIST has identified the CRYSTALS-KYBER KEM algorithm for standardization. The four algorithms selected for a fourth round are BIKE, Classic McEliece, HQC, and SIKE. In this paper, we explore all these algorithms.

Keywords

post-quantum cryptography, KEM, standardization process, NIST, CRYSTALS-KYBER, BIKE, Classic McEliece, HQC, SIKE

1. Introduction

As quantum computing advances, the implementation of post-quantum Key Encapsulation Mechanisms (KEMs) becomes critical for securing data against future threats.

Quantum computing represents a paradigm shift in computational power, capable of solving complex mathematical problems much faster than classical computers. This capability poses a significant threat to current cryptographic systems, particularly those relying on public-key algorithms like RSA and ECC, which are vulnerable to quantum attacks. The most widely used key exchange algorithms today are based on hard mathematical problems, such as integer factorization and the discrete logarithm problem. However, these problems can be efficiently solved by a quantum computer [1].

Key Encapsulation Mechanisms are cryptographic protocols designed to securely exchange symmetric keys over insecure channels. They are a cornerstone of many secure communication systems, enabling the safe transmission of encryption keys that can then be used for symmetric encryption. In the quantum era, the security of these key exchanges is paramount.

To counteract the threat posed by quantum computing, researchers are developing post-quantum cryptographic algorithms, including post-quantum KEMs. These mechanisms are based on mathematical problems believed to be resistant to quantum attacks.

The transition to post-quantum cryptography, including KEMs, is a proactive measure to secure communications against future quantum threats. Standardization bodies like the National Institute of Standards and Technology (NIST) are actively working on evaluating and standardizing post-quantum cryptographic algorithms, including KEMs, to provide clear guidelines and frameworks for adoption.

As quantum computing advances, the cryptographic landscape must evolve to ensure the continued security of key exchanges and communications. Post-quantum Key Encapsulation Mechanisms are at the forefront of this evolution, offering new approaches to secure key exchange that are resistant to quantum attacks. Understanding and implementing these mechanisms will be crucial for protecting sensitive information in the quantum era, ensuring that cryptographic systems remain robust and secure against emerging threats.

2. Literature review and problem statement

Cryptography is an essential aspect of modern life, providing the necessary security and trust in our digital interactions, financial transactions, communication, and data privacy. Its widespread use ensures the confidentiality, integrity, and authenticity of information in various aspects of our daily lives.

CPITS-II 2024: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, October 26, 2024, Kyiv, Ukraine

* Corresponding author.
† These authors contributed equally.

pavlo.a.vorobets@lpnu.ua (P. Vorobets); oleksandr.p.vakhula@lpnu.ua (O. Vakhula); andrii.y.horpeniuk@lpnu.ua (A. Horpenyuk); n.korshun@kubg.edu.ua (N. Korshun)

0009-0007-3870-829X (P. Vorobets); 0009-0008-5367-3344 (O. Vakhula); 0000-0001-5821-2186 (A. Horpenyuk); 0000-0003-2908-970X (N. Korshun)

As quantum computing continues to advance, the threat it poses to classical cryptographic systems becomes more apparent. Quantum computers, once fully realized, will be able to efficiently break commonly used cryptographic schemes like RSA and ECC through Shor's algorithm, rendering traditional cryptographic infrastructure insecure. In response to this looming threat, post-quantum cryptography has emerged as a field focused on developing cryptographic algorithms that are resistant to quantum attacks. Key Encapsulation Mechanisms, which are used for secure key exchange, are a key focus in this field [2].

The National Institute of Standards and Technology (NIST) launched its Post-Quantum Cryptography Standardization project in 2017, aiming to identify and standardize quantum-resistant cryptographic algorithms. Many algorithms have been proposed, with a significant portion focusing on KEMs due to their critical role in secure communications [3].

To address this, post-quantum key encapsulation mechanisms are being developed to ensure the secure exchange of encryption keys, even in the presence of powerful quantum adversaries [4].

3. Introduction to key encapsulation mechanisms in post-quantum cryptography

A Key Encapsulation Mechanism (KEM) is a cryptographic protocol used to securely exchange encryption keys between two parties. The primary goal of KEMs is to generate and securely encapsulate a random key, which can then be used for further cryptographic operations, such as symmetric encryption.

A KEM typically consists of three main phases:

Key Generation: The sender generates a pair of keys (public and private keys). The public key is shared with the recipient.

Encapsulation: The sender uses the public key to generate a random secret and encapsulates it. The encapsulated secret is sent to the recipient.

Decapsulation: The recipient uses their private key to recover the original secret from the encapsulated data.

