Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 29 (2026), No. 4, pp. 1799–1821
DOI : 10.47974/JDMSC-2591

Comparative analysis of Kyber-512, RSA, ECC and AES-128 for secure communication in resource-constrained systems

Kritsanapong Somsuk
Department of Computer and Communication Engineering
Faculty of Technology and Engineering
Udon Thani Rajabhat University
Udon Thani 41000
Thailand

Abstract

Kyber is one of post quantum cryptography techniques that has been formally standardized by NIST. In fact, Kyber-512 exhibits the lowest security but achieves the fastest speed relative to other algorithms within the same family. Hence, this algorithm is appropriate to be implemented on resource-constrained devices regularly seen in IoT applications. The design of this research is divided into two main sections. The first part is a comparison about the process to exchange the secret key across Kyber-512, RSA (2048 bits), and ECC over a prime field (224 bits), using a 256-bit secret key as the secret message. Experimental results indicate that although Kyber-512 requires more processing time than RSA and ECC, it achieves average encryption and decryption periods only 8.3 milliseconds and 2.5 milliseconds, respectively, simulated on a general-purpose computer. However, the primary benefit of Kyber-512 is the resilience against quantum computer attacks. The second part evaluates the performance of Kyber-512 and AES-128, both of which provide analogous security levels. This study exploits two forms of data: a structured dataset and grayscale digital images. The results demonstrate that Kyber-512 requires a significantly greater processing time than AES-128 in both contexts. Therefore, the recommendation is that Kyber-512 should only be used for exchanging the secret key, whereas AES-128 should be employed for encryption and decryption processes.

Subject Classification: *94A60, 00A69, 68R01.*

Keywords: *Kyber-512, AES-128, RSA, ECC over prime filed.*

ORCID-ID: **0000-0002-7264-4628**

E-mail: **kritsanapong@udru.ac.th**

1. Introduction

Communication through network technologies has gained significant popularity because of the simplicity and rapidity. In addition, the internet can be selected to enable the internet of Things (IoT), allowing things to communicate across the internet through various communications. This is achieved by connecting many devices to the internet, enabling them to share information. This process often involves the transmission of significant amounts of information, including sensitive or secret information. Devices connected to the internet typically have limited processing capabilities, making them unsuitable for complex calculations. Therefore, these devices primarily serve for gathering information, while actual data processing is executed by high-performance computers. However, data transmission over networks is inherently insecure. Thus, it is essential to ensure data confidentiality. Cryptography [1, 2] which is the technique to secure the secret information is regularly divided into two groups. The first category is called symmetric key cryptography [3, 4], which employs the secret key for both the encryption and decryption processes. The benefits of this method involve comparatively rapid processing speeds and high security. However, an important challenge emerges when the transmitter and recipient are separated, making secure communication of the secret key extremely difficult. To resolve this problem, asymmetric key cryptography [5] or public key cryptography was developed. This method employs two mathematically correlated keys: the public key and the private key. In data security, the public key is selected for encryption process, while the private key serves for decrypting the encrypted message. This approach provides a better level of security. Nevertheless, a significant limitation of all algorithms in this category is their comparatively high processing time, particularly when deployed on resource-constrained devices such as IoT devices [6]. Therefore, public key algorithms are typically not utilized for data encryption. They are used only for exchanging the secret key [7], then it will be employed by the algorithm in symmetric key cryptography for data encryption. However, the development of quantum computers provides an important threat to traditional cryptographic systems, especially within the area of public key cryptography. These systems are vulnerable to quantum algorithms, especially Shor's algorithm [8], which can factor the large composite integer to break RSA [9].

Post Quantum cryptography (PQC) [10, 11] is a novel category of cryptographic algorithms designed to replace traditional public key

cryptography. The primary benefit of PQC is the capacity for resilience against quantum computer attacks. Kyber [12, 13] is the algorithm in PQC receiving standardization by NIST. This technique is available in three variants: Kyber-512, Kyber-768, and Kyber-1024, each with distinct security levels. Kyber-512 provides the minimal security level because of the utilization of reduced parameters, making it frequently applied in low-resource settings. In addition, the security level is equivalent to AES-128 [14]. However, Kyber is advised to exchange the secret key, while strong algorithms in symmetric key cryptography break are chosen for data encryption.

This research aims to compare the efficiency of Kyber-512 and AES-128 in preserving data secrecy during transmission from IoT device to a high-performance computer. The objective is to determine the best appropriate cryptographic method for usage. The experiment has two segments:

1. An evaluation of the processing time required for exchanging the secret key between Kyber-512 and other algorithms in public key cryptography to determine acceptability.
2. An analysis of the computation time in encryption and decryption between Kyber-512 and AES-128 is tested to find out whether method offers superior processing efficiency under equivalent security parameters.

Kyber-512 and AES-128 were chosen because of their position as the least complex algorithms within their respective families, making them suitable for the implementation on IoT devices.

2. Related Works

This section examines relevant studies, focusing on three primary domains: public key cryptographic algorithms typically employed for secret key exchange; AES, an important symmetric key cryptography algorithm for data protection; and Kyber-512, a post-quantum cryptographic algorithm appropriate for implementation on low-power computing devices such as IoT devices.

2.1 *RSA*

RSA [9] is the technique in public key cryptography developed by Ron Rivest, Adi Shamir, and Leonard Adleman in 1978. The security relies on the computational challenge of factoring large modulus [15, 16, 17, 18]. RSA has been demonstrated that, with the complete realization of quantum computers, RSA will become insecure, since it may be compromised by Shor's Algorithm [8], which employs the quantum Fourier transform to calculate the period for integer factorization. However, while quantum computers are still under development, RSA continues to be one of the most used cryptographic algorithms for data encryption [19, 20] and digital signature [21, 22, 23] at present.

