Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 3, June 2026, pp. 2665~2676
ISSN: 2302-9285, DOI: 10.11591/eei.v15i3.11157

Design and evaluation of a secure key exchange protocol using the Kyber-Dilithium algorithm

Bambang Harjito, Muhammad Defaroyan, Fajar Muslim, Ery Permana Yudha, Endra Pratama
Cyber Security and Cloud Technology, Department of Informatics, Faculty of Information Technology and Data Science, Sebelas Maret University, Surakarta, Indonesia

Article Info

Article history:

Received Aug 8, 2025
Revised Mar 11, 2026
Accepted Mar 31, 2026

Keywords:

Authenticated key exchange
Dilithium
Kyber
Post quantum cryptography
Transport layer security

ABSTRACT

Over 90% of the billions of people who use the internet globally use it through the transport layer security (TLS) protocol. TLS is a security standard that performs network authentication and data encryption when accessing the internet. Authenticated key exchange (AKE) is the protocol TLS uses for network authentication and key establishment during the TLS Handshake process. The AKE protocol utilizes a public key cryptosystem (PKC) and digital signatures with algorithms commonly used, namely elliptic curve cryptography (ECC) and Rivest-Shamir-Adleman (RSA). Future advancements in quantum computing may compromise the security of the widely used ECC and RSA algorithms. This research conducts an implementation and comparative analysis of post-quantum algorithms resistant to quantum computer attacks, specifically Kyber-Dilithium, in the context of the AKE protocol. The implementation is performed at three security levels: 128-bit, 192-bit, and 256-bit. The results show that the Kyber-Dilithium is greater than those of the RSA variant and much larger than those of the ECC variant. In contrast to the ECC and RSA variants, the Kyber-Dilithium algorithm variants perform better across all security levels, even if their byte sizes are greater.

Corresponding Author:

Bambang Harjito
Cyber Security and Cloud Technology, Department of Informatics
Faculty of Information Technology and Data Science, Sebelas Maret University
Ir. Sutami, No.36 A Kentingan Jebres, Surakarta, 57126, Indonesia
Email: bambang_harjito@staff