The most widely used key exchange algorithms today are based on hard mathematical problems, such as integer factorization and the discrete logarithm problem. But these problems can be efficiently solved by a quantum computer.

KEMs are generally designed to be non-interactive, meaning they only require a single communication round (i.e., one message from the sender to the recipient).

The main security goal of a KEM is to prevent attackers from gaining any useful information about the shared secret. These goals are formalized using security definitions, often based on the IND-CCA (Indistinguishability under Chosen Ciphertext Attack) security model [5].

A KEM can be seen as similar to a Public Key Encryption (PKE) scheme since both use a combination of public and private keys. In a PKE, one encrypts a message using the public key and decrypts using the private key. In a KEM, one uses the public key to create an "encapsulation"—giving a randomly chosen shared key—and one decrypts this "encapsulation" with the private key.

Public key encryption is often used to transmit symmetric encryption keys, which are then used to encrypt the originally intended plain-text content needing encryption protection. Symmetric keys are faster and stronger (for smaller key sizes) than asymmetric encryption, and so PKEs are often just used as a secure transport vehicle for the symmetric keys that do all the direct encryption work. PKEs have worked great for decades, but they have at least one big inherent flaw [6].

When the public key is longer than the content being encrypted (such as is usually the case with the symmetric key in key exchanges), it allows attackers a very easy way to derive the original private key. To prevent this scenario, when the message content to be encrypted (e.g., the symmetric key) is shorter than the asymmetric private key used to do the encryption, PKEs will usually add additional "padding" to the message to be encrypted (e.g., the symmetric key) to remove the vulnerability.

Key encapsulation methods, also known as key encapsulation schemes, are a type of asymmetric encryption technique designed to improve the secure transmission (or generation) of symmetric keys because they don't need random padding added to short messages to stay secure. Many postquantum cryptographic algorithms are especially conducive to creating KEMs and because postquantum algorithms often have even longer asymmetric keys, you will see many quantum-resistant teams offering KEMs instead of PKEs. Also, some post-quantum cryptographic algorithms will offer both PKE and KEM versions [7].

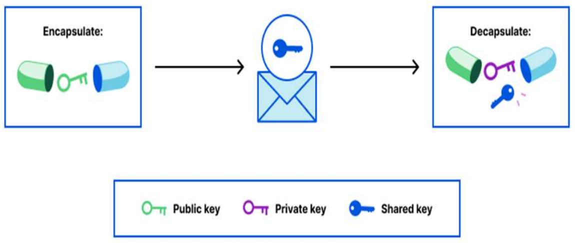

Figure 1: Key Encapsulation Mechanisms

3.1. Types of KEMs in post-quantum cryptography

In the post-quantum world, the security of KEMs must rely on quantum-resistant mathematical problems. Here are the key post-quantum techniques used for designing KEMs:

- Lattice-based cryptography—uses lattices and their associated mathematical properties to provide security. A lattice is a set of points in a multi-dimensional space that form a regular grid-like structure. Lattice-based cryptography is one of the most promising post-quantum techniques because it provides strong security guarantees and relatively efficient operations. It is based on the hardness of problems like Learning with Errors and Short Integer Solution [8]. Lattice-based KEMs are attractive due to their relatively small key sizes and fast operations, making them suitable for practical implementations [9].
- Code-based cryptography—relies on error-correcting codes to provide security. The security of these

schemes is based on the hardness of decoding certain structured codes, making them resistant to quantum attacks. Code-based cryptography uses the difficulty of decoding random linear codes, particularly generalized Goppa codes. One of the oldest forms of post-quantum cryptography, it offers a high level of security but often comes with larger key sizes [10].

- Hash-based cryptography—built upon cryptographic hash functions. These schemes are based on the hardness of finding collisions in the hash function, offering a potential post-quantum solution. Hash-based cryptography's fundamental benefit is that it is a commonly used, well-researched technique that ensures great resistance to quantum assaults, making it a candidate for long-term security in the post-quantum period (as a long enough key is utilized). While primarily used in signature schemes, hash-based cryptography can also be adapted for KEM. These rely on the difficulty of finding preimages or collisions in cryptographic hash functions.
- Multivariate polynormal cryptography—relies on algebraic equations with multivariate polynomials. Although secure, the size of the public and private keys can be large [11, 12].
- Isogeny-based cryptography—based on the mathematics of elliptic curves and isogenies. These schemes rely on constructing mappings between elliptic curves. It offers some of the smallest key sizes of any post-quantum cryptosystem but is relatively slow [13–16].