Algorithm 2.1: RSA
Input: m (Original Plaintext) where 1 < m < n
Output: m

1	##Key Generation Process##
2	p_i ← Generate i prime numbers randomly
3	$\Phi(n)$ ← Compute Euler's function from $\Phi(n) = \prod_{j=0}^{i-1}(p_j - 1)$
4	n ← Compute modulus from $\prod_{j=0}^{i-1} p_j$
5	e ← Select the public key, e, from the following conditions 1 < e <
6	$\Phi(n)$ and gcd(e, $\Phi(n)$) = 1
7	d ← Compute the private key, d, from $d = e^{-1} \bmod \Phi(n)$
8	Public e, n to everyone in group
9	##Encryption Process##
10	Receive e and n from Receiver
11	c ← Compute c from $m^e \bmod n$
12	Send c to receiver
13	##Decryption Process##
	Receive c from sender
14	m ← Recover m from $c^d \bmod n$

Algorithm 2.1 defines RSA procedure for data security. The process is divided into three steps including key generation, encryption and

decryption. The key generation process involves the selection of at least two large prime numbers to compute the modulus and Euler's function to produce the public key and the private key. The encryption process uses the public key to transform plaintext into ciphertext, while the decryption process exploits the private key to retrieve the original message. To ensure RSA maintains an adequate degree of security, the modulus must be generated at least 2048 bits [24, 25] and derived from the same size of prime numbers. Although RSA is extensively used in modern applications for secure the secret information, it has certain limitations, especially in contexts demanding high-speed processing under limitation of resources. Researchers have proposed numerous enhancements to overcome these vulnerabilities, including improved decryption algorithms and multi-prime variations to improve speed or enhance security. However, the emergence of quantum computing presents a substantial risk to the long-term sustainability of RSA, since algorithms such as Shor's algorithm may effectively factor large composite numbers, hence damaging the RSA's security. This has resulted in an increasing interest in post-quantum cryptography alternatives.

2.2 *Elliptic Curve Cryptography*

Elliptic Curve Cryptography (ECC) is the other algorithm in public key cryptography. It was proposed in the mid-1980s by Victor Miller and Neal Koblitz [26, 27]. The concept of ECC is based on elliptic curves defined over finite fields, with all necessary cryptographic parameters shown as points on the curve. The principal advantage of ECC is the capacity to provide a security level comparable to RSA while using a smaller key size. For instance, ECC with a 224-bit key offers a security level equivalent to RSA with a 2048-bit [28]. Furthermore, ECC is especially advantageous for applications on devices with limited computing power. In addition, it provides benefits in terms of key size efficiency. Operations involving point addition and scalar multiplication require the execution to ensure both security and efficiency. In fact, this method is divided into several categories based on the characteristics of the field, including ECC over fields of characteristic two and ECC over prime fields. Each category has distinct trade-offs between efficiency and compatibility depending on the specific application. Although its popularity in secure communications, ECC is as vulnerable to quantum computing risks as RSA. Once quantum computers become available, some algorithms may effectively address the

discrete logarithm problem to break ECC. Therefore, ECC is considered insecure from quantum computer.

Algorithm 2.2: Elliptic Curve Cryptography over Prime Field **Input:** m (Original Plaintext) **Output:** m	
1	##Key Generation Process##
2	p ← Select a strong prime number
3	Select the equation from $y^2 = x^3 + ax + b \bmod p$
4	$P = (x_p, y_p)$ ← Select a base point, which is a point on the curve
5	j ← Select integer in the condition' s elliptic curve
6	Q ← Compute the new point from jP
7	Public P, Q to everyone in group
8	##Encryption Process##
9	Receive P and Q and the equation from Receiver
10	k ← Select integer in the condition' s elliptic curve
11	$R = (x_r, y_r)$ ← Compute the new point from kQ
12	Y_0 ← Compute the new point from kP
13	y_1 ← Compute y_1 from $x_r x_m \bmod p$
14	y_2 ← Compute y_2 from $y_r y_m \bmod p$
15	Send Y_0, y_1 and y_2 to receiver
16	##Decryption Process##
17	Receive Y_0, y_1 and y_2 from sender
18	$R = (x_r, y_r)$ ← Compute the point from jY_0
19	x_m ← Compute x_m from $y_1 x_r^{-1} \bmod p$
20	y_m ← Compute y_m from $y_2 y_r^{-1} \bmod p$

Algorithm 2.2 indicates the procedure to implement ECC over prime field for data security. The primary calculations of this approach rely on two key operations: point doubling, implemented for computing 2P, and point addition, applied to determine P+Q, where P and Q are points on the elliptic curve. These operations form the basis of scalar multiplication, which is crucial for data encryption and key exchanging in the systems. To enhance computational efficiency and security, several algorithms have

been developed [29], [30]. These enhancements aim to reduce the number of costly modular inversions and strengthen resistance to side-channel attacks, particularly in resource-constrained environments, including embedded systems and IoT devices. It is important to observe that ECC serves not only for data security but also that its mathematical foundation for cryptographic research. Specifically, some techniques of elliptic curve arithmetic may be used to factor large numbers [31], which is relevant to security in cryptographic systems.

2.3 *AES*

Advanced Encryption Standard (AES) [32, 33] is one of symmetric key cryptography algorithms developed by Joan Daemen and Vincent Rijmen in 2001. In fact, AES formally certified by The National Institute of Standards and Technology (NIST) is presented to replace data encryption standard (DES) and its improvement. In addition, this method is known as the excellent security, efficiency and flexibility for securing the secret information. Moreover, AES which is still widely used in several methods, is classified into three primary sizes: AES-128, AES-192, and AES-256. They are based on the length of secret key, including 128 bits, 192 bits, and 256 bits, respectively. Each variation employs a distinct number of processing rounds: 10 rounds in AES-128, 12 rounds in AES-192, and 14 rounds in AES-256. The rounds include operations such as SubBytes, ShiftRows, MixColumns, and AddRoundKey. Recent study in quantum cryptanalysis have heightened interest in incorporating AES into hybrid systems with post-quantum algorithms, thereby ensuring future resilience while maintaining performance. A comprehensive explanation of AES, including its internal architecture and functionalities, is available in [34].