During the transition to post-quantum cryptography, hybrid KEMs are being used. These combine classical cryptographic methods (like RSA or ECC) with post-quantum methods. For example, a hybrid KEM could simultaneously run both a lattice-based KEM (for post-quantum security) and an RSA-based KEM (for classical security), ensuring safety from both quantum and classical attacks [17].

3.2. Practical challenges in implementing post-quantum KEMs

Implementing post-quantum Key Encapsulation Mechanisms in real-world systems presents several practical challenges.

Many post-quantum KEMs, especially those based on lattice and code-based cryptography, involve much larger key sizes compared to classical systems. For example, code-based KEMs like Classic McEliece have public keys that are several hundred kilobytes in size, which is significantly larger than RSA or ECC public keys. Larger key sizes increase the bandwidth needed for key exchange, which can slow down communication, particularly in low-bandwidth environments such as mobile networks, IoT devices, or satellite communications.

Post-quantum cryptographic schemes often require more computational power than traditional cryptographic algorithms.

Table 1
Performance Assessment

Algorithm	Type	Performance
CRYSTALS-KYBER	Lattice-based	Overall performance of CRYSTALS-KYBER in software, hardware, and hybrid settings is excellent.
BIKE	Code-based	The performance of BIKE would be suitable for most of the applications as confirmed by several hardware benchmarks.
HQC	Code-based	The bandwidth of the HQC exceeds that of BIKE, HQC's key generation but decapsulation only requires a fraction of the kilocycles required by BIKE. HQC is one of the top two alternate KEMs. The overall performance of the HQC is not optimal but still, it is acceptable.
Classic McEliece	Code-based	It has the smallest ciphertext among any of the NIST PQC candidates
SIKE	Isogeny-based	It has relatively low communication costs. However, performance on embedded devices may be an issue because of the time to perform a single key encapsulation/decapsulation.

For instance, lattice-based schemes like CRYSTALS-KYBER are efficient in terms of security but may still require more CPU cycles compared to classical systems. Implementing these algorithms on resource-constrained devices (e.g., IoT devices) can be difficult, as they may not have the processing power required to handle the increased computational load. The increased computation can introduce latency in key exchange processes. This can be a significant issue for real-time systems (e.g., VoIP, video conferencing), where even small delays in establishing secure connections can degrade user experience [17].

Many existing security protocols (e.g., TLS, SSH, IPsec) are based on classical cryptographic primitives like RSA and ECC. Implementing post-quantum KEMs requires either significant changes to these protocols or hybrid systems that combine classical and post-quantum techniques. Modifying or extending widely used protocols to support post-quantum KEMs can be complex [18]. It requires not only software updates but also widespread adoption to ensure that all parties in a communication system can use the new algorithms [8].

One solution to this challenge is the use of hybrid cryptography, where both classical and post-quantum algorithms are used together in a key exchange process. However, this increases the complexity of the system and can further slow down performance. Hybrid methods can add overhead, as two cryptographic algorithms (classical and post-quantum) are run in parallel, increasing computational and communication costs [19].

Many post-quantum algorithms, especially those based on lattice cryptography, are vulnerable to side-channel attacks such as timing attacks, power analysis, and fault injection attacks. These attacks exploit the physical characteristics of a system to recover secret keys. Protecting implementations of post-quantum KEMs from side-channel attacks requires additional countermeasures, such as constant-time implementations or masking techniques. These countermeasures can increase the complexity and reduce the performance of the system [20].

While NIST is in the process of standardizing post-quantum algorithms, the field is still evolving. Organizations face uncertainty when selecting which post-quantum algorithm to implement, as premature adoption could lead to interoperability issues or the need for future upgrades. Additionally, ensuring interoperability between different implementations of post-quantum KEMs remains a challenge. After careful consideration during the third round of the NIST PQC Standardization Process, NIST has identified a candidate algorithm for standardization. NIST will recommend the primary KEM algorithm to be implemented for most use cases: CRYSTALS-KYBER. The four algorithms selected for this fourth round are BIKE, Classic McEliece, HQC, and SIKE. Many industries rely on cryptographic standards and certifications to ensure security (e.g., FIPS 140-2/3). Post-quantum cryptographic systems will need to go through extensive certification processes to be widely adopted in regulated environments, which can be time-consuming.

While post-quantum KEMs are designed based on problems thought to be hard for quantum computers (e.g., lattice problems, isogenies, etc.), there is still some uncertainty about the long-term security of these algorithms. It is possible that future mathematical breakthroughs or quantum algorithms could weaken these assumptions. Building trust in the security of post-quantum KEMs is essential for their widespread adoption. However, businesses may be reluctant to deploy post-quantum solutions until there is a high level of confidence in their security. There is also resistance to change in the industry. Many organizations have deeply entrenched cryptographic infrastructures that rely on RSA or ECC, and the cost of transitioning to post-quantum cryptography may be prohibitive in the short term.

During the transition to post-quantum cryptography, systems will need to support both classical and quantum-safe algorithms simultaneously. This creates additional complexity in terms of key management, negotiation of algorithms, and ensuring that both parties in a communication can agree on a common cryptographic approach. Implementing dual cryptographic systems can lead to security risks if not done properly, such as potential downgrade attacks where an adversary forces the use of weaker, classical algorithms.

The transition to post-quantum KEMs is necessary to ensure long-term security in the face of quantum computing advancements, but it comes with practical challenges related to efficiency, key sizes, integration, and trust. Overcoming these challenges will require advances in both cryptographic research and engineering, as well as widespread industry adoption of post-quantum standards [21].

4. Overview of post-quantum KEM algorithms

NIST has selected several KEMs for standardization, including CRYSTAL-Kyber and BIKE, Classic McEliece, HQC, and SIKE. These algorithms represent some of the most promising candidates for post-quantum Key Encapsulation Mechanisms that have emerged from the NIST Post-Quantum Cryptography Standardization project. Each one is based on different hard mathematical problems, providing diverse approaches to ensuring security in the post-quantum era [22–24].

4.1. CRYSTALS-Kyber

CRYSTALS-Kyber (Cryptographic Suite for Algebraic Lattices) is one of the most prominent lattice-based KEMs and was selected as a NIST finalist due to its efficiency, security, and small key and ciphertext sizes. It is based on the Module Learning With Errors (Module-LWE) problem, a variant of the Learning With Errors (LWE) problem, which is widely regarded as hard for quantum computers to solve [25].

The construction of Kyber follows a two-stage approach: first introduce an INDCPA-secure public-key encryption scheme encrypting messages of a fixed length of 32 bytes, which is called Kyber.CPAPKE. Then use a slightly tweaked Fujisaki–Okamoto (FO) transform to construct the IND-CCA2-secure KEM, which is called Kyber.CCAKEM.

Kyber defines three parameter sets, which we call Kyber512, Kyber768, and Kyber1024. The parameters are listed in Table 2.

Table 2
Parameter sets for Kyber

NIST Level	Designation	n	k	q	n_1	n_2	(du, dv)	b
Level 1	Kyber512	256	2	3329	3	2	(10,4)	2^{-130}
Level 2	Kyber768	256	3	3329	2	2	(10,4)	2^{-164}
Level 3	Kyber1024	256	4	3329	2	2	(11,5)	2^{-174}

n is set to 256 because the goal is to encapsulate keys with 256 bits of entropy (i.e., use a plaintext size of 256 bits in Kyber.CPAPKE.Enc). Smaller values of n would require encoding multiple key bits into one polynomial coefficient, which requires lower noise levels and therefore lowers security. Larger values of n would reduce the capability to easily scale security via parameter *k*.

q as a small prime satisfying $n \mid (q-1)$; this is required to enable fast NTT-based multiplication. There are two smaller primes for which this property holds, namely 257 and 769. However, for those primes we would not be able to achieve the negligible failure probability required for CCA security, so was chosen the next largest, i.e., q = 3329.

k is selected to fix the lattice dimension as a multiple of n; changing *k* is the main mechanism in Kyber to scale security (and as a consequence, efficiency) to different levels.

Kyber offers a good balance between key/ciphertext size and performance. It is more efficient in both space and speed compared to many other post-quantum candidates.

Public Key Size: Ranges from 736 bytes (Kyber512) to 1,568 bytes (Kyber1024).

Ciphertext Size: Ranges from 800 bytes (Kyber512) to 1,568 bytes (Kyber1024).

Kyber's computational efficiency makes it suitable for a wide range of applications, including low-power devices such as IoT and mobile devices. Kyber provides a very efficient encapsulation and decapsulation process, particularly when compared to other post-quantum KEMs.

4.2. BIKE

BIKE (Bit-Flipping Key Encapsulation) is a code-based cryptographic system that uses Quasi-Cyclic Moderate-Density Parity-Check (QC-MDPC) codes. It's designed to offer efficient key encapsulation while relying on the hardness of decoding random linear codes, a well-established hard problem in cryptography.

BIKE is based on the decoding of QC-MDPC codes, which involves error correction methods. This problem has been studied for decades, and no quantum or classical efficient algorithms are known to solve it. The core cryptographic mechanism is the decoding process for MDPC codes, which involves flipping bits to correct errors in a way that only the legitimate parties can succeed.