2.4 *Kyber-512*

Kyber is known as one of public key encryption techniques classified as Post-Quantum Cryptography (PQC). In fact, NIST formally defined this method as part of the first collection of post-quantum algorithms. In addition, Kyber relies on lattice-based cryptography, based on the Module Learning With Errors (M-LWE) problem. The concept of M-LWE incorporates noise into the system, making the fundamental mathematical structure challenging for both classical and quantum computers to resolve. Kyber is offered in three security levels: Kyber-512, Kyber-768, and Kyber-1024. Each version provides a different degree of security and computational cost. Table 1 delineates a comparison of security levels

between Kyber and AES. Kyber-512, with a security level comparable to AES-128, is the lowest efficient algorithm in the group. However, the efficiency renders Kyber-512 especially appropriate for applications requiring devices with limited computing capabilities, such as IoT devices. The design of Kyber focuses on efficiency, simplicity, and ease of implementation, leading to its acceptance as the key encapsulation mechanism (KEM) in the NIST post-quantum cryptography standardization process. Although higher-security versions such as Kyber-768 and Kyber-1024 are favored for the protection of long-term and high-value data, Kyber-512 offers a compromise between security and performance, rendering it suitable for real-time communication and secure key exchange in embedded devices.

Table 1

Comparison of the security levels between Kyber and AES

Kyber-512	AES-128
Kyber-768	AES-192
Kyber-1024	AES-256

Algorithm 2.3: Kyber-512 **Input:** m (Original Plaintext), q = 3329 **Output:** m	
1	##Key Generation Process##
2	A ← Generate Matrix in $\mathbb{Z}_q^{k\times k}$ that in Polynomial of degree 256
3	s ← Generate vector in $\mathbb{Z}_q^k$ that in Polynomial of degree 256
4	e ← Generate vector in $\mathbb{Z}_q^k$ that in Polynomial of degree 256
5	t ← Compute (As + e) mod q
6	Public A, t to everyone in group
7	## Encapsulation Process##
8	Receive A and t from Receiver
9	r ← Generate vector in $\mathbb{Z}_q^k$ that in Polynomial of degree 256
10	e_1 ← Generate vector in $\mathbb{Z}_q^k$ in Polynomial of degree 256
11	e_2 ← Generate an integer in $\mathbb{Z}_q$ in Polynomial of degree 256
12	m′ ← Transform m to Polynomial of degree 256

13	u ← Compute $(A^T r + e_1) \bmod q$
14	v ← Compute $(t^T r + e_2 + m') \bmod q$
15	Send u, v to receiver
16	##Decapsulation Process##
17	Receive u, v from sender
18	$v' =$ Compute $(v - s^T u) \bmod q$
19	Recover m by approximating the result within the predefined boundary values

Algorithm 2.3 demonstrates the implementation of Kyber-512 to secure the secret information and exchange the secret key. The variables *s*, *e*, *r*, e_1, and e_2 must be generated through small coefficients selected from a centered binomial distribution. In fact, these values serve as noise to increase the security of the system. The incorporation of noise achieves two primary functions. Initially, it strengthens the hardness assumption predicated on M-LWE, which is essential to the security of Kyber. Secondly, it ensures that the original message *m* may be precisely retrieved during the decapsulation procedure. Incorrect measurement of noise during the decryption can result in incorrect outcomes or expose the system to certain cryptographic vulnerabilities. Kyber-512 is divided into three processes: key generation, encapsulation, and decapsulation. The precision of retrieving the shared secret during decapsulation is significantly dependent on the proper management of noise and modular arithmetic. The centered binomial distribution is selected to achieve a compromise between computational efficiency and protection against side-channel attacks, especially in resource-constrained environments. The efficiency of this architecture and quantum-resistant security make it particularly suitable for applications requiring both performance and data security.

3. Methodology

This aim of this study is to discover an effective method for securing the secret data. Assuming that the secret message is transferred from IoT devices to high-performance computers, which handle sensitive information in substitute of the resource-constrained IoT devices. The experiment consists of two components. Experiment 1 evaluates the processing time and security associated with exchanging the secret key among Kyber-512, RSA with a 2048-bit and ECC with a 224-bit length. The

aim is to identify the best effective algorithm for key exchange. This experiment also examines the effects of quantum computer attacks, specifically the robustness of each algorithm against Shor's Algorithm. Experiment 2 includes a comparative analysis of the data encryption duration and security efficacy of Kyber-512 and AES-128. The sensitive data is split into two sections. The first part is expected to include data gathered from an IoT device, consisting of three elements: timestamp, temperature, and humidity, initially arranged in the table. Prior to encryption process, the original data in table is transformed into JSON format and then encrypted into ciphertext to transmiss to the recipient. This conversion enables adaptable data transport and promotes future scalability. The recipient decrypts the ciphertext and extracts the original message in JSON format. The last step is transforming the JSON back into a table to restore the data to the original format. This experiment employs several data sizes to evaluate the encryption and decryption durations. Algorithm 3.1 delineates the methodology for protecting a dataset in CSV or XLSX format.