BIKE's first building block is a public key encryption scheme based on a variant of the Niederreiter framework. The plaintext is represented by the sparse vector (e0, e1), and the ciphertext by its syndrome. The decryption is performed with a decoding procedure. Next, this PKE is converted into an IND-CCA KEM with the application of the Fujisaki-Okamoto transformation. For the scheme to be truly IND-CCA, there must be conditions on the decoding failure rate (also called DFR), which is the case here with the chosen decoder [26, 27].

As defined in the specifications, the parameters should satisfy several constraints. The block length r should be a prime number, and 2 should be primitive modulo r. The parameter w should be such that $w = 2d \approx \sqrt{n}$ with d being odd. In addition, the error weight should be such that $t \approx \sqrt{n}$. Instantiated parameters are present in Table 3.

Table 3
Parameter sets for BIKE

	r	w	t	l	Public key, bits	Private key, bits	Ciphertext, bits
Level 1	12323	142	134	256	12323	2244	12579
Level 2	24659	206	274	256	24659	3346	24915
Level 3	40973	199	264	256	40973	4640	4640

While BIKE offers relatively compact ciphertext sizes, it tends to require larger public keys compared to lattice-based KEMs like Kyber.

Public Key Size: Ranges from 1,254 bytes to 4,140 bytes, depending on the security level.

Ciphertext Size: Approximately 154 bytes to 284 bytes.

BIKE is relatively efficient in the key encapsulation process due to the use of fast bit-flipping decoding algorithms, although it can be slower than some lattice-based systems in some use cases. The reliance on error-correcting codes is a tried-and-tested approach, giving confidence in its long-term security against both classical and quantum attacks.

4.3. Classic McEliece

Classic McEliece is one of the oldest and most established post-quantum cryptosystems, first proposed in 1978. It relies on the hardness of decoding random Goppa codes, a task that has resisted efficient attacks for over four decades. Classic McEliece has gained attention due to its long-standing security record and robust resistance to quantum attacks [28].

The original purpose is to encode data and transmit it on a noisy channel, allowing the receiver to remove the errors to get the correct message. If the decoder is kept secret and cannot be deduced from the encoder, it makes encoding with errors a one-way trapdoor function: the sender encodes with the public encoder and adds as many errors as the decoder can remove. The receiver with the decoder is then the only one who can remove the errors and read the message.

The public key size grows substantially from 261120 bytes at NIST level 1 to 1357824 bytes at NIST level 5c. The private key size also sees a significant increment from 6492 bytes at level 1 to 14120 bytes at level 5c. These increases align with the general principle that larger key sizes translate into stronger security, making the system more resilient against cryptographic attacks. The ciphertext size and the session key size also increase as the NIST level progresses, pointing to stronger security and larger communication overheads [29]. However, the session key size remains consistent at 32 bytes, as its primary role is to ensure confidentiality and integrity during a session, regardless of the NIST level.

Table 4
Parameter sets for Classic McEliece

NIST Level	Designation	Public key, bytes	Private key, bytes	Ciphertext, bytes	Session key, bytes
Level 1	mceliece348864	261120	6492	96	32
Level 3	mceliece460896	524160	13608	156	32
Level 5	mceliece6688128	1044992	13932	208	32
Level 5b	mceliece6960119	1047319	13948	194	32
Level 5c	mceliece8192128	1357824	14120	208	32

Classic McEliece is known for its very large public key sizes, which are its primary drawback, but it has very small ciphertext sizes and extremely fast decapsulation.

Public Key Size: Up to 1 MB or more for higher security levels.

Ciphertext Size: 208 bytes.

Although the public keys are large, the small ciphertext size and fast decryption process make McEliece suitable for high-performance applications where the size of the public key is not a critical issue. The major disadvantage of McEliece is the size of its public keys, which can be as large as several hundred kilobytes to over a megabyte. This makes it less practical for systems with bandwidth or storage limitations.

4.4. HQC

HQC (Hamming Quasi-Cyclic) is another code-based cryptographic system that uses Quasi-Cyclic codes to achieve secure key encapsulation. HQC is based on the hardness of decoding random linear codes but uses a different type of code construction compared to Classic McEliece and BIKE.

HQC is built on the difficulty of decoding random linear codes, which is believed to be a hard problem both for classical and quantum computers.