Algorithm 3.1: Securing the secret data in csv or xlsx format
Input: d (dataset in csv or xlsx format)
Output: Original file in Table format

1	##Process in Sender Side##
2	d ← Read dataset in csv file or xlsx
3	j ← Convert m to JSON file
4	s ← Convert j to String format
5	m ← Convert s to message for Kyber-512 or AES-128
6	c ← Encrypt m by using Kyber-512 or AES-128
7	Sent c to Receiver
8	##Process in Receiver Side##
9	Receive c from sender
10	m ← Decrypt c by using Kyber-512 or AES-128
11	s ← Convert m to String format
12	j ← Convert s to JSON format
13	d ← Convert j to structure data (Table)

Assuming that the data acquired from the IoT device includes digital images, recorded by CCTV cameras connected to IoT systems. Processing images for training and testing directly on IoT devices is inappropriate because of their constrained computing capabilities. Therefore, it is essential to transfer the images to a server to construct the model from these datasets. Nevertheless, if digital images include secret information,

cryptography methods must be deployed to ensure their security. This experiment performs a comparison between Kyber-512 and AES-128. The process starts with the acquisition of the digital image, followed by its conversion to grayscale, which reduces image resolution and improves processing speed. Each pixel, shown as an 8-bit, is then removed and concatenated to create the plaintext for encryption and decryption processes. The result is divided into smaller segments according to the plaintext constraints established by each technique. Algorithm 3.2 delineates the methodology for protecting digital images.

Algorithm 3.2: Securing the digital image **Input:** im (digital image), s (size of image for processing) **Output:** Original Image	
1	##Process in Sender Side##
2	im ← Read digital image file
3	im_gray ← Convert im to gray scale image
4	l ← size of im_gray
5	IF l is not equal to m Then
6	im_gray ← Resize im_gray to s
7	End IF
8	s ← Each extracted 8-bit pixel value is concatenated sequentially
9	m ← Convert s to message for Kyber-512 or AES-128
10	c ← Encrypt m by using Kyber-512 or AES-128
11	Sent c to Receiver
12	##Process in Receiver Side##
13	Receive c from sender
14	m ← Decrypt c by using Kyber-512 or AES-128
15	s ← Divide m into groups of 8 bits to represent the value of each pixel.
16	im_gray ← Convert s to grayscale image

The experimental results of both studies indicate that if AES-128 demonstrates a more powerful processing speed than Kyber-512, it is advisable to select Kyber-512 for exchanging the secret key for AES-128. On the other hand, if Kyber-512 outperforms AES-128, it is recommended to eliminate the key exchange step and implement Kyber-512 directly for data encryption.

4. Experimental Results

This section defines the experimental results of the investigation, divided into two parts. To reduce experimental bias, a consistent set of

tools and equipment was employed throughout the study. The instruments used in this research are as follows: The programming language was Python 3.9.7, and the NumPy library was selected for solving several equations related to Kyber-512. Crypto.Cipher module from the PyCryptodome library was employed to implement AES. An Intel® Core™ i5 CPU operating at 2.53 GHz with 8 GB of RAM was used for all implementations. Although the tool used in this research not being an IoT device, the aim is to evaluate and determine the method that requires the minimal computing resources and processing duration. The method that satisfies these requirements will be chosen on actual IoT devices, guaranteeing low processing duration while preserving a similar degree of security.

Experiment 1 includes a comparative analysis of the duration required to exchange the secret key among three algorithms, including Kyber-512, RSA (2048 bits), and ECC (224 bits). The secret key with 256 bits was chosen for implementation. This key may be modified for AES-128 by partitioning the output into two segments and using just one segment as the AES key. The experimental results are shown in Table 2.

Table 2

Comparison about time to exchange the secret key exchange between Kyber-512, RSA, and ECC

Algorithm	Encryption Time (milli Seconds)	Decryption Time (milli Seconds)
Kyber-512	8.3	2.5
RSA (2048 bits)	0.543	1.29
ECC (224 bits)	0.11	0.18

Table 2 displays the computation time in milliseconds for the encryption and decryption operations of three different algorithms for exchanging the secret key, including Kyber-512, RSA (2048 bits), and ECC (224 bits). The results indicate that Kyber-512 achieved the highest time in both encryption and decryption processes, 8.3 milliseconds and 2.5 milliseconds, respectively. In fact, RSA (2048 bits) exhibited better performance, requiring only 0.543 milliseconds for encryption process and 1.29 milliseconds for decryption process. However, RSA is vulnerable to quantum attacks, particularly Shor's algorithm, rendering it unsuitable for future-proof applications. Additionally, ECC (224 bits) demonstrated

Timestamp	Humidity (%)	Temperature (C)
1/10/2024 10:00	63	28
1/10/2024 10:05	64	29
1/10/2024 10:10	66	27
1/10/2024 10:15	68	30
1/10/2024 10:20	70	31
1/10/2024 10:25	66	29
1/10/2024 10:30	64	28
1/10/2024 10:35	69	30
1/10/2024 10:40	71	32
1/10/2024 10:45	65	28
1/10/2024 10:50	68	30
1/10/2024 10:55	67	29
1/10/2024 11:00	66	28
1/10/2024 11:05	64	27
1/10/2024 11:10	69	30

Figure 1

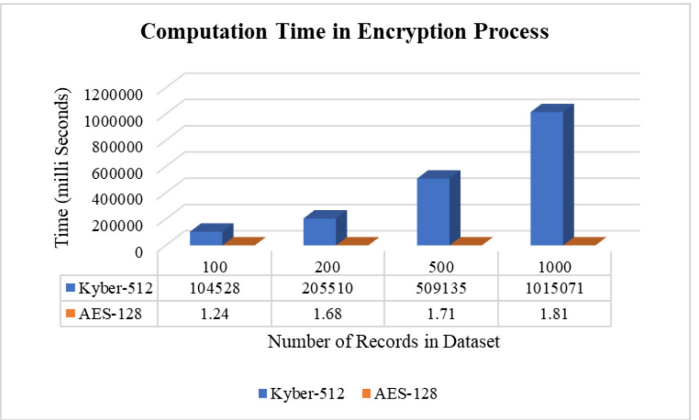
Example of dataset to be encrypted by using Kyber-512 and AES-128

superior performance compared to the other three techniques, requiring only 0.11 milliseconds for encryption process and 0.18 milliseconds for decryption process. Nevertheless, ECC is vulnerable to quantum attacks.

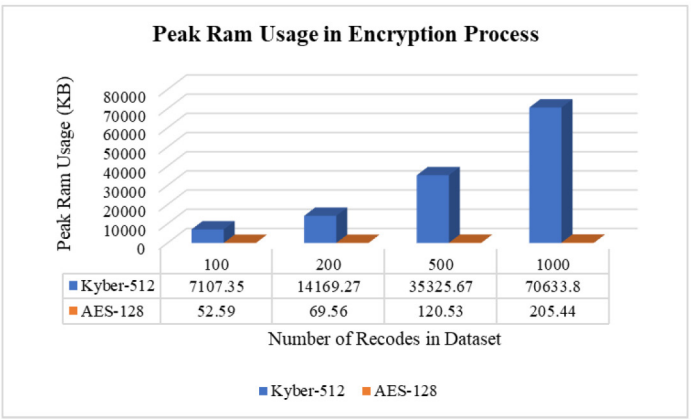
Experiment 2 involves a comparison about computation time required for the encryption and decryption processes. This experiment has two separate parts. Part 1 matches the transmission of a dataset gathered by an IoT device alongside several sensors, prior to its delivery to the recipient. The dataset is encrypted to guarantee confidentiality. The dataset used in this experiment is divided into several sizes based on the amount of data. Each record has three attributes: Timestamp, Humidity, and Temperature, as seen the example in Figure 1.

Figure 1 shows an example dataset obtained from an IoT device. The data is sent to a server that assumes the learning responsibilities from the IoT device. IoT devices are unsuitable for data training and testing because of constraints in computational resources, including CPU, memory, and battery consumption, especially for training, which necessitates high-performance computing capabilities.

Figure 2 illustrates the comparison related to computational time and peak RAM usage during the encryption process between Kyber-512 and AES-128, with plaintext formatted as CSV or XLSX files. Figure 2.a demonstrates the encryption duration (in milliseconds) required by Kyber-512 and AES-128 for different dataset sizes, specifically for 100, 200,



a) Computation Time in Encryption Process



b) Peak Ram Usage in Encryption Process

Figure 2

Comparison about Computation time and Peak Ram Usage in Encryption Process between Kyber-512 and AES-128 using plaintext in the form of CSV or XLSX files

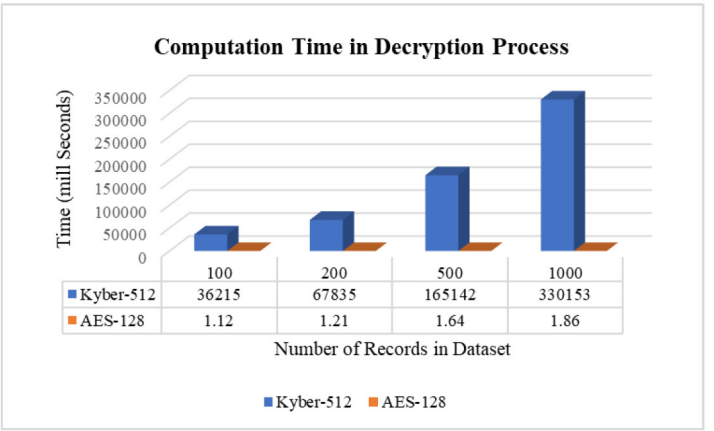
500, and 1000 records. The results clearly indicate significant variations in computing efficiency between the two techniques. Kyber-512 exhibits a linear increase in encryption duration when the quantity of records increases. Processing 100 records takes roughly 7107.35 milliseconds, whereas processing 1000 records escalates to 70,633.8 milliseconds. This illustrates the significant computational complexity of Kyber-512, which

may present difficulties for real-time or large-scale data encryption, particularly on devices with constrained computing power, such as IoT devices. AES-128 demonstrates regularly low encryption times, ranging from 1.24 milliseconds for 100 records to 1.81 milliseconds for 1,000 records. The little increase in time is insignificant, demonstrating exceptional scalability and efficiency as the dataset expands. This confirms AES-128's appropriateness for applications necessitating great performance and minimal latency. AES-128 significantly surpasses Kyber-512 in encryption speed across all dataset sizes from a performance viewpoint.

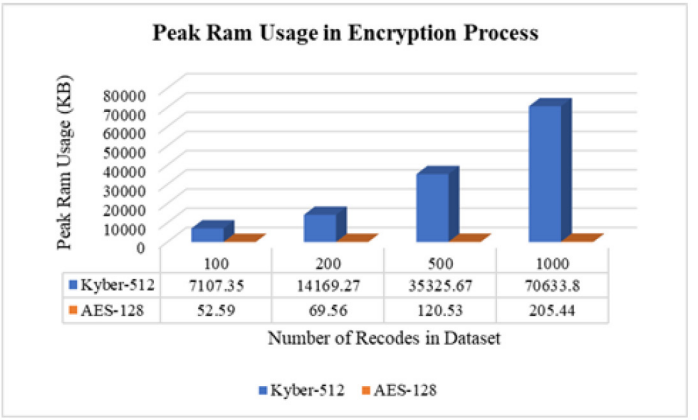
Furthermore, figure 2.b shows the peak RAM usage throughout the encryption process using Kyber-512 and AES-128 across different dataset sizes. The results indicate that Kyber-512 requires much more RAM than AES-128 in every situation. For example, encrypting 1,000 records using Kyber-512 requires around 70,633.8 KB, whereas AES-128 necessitates only 205.44 KB. Kyber-512 exhibits a significant increase in memory use as the dataset size expands, while AES-128 has a very low and stable memory usage. This shows a significant drawback of Kyber-512 in memory-constrained contexts and emphasizes the suitability of AES-128 for lightweight applications.