HQC uses SHAKE256 for multiple purposes e.g., as a PRNG for fixed weight vector generation and random vector generation in Key Generation, as a PRNG for fixed weight vector generation in Encryption, and for hashing in encapsulation and decapsulation. HQC-KEM uses polynomial multiplication in various stages of its operation. In HQC, the fundamental mathematical structure involves cyclic codes, and polynomial operations over finite fields play a crucial role in both the encryption and decryption processes [30].

Polynomial Multiplication is used in the public key generation stage. The public key involves a codeword that is derived from the multiplication of two polynomials, one of which is a secret, and another is a random element. The process relies on encoding the secret key, which consists of small random polynomials, and performing multiplication in a finite field to produce part of the public key.

During the encapsulation process, the key encapsulator generates a random message and encodes it using a public codeword. The encoding procedure involves polynomial multiplication between the message (represented as a polynomial) and the public key polynomial. The ciphertext is generated as the sum of a product of polynomials (including the random polynomials and the public key) along with some error terms. These multiplications are done in a ring of polynomials, where coefficients are taken modulo a prime number.

On the receiver's side, the decryption process also involves polynomial multiplication. The receiver uses their private key to multiply it with part of the ciphertext. By multiplying the ciphertext polynomial by the private key and removing the error components, the original message can be recovered. The structure of the private key, being sparse (i.e., consisting mostly of small entries), ensures that this operation is efficient despite the potential for larger polynomials.

In all cases, these polynomial multiplications are performed in a finite ensuring that the polynomials remain manageable in size and that the modular arithmetic preserves the structure of the quasi-cyclic codes. Polynomial multiplication in HQC is typically implemented using efficient algorithms such as the Number Theoretic Transform (NTT), which is a variant of the Fast Fourier Transform (FFT) for polynomials over finite fields, to speed up the process of large polynomial multiplications. Thus, polynomial multiplication is a critical and recurring operation in the key generation, encryption, and decryption steps of HQC-KEM.

Table 5
Parameter sets for HQC

NIST Level	Designation	n	k	t	Public key, bytes	Private key, bytes	Ciphertext, bytes
Level 1	hqc-128	35338	17669	132	2249	56	4497
Level 2	hqc-192	71702	35851	200	4522	64	9042
Level 3	hqc-256	115274	57637	262	7245	72	14485

HQC's key sizes are intermediate between those of McEliece and BIKE, but its ciphertexts are generally larger.

Public Key Size: Ranges from 2,249 bytes to 7,245 bytes.

Ciphertext Size: Approximately 7,245 bytes to 7,870 bytes.

HQC provides security levels aligned with NIST's requirements, targeting 128-bit and 256-bit classical security levels.

HQC is more efficient in terms of key generation and encapsulation compared to Classic McEliece, while still offering strong security guarantees. As a code-based system, HQC benefits from decades of cryptographic research, giving confidence in its resistance to quantum and classical attacks. Compared to other post-quantum KEMs, HQC produces relatively large ciphertexts, which may be a disadvantage in bandwidth-constrained applications.

4.5. SIKE

SIKE (Supersingular Isogeny Key Encapsulation) is based on isogeny-based cryptography, a relatively new and promising post-quantum cryptographic approach. SIKE uses the difficulty of finding isogenies (mappings) between elliptic curves. While elliptic curve cryptography (ECC) is vulnerable to quantum attacks, isogeny-based systems remain secure.

SIKE is protected by the computational supersingular isogeny (CSSI) problem and allows for an IND-CCA2 key establishment between two parties [31].

The underlying hard problem for SIKE, the Computational Supersingular Isogeny problem, involves finding a secret isogeny (a specific type of function between elliptic curves) between two given supersingular elliptic curves. This is believed to be computationally infeasible, even for quantum computers, making it a good foundation for post-quantum security. SIKE provides IND-CCA2 security, which is a strong form of security for encryption schemes. This means that even if an attacker can request decryptions of ciphertexts of their choice, they cannot learn any useful information about the encryption of a different message. This ensures that the key establishment process between two parties is secure even against active attackers who can manipulate and intercept communications.

SIKE primes are carefully chosen to optimize both performance and security in the context of supersingular elliptic curves and the supersingular isogeny problem. SIKE uses a prime number p of a particular form to define the finite field F_p over which elliptic curves are constructed. The form of these primes allows efficient isogeny computations and ensures that the curves used are supersingular.

SIKE primes are of the form:

$$p=2^a*3^b*f-1$$

where a and b are large integers.

f is a small cofactor, typically set to 1 in many cases.

The prime is of a size that ensures 128-bit, 192-bit, or 256-bit security levels, depending on the target.

Table 6
Parameter sets for HQC

NIST Level	Prime Form	Designation	Public key, bytes	Private key, bytes	Ciphertext, bytes
Level 1	$2^{216}3^{137}-1$	SIKEp434	330	374	346
Level 2	$2^{250}3^{159}-1$	SIKEp503	378	434	402
Level 3	$2^{305}3^{192}-1$	SIKEp610	462	524	486
Level 5	$2^{372}3^{239}-1$	SIKEp751	564	644	596

SIKE offers one of the smallest key and ciphertext sizes of any other post-quantum cryptosystem, making it particularly attractive for use in bandwidth-constrained environments. However, it tends to have slower performance compared to other post-quantum KEMs.

Public Key Size: As small as 330 bytes.

Ciphertext Size: Ranges from 346 to 596 bytes.

SIKE's compact key and ciphertext sizes make it one of the most bandwidth-efficient post-quantum cryptosystems, making it attractive for specific use cases where data size and storage are a concern, despite its slower performance compared to other schemes.

5. Conclusions

Cryptographers, researchers, and industry experts are working together to develop and test these algorithms to ensure their security and efficiency in real-world applications. These algorithms are being evaluated for their ability to resist both classical and quantum attacks as part of the NIST Post-Quantum Cryptography Standardization process. The goal is to identify cryptosystems that will be secure in a future where quantum computers could break existing cryptography, while also being efficient in real-world applications.

KEMs are critical for secure key exchange in cryptographic protocols, enabling two parties to securely establish a shared secret over an insecure channel. In post-quantum cryptography, various KEMs are being explored for their security and practicality in terms of key sizes, speed, and resilience against quantum attacks [32].

The development of these KEMs involves close collaboration between academia, industry, and government organizations. The NIST process, for example, has provided a platform where researchers can submit their cryptographic algorithms for rigorous evaluation by the global cryptographic community. This collaboration ensures that these algorithms are tested for:

Security: To withstand both classical and quantum attacks.

Performance: In terms of speed, key size, and memory usage in practical applications.

Real-World Implementation: Testing includes both hardware and software implementations to ensure that the algorithms are suitable for a range of use cases, from small devices (like IoT) to high-performance systems (like cloud servers) [33, 34].

The ongoing development of post-quantum KEMs like Crystal-Kyber, BIEK, HQC, Classic McEliece, SIKE, and others is crucial to ensuring secure communication in a quantum future. Each of these algorithms brings unique advantages in terms of performance, security, and efficiency, and the NIST competition is helping to refine these technologies for eventual standardization and widespread adoption.

References

[1] L. K. Grover, A Fast Quantum Mechanical Algorithm for Database Search, Proceedings of the 28th Annual ACM Symposium on Theory of Computing (1996).

[2] D. J. Bernstein, J. Buchmann, E. Dahmen, Code-based Cryptography (2016).

[3] Horpenyuk, I. Opirskyy, P. Vorobets, Analysis of Problems and Prospects of Implementation of Post-Quantum Cryptographic Algorithms, in: Classic, Quantum, and Post-Quantum Cryptography, vol. 3504 (2023) 39–49.

[4] Bernhardt, Quantum Computing for Everyone, Cambridge, MA: MIT Press (2019).

[5] V. Cini, et al., CCA-Secure (Puncturable) KEMs from Encryption with Non-Negligible Decryption Errors, Advances in Cryptology – ASIACRYPT 2020. Lecture Notes in Computer Science, vol. 12491. (2020). doi: 10.1007/978-3-030-64837-4_6.

[6] Woodward, Will Quantum Computers Be the End of Public Key Encryption? J. Cyber Secur. Technol. 1(1) (2016). 1–22. doi: 10.1080/23742917.2016.1226650.

[7] L. Chen, Cryptography Standards in Quantum Time: New Wine in an Old Wineskin? IEEE Security & Privacy 15(4) (2017) 51–57. doi: 10.1109/MSP.2017.3151339.

[8] P. Hauke, et al., Perspectives of Quantum Annealing: Methods and Implementations, Reports on Progress in Physics 83(5) (2020).

[9] J. Bernstein, Visualizing Size-Security Tradeoffs for Lattice-Based Encryption, IACR Cryptol, ePrint Arch (2019).

[10] M. Baldi, P. Santini, G. Cancellieri. Post-Quantum Cryptography based on Codes: State of the Art and Open Challenges, AEIT International Annual Conference. (2017). doi: 10.23919/aeit.2017.8240549.

[11] Casanova, et al., A Great Multivariate Short Signature, Submission to NIST (2017).

[12] R. A. Grimes, Cryptography Apocalypse (2020).