Figure 3 compares the computation time and peak RAM usage throughout the decryption process between Kyber-512 and AES-128, using plaintext formatted as CSV or XLSX files. Figure 3.a demonstrates the decryption duration (in milliseconds) for Kyber-512 and AES-128 across different dataset sizes: 100, 200, 500, and 1,000 records. Similarly, performance trends are seen for both techniques when compared to the encryption process shown in Figure 2. In fact, Kyber-512 has a continuously raised decryption duration, escalating markedly with dataset size from 6,430 ms for 100 records to 63,502 ms for 1,000 records. On the other hand, AES-128 has exceptional performance, with decryption durations marginally exceeding its encryption durations. For example, decrypting 100 records requires only 1.12 milliseconds, whereas decrypting 1,000 records necessitates 1.86 milliseconds. The little increase further substantiates the scalability and computational efficiency of AES-128.

Furthermore, Figure 3.b illustrates the peak RAM use during the decryption process using Kyber-512 and AES-128 across various dataset sizes. Similarly, to the encryption procedure, Kyber-512 exhibits much greater RAM usage than AES-128. As the record count escalates from 100 to 1,000, Kyber-512's RAM consumption surges significantly from 7,107.35 KB to 70,633.8 KB, while AES-128's consumption stays relatively low,



a) Computation Time in Decryption Process



b) Peak Ram Usage in Decryption Process

Figure 3

Comparison about Computation time and Peak Ram Usage in Decryption Process between Kyber-512 and AES-128 using plaintext in the form of CSV or XLSX files

rising slightly from 52.59 KB to 205.44 KB. This result verifies that AES-128 is much more memory-efficient, making it a superior option for memory-constrained systems, whereas Kyber-512 may be more appropriate for applications requiring quantum resistance in efficiency.

Comparing the results of encryption (Figure 2) and decryption (Figure 3) reveals that AES-128 significantly surpasses Kyber-512 in both

tasks. Kyber-512, while safe and immune to quantum attacks, exhibits significantly increased computing time in both phases, particularly with large datasets. This performance disparity underscores the advisability of using a hybrid cryptographic approach: Apply Kyber-512 only for the first key exchange to classify on its post-quantum security. In contrast, AES-128 is used for the encryption and decryption of real data, guaranteeing rapid and efficient processing, especially appropriate for resource-constrained IoT situations.

Part 2 exhibits the transmission of a digital image acquired by an IoT device connected to a camera. Prior to transmission to the server, the digital image, typically sized at 300 × 300 pixels, is transformed to a greyscale format to minimize its size, hence reducing processing time. Each pixel is then transformed into an 8-bit value and concatenated to create the plaintext used for encryption and decryption processes.

Figure 4

Original Image for the Experiment

Figure 4 displays the original color image used in this research. Prior to the encryption and decryption operations, the image is reduced to 300 × 300 pixels and transformed into a grayscale format.

Figure 5 displays the outcomes of encrypting and decrypting the original image from Figure 4 using Kyber-512 and AES-128. Figure 5.a illustrates a piece of the ciphertext generated by Kyber-512. Figure 5.b illustrates the greyscale picture restored after decryption using Kyber-512.

1248, 3055, 1403, 3214, 2764, 1177, 1314, 3074, 1056, 2583, 898, 555, 537, 2287, 599, 1093, 1754, 250, 224, 3218, 2208, 615, 3077, 729, 1669, 1078, 2072, 1270, 3110, 1077, 3252, 1675, 1604, 1545, 1845, 3064, 1378, 2740, 3024, 247, 3061, 3277, 2206, 3259, 554, 3137, 803, 3259, 3037, 1044, 3288, 3077, 1163, 2885, 2010, 279, 130, 826, 2640, 226, 1188, 3101, 2994, 2928, 1082, 2161, 3286, 997, 1935, 2200, 1474, 340, 2916, 217, 3058, 1549, 1929, 1912, 2507, 2829, 2469, 2994, 37, 2085, 1285, 1483, 1385, 474, 914, 3073, 902, 129, 1010, 85, 258, 755, 1679, 395, 2963, 586, 204, 489, 1678, 376, 2860, 1279, 2585, 2653, 2032, 2103, 820, 1397, 53, 1024, 1783, 2320, 1975, 360, 2555, 2122,

a) A part of the ciphertext generated from encrypting the original image using Kyber-512

b) Recovered Image from Kyber-512

b'\xa5)~\xaaX\x89\x91@L\xa5\x98\xc2P*
\xfcn\x96f|\xbc\xa2{\x12o\x99!|\xff\x1d\x
84\x83\xfe\xf1-\xbbdw*\xc0\x07\xbf\xd2
\x86\xc6\xe4PIW\xf9}\xcf\x98\x01\xe43\x
8d"\x84)\x86d\xd6m6\xa7\x93\xbc\x90\x
c6\xd5vpW\xabzr*v\xe2\x0b\x1c\xd2\xff
2x[\x8c\xe4Lux\x96v\x07D^5\xb2\xd4\xc
eF\x1fG%\x0f\xd4\xfa\xf2\xa9\x9exPS\xc
8\xc5J=\xe5\xee~\x07\xd1\xe6\xafb\x8c
\xcb\xe0\x852_\xf44\xbc\x9fKN\x06~\xff
\x19Z\xc8\xce$\x14\xd6\x05.\xbc\t\xa7a\
x0c\x98\x97a\x83`\x9e\x07\xdfCP5\x9d\x
aa\xb7\x975\xa2\x8b\xf0\x0f\x05s\x17\x1
eZ\xe6A\xcd\xbaT[\xf7wFK\xce\x97\xb1e

c) A part of the ciphertext generated from encrypting the original image using AES-128

d) Recovered Image from AES-128

Figure 5

Results from encryption and decryption using Kyber-512 and AES-128

Figure 5.c illustrates a segment of the ciphertext produced by AES-128, whereas Figure 5.d displays the greyscale image retrieved after decryption using AES-128. The ciphertext generated by Kyber-512 comprises the coefficients of polynomials of degree 256.