[13] A. Bessalov, et al., Implementation of the CSIDH Algorithm Model on Supersingular Twisted and Quadratic Edwards Curves, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3187, no. 1 (2022) 302–309.

[14] A. Bessalov, et al., Modeling CSIKE Algorithm on Non-Cyclic Edwards Curves, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3288 (2022) 1–10.

[15] A. Bessalov, et al., Multifunctional CRS Encryption Scheme on Isogenies of NonSupersingular Edwards Curves, in: Workshop on Classic, Quantum, and Post-Quantum Cryptography, vol. 3504 (2023) 12–25.

[16] A. Bessalov, et al., CSIKE-ENC Combined Encryption Scheme with Optimized Degrees of Isogeny Distribution, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, vol. 3421 (2023) 36–45.

[17] V. Pastushenko, D. Kronberg, Improving the Performance of Quantum Cryptography by Using the Encryption of the Error Correction Data, Entropy 25(956) (2023). doi: 10.3390/e25060956.

[18] U. Banerjee, S. Das, A. P. Chandrakasan, Accelerating Post-Quantum Cryptography using an Energy-Efficient TLS Crypto-Processor, IEEE International Symposium on Circuits and Systems (2020). doi: 10.1109/iscas45731.2020.9180550.

[19] M. Kumar. Post-Quantum Cryptography Algorithm's Standardization and Performance Analysis, Array, 15 (2022). doi: 10.1016/j.array.2022.100242.

[20] F. Borges, P. R. Reis, D. Pereira. A Comparison of Security and Its Performance for Key Agreements in Post-Quantum Cryptography, IEEE Access, 8 (2020) 142413–142422. doi: 10.1109/access.2020.3013250.

[21] Bellizia, et al., Post-Quantum Cryptography: Challenges and Opportunities for Robust and Secure HW Design, IEEE International Symposium on Defect and fault tolerance in VLSI and Nanotechnology systems (DFT) (2021) 1–6. doi: 10.1109/DFT52944.2021.9568301.

[22] L. Chen, et al., Report on Post-Quantum Cryptography, NIST Publications (2016). doi: 10.6028/NIST.IR.8105.

[23] G. Alagic, et al., Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process, NIST Publications (2020). doi: 10.6028/NIST.IR.8309.

[24] G. Alagic, et al., Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process, NIST Publications (2022). doi: 10.6028/NIST.IR.8413.

[25] J. Bos, et al., CRYSTALS - Kyber: A CCA-Secure Module-Lattice-based KEM, 2018 IEEE European Symposium on Security and Privacy (EuroS&P) (2018) 353–367. doi: 10.1109/EuroSP.2018.00032.

[26] L. Demange, M. Rossi, A Provably Masked Implementation of BIKE Key Encapsulation Mechanism, Cryptology ePrint Archive (2024). doi: 10.62056/aesgvua5v.

[27] Y. Lia, L. -P. Wang, Security Analysis of the Classic McEliece, HQC and BIKE Schemes in Low Memory, J. Inf. Secur. Appl. 79 (2023). doi: 10.1016/j.jisa.2023.103651.

[28] O. Kuznetsov, et al., Trade-offs in Post-Quantum Cryptography: A Comparative Assessment of BIKE, HQC, and Classic McEliece, in: Classic, Quantum, and Post-Quantum Cryptography, vol. 3504 (2023) 12–25.

[29] C. Nugier, V. Migliore, Acceleration of a Classic McEliece Postquantum Cryptosystem with Cache Processing, in: IEEE Micro, 44(1) (2024) 59–68. doi: 10.1109/MM.2023.3304425.

[30] R. Azarderakhsh, et al., Hardware Deployment of Hybrid PQC, Cryptology ePrint Archive (2021).

[31] R. Elkhatib, B. Koziel, R. Azarderakhsh, Faster Isogenies for Quantum-Safe SIKE, Topics in Cryptology – CT-RSA (2022) 49–72. doi: 10.1007/978-3-030-95312-6_3.

[32] M. Raavi1, et al., Security Comparisons and Performance Analyses of Post-Quantum Signature Algorithms, ACNS 2021: Applied Cryptography and Network Security (2021) 424–447. doi: 10.1007/978-3-030-78375-4_17.

[33] O. I. Harasymchuk, et al., Generator of Pseudorandom Bit Sequence with Increased Cryptographic Security, Metallurgical and Mining Industry: Sci. Tech. J. 5 (2014) 25–29.

[34] V. Maksymovych, et al., Combined Pseudo-Random Sequence Generator for Cybersecurity, Sensors 22 (2022). doi: 10.3390/s22249700.