Table 3

Comparison of encryption and decryption times for image encryption between Kyber-512 and AES-128

Algorithm	Encryption Time (milli Seconds)	Peak RAM In Encryption (KB)	Decryption Time (milli Seconds)	Peak RAM In Decryption (KB)
Kyber-512	1072622	73112.71	350821	6097.24
AES-128	3.26	1443.81	2.13	2111.99

Table 3 displays the encryption and decryption periods (in milliseconds) for an image that has been reduced to 300 × 300 pixels and transformed to greyscale prior to processing. The comparison includes two cryptographic algorithms: Kyber-512 and AES-128. Kyber-512 required 20,442 milliseconds for image encryption and 66,280 milliseconds for decryption. This indicates the significant computational complexity of Kyber-512, characteristic of lattice-based post-quantum cryptography methods. This complexity is a compromise for its quantum-resistant security, rendering it suitable for secure situations but perhaps ineffective for real-time or resource-limited applications, including IoT devices. AES-128 exhibited incredible efficiency, requiring just 3.26 milliseconds for encryption and 2.13 milliseconds for decryption. The results validate the rapidity and minimal resource consumption of AES-128, confirming its appropriateness for real-time image processing and transmission in systems with constrained computing capacity. The peak RAM usage of AES-128 is uniformly inferior to that of Kyber-512 across all dataset sizes. This discovery emphasizes the memory efficiency of AES-128, making it more suitable for systems with constrained memory capacity. In contrast, the significantly larger RAM required by Kyber-512 may provide obstacles for implementation on resource-limited systems.

The results clearly demonstrate that AES-128 much leads Kyber-512 regarding computational efficiency in the context of visual data. AES-128 lacks resilience to quantum attacks, whereas Kyber-512 is especially designed for post-quantum security. Therefore, in actual situations when quantum resilience is not urgently required, AES-128 is the preferred option because of its rapidity and efficiency. In contrast, Kyber-512 may be designated for high-security applications, particularly in systems designed to counter future quantum attacks.

Conclusions

The aim of this study is to determine the most effective cryptographic method for securing secret information sent between low-power devices and servers. The research is divided into two primary experiments. The first experiment evaluates the efficacy of Kyber-512, RSA (2048 bits), and ECC over prime field (224-bit) to identify the optimal technique for exchanging the secret key. The results indicate that Kyber-512 requires more processing duration than RSA and ECC. However, the average encapsulation and decapsulation durations were 8.3 milliseconds and 2.5 milliseconds, respectively, which remain satisfactory for several practical applications. Significantly, Kyber-512 exhibits resistance to quantum computer attacks, making it a more future-proof option compared to RSA and ECC. Therefore, the application of Kyber-512 for key exchange in systems necessitating post-quantum security is advised. The second experiment assesses the efficacy of Kyber-512 and AES-128 in data encryption and decryption. These two algorithms were chosen for their analogous security ratings. The evaluation data included two categories: made-up sensor datasets, indicative of data gathered from IoT devices, and greyscale digital images. The results of the study demonstrate that Kyber-512 requires much more processing time than AES-128 in both cases. Therefore, Kyber-512 is inappropriate for use in large data encryption on low-power devices. It should be used only for exchanging the secret key, following which the shared key may be employed by AES-128 for data encryption and decryption. This hybrid methodology offers quantum-resilient security with efficient performance, making it an effective option for secure communication in resource-limited settings.

References

[1] G. Brassard, "Relativized cryptography", *IEEE Transactions on Information Theory*, vol.29, no. 6, pp. 877-894 (1983).

[2] C.H. Meyer, "Cryptography-a state of the art review", in Proc. of VLSI and Computer Peripherals. COMPEURO 89, IEEE, West Germany, pp. 4/150-4/154 (1989).

[3] A. Braeken, "Public key versus symmetric key cryptography in client–server authentication protocols", *International Journal of Information Security*, vol.21, no. 1 pp. 103-104 (2022).

[4] A.J. Elbirt and C. Paar, "An instruction-level distributed processor for symmetric-key cryptography", *IEEE Transactions on Parallel and distributed Systems*, vol.16, no. 5, pp. 468-480 (2005).

[5] W. Diffie and M.E. Hellman, "New directions in cryptography", *IEEE Trans. Inf. Theory*, vol. 22, pp. 644–654 (1976).

[6] F. Thabit, O. Can, A.O. Aljahdali, G.H. Al-Gaphari, and H.A. Alkhzaimi, "Cryptography algorithms for enhancing IoT security", *Internet of Things*, vol. 22, pp. 100759 (2023).

[7] K. Somsuk and C. Sanemueang, "Increasing security to public key cryptography for point-to-point communication", *Journal of Discrete Mathematical Sciences & Cryptography*, vol.26, no. 1, pp. 215-229 (2023).

[8] P.W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring", In Proc. of Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, pp. 124–134 (1994).

[9] R.L. Rivest, A. Shamir and L. Adleman, "A method for obtaining digital signatures and public key cryptosystems", *Commun. ACM*, vol. 21, pp. 120–126 (1978).

[10] D.T. Dam, T.H. Tran, V.P. Hoang, C.K. Pham and T.T. Hoang, "A survey of post-quantum cryptography: Start of a new race", *Cryptography*, vol. 7, no. 3, pp. 40 (2023).

[11] F. Borges, P.R. Reis and D. Pereira, "A comparison of security and its performance for key agreements in post-quantum cryptography", *IEEE Access*, vol. 8, pp. 142413-142422 (2020).

[12] D. C. Lawo, R. Frantz, A. Cano Aguilera, X. Arnal I Clemente, M. P. Podleś, J.L. Imaña, I. Tafur Monroy and J.J. Vegas Olmos, "Falcon/Kyber and Dilithium/Kyber Network Stack on Nvidia's Data Processing Unit Platform", *IEEE Access*, vol.12, pp. 38048-38056 (2024).

[13] H. Kim, H. Jung, A. Satriawan and H. Lee, "A Configurable ML-KEM/Kyber Key-Encapsulation Hardware Accelerator Architecture", *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 71, no. 11, pp. 4678-4682 (2024).

[14] K.L. Tsai, Y.L. Huang, F.Y., Leu, I. You, Y.L. Huang and C.H. Tsai, "AES-128 based secure low power communication for LoRaWAN IoT environments", *IEEE Access*, vol.6, pp. 45325-45334 (2018).

[15] R. Zhang, J. Bi, L. Li and H. Peng, "An optimal bound for factoring unbalanced RSA moduli by solving Generalized Implicit Factoriza-

tion Problem", *The Journal of Supercomputing*, vol.81, no. 1, pp. 1-24 (2025).

[16] M. Wiener, "Cryptanalysis of short RSA secret exponents", *IEEE Transactions on Information Theory*, vol. 36, pp. 553-558 (1990).

[17] J.M. Pollard, "Monte Carlo methods for index computation ($mod p$)", *Mathematics of computation*, vol. 32, no. 143, pp. 918- 924 (1978).

[18] V. Dossou-Yovo, A. Nitaj and A. Togbé, "Improved cryptanalysis of RSA", *Journal of Discrete Mathematical Sciences & Cryptography*, vol.27, no. 3, pp. 945-961 (2024).

[19] D.S. Abdul-Zahra and H.R. Yassein, "Proposed development of RSA via new multidimensional algebra", *Journal of Discrete Mathematical Sciences & Cryptography*, vol.28, no. 2, pp. 375 - 380 (2025).

[20] K. Song, H. Hu, L. Yan, X. Hou and J. Wei, "Parallel RSA encryption algorithm based on a ternary optical computer", *Applied Optics*, vol.63, no. 25, pp. 6636-6645 (2024).

[21] D. Singh and S. Kumar, "Image authentication and encryption algorithm based on RSA cryptosystem and chaotic maps", *Expert Systems with Applications*, vol.274, no. 2, pp. 126883 (2025).

[22] K. Somsuk, "The special algorithm based on RSA cryptography for signing and verifying digital signature", *Heliyon*, vol.11, pp. e42481 (2025).

[23] K. Somsuk, "The development of signing and verification methods for high speed digital signatures on electronic official documents by using RSA cryptography", *Cogent Engineering*, vol.11, no. 1, pp. 2432513 (2024).

[24] R.S. Durge and V.M. Deshmukh, "Securing Cloud Data: A hybrid encryption approach with RSA and AES for enhanced security and performance", *Journal of Integrated Science and Technology*, vol.13, no. 3, pp. 1060-1060 (2025).

[25] R. Acheampong, D.M. Popovici, T. Balan, A. Rekeraho and M.S. Ramos, "Enhancing Security and Authenticity in Immersive Environments", *Information*, vol.16, no. 3, pp. 191 (2025).

[26] N. Koblitz, "Elliptic Curve Cryptosystems", *Mathematics of Computation*, vol. 48, pp. 203–209 (1987).

[27] V.S. Miller, "Uses of elliptic curves in cryptography", *Lecture Notes in Computer Science*, vol. 218, pp. 417–428 (1986).

[28] K. Jang, V. Vikas, A. Baksi, S. Sarkar and H. Seo, "New Quantum Cryptanalysis of Binary Elliptic Curves", *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2025, no. 2, pp. 781-804 (2025).

[29] H. Junru, "The improved elliptic curve digital signature algorithm", In Proc. of International Conference on Electronic & Mechanical Engineering and Information Technology, IEEE, Harbin, China, pp. 257-259 (2011).

[30] M.I. Reddy, M. P. Reddy, R. O. Reddy and A. Praveen, "Improved elliptical curve cryptography and chaotic mapping with fruitfly optimization algorithm for secure data transmission", *Wireless Networks*, vol. 30, no. 3, pp. 1151-1164 (2024).

[31] H. W. Lenstra Jr., "Factoring integers with elliptic curves", *Annals of Mathematics*, vol.126, no. 3, pp. 649–673 (1987).

[32] A. Hafsa, A. Sghaier, J. Malek and M. Machhout, "Image encryption method based on improved ECC and modified AES algorithm", *Multimedia Tools and Applications*, vol. 80, pp.19769-19801 (2021).

[33] M.Y. Shakor, M. I. Khaleel, M. Safran, S. Alfarhood and M. Zhu, "Dynamic AES encryption and blockchain key management: a novel solution for cloud data security", *IEEE Access*, vol.12, pp. 26334-26343 (2024).

[34] H.M. Mohammad and A.A. Abdullah, "Enhancement process of AES: a lightweight cryptography algorithm-AES for constrained devices", *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 20, no. 3, pp. 551-560 (2022).

Received April, 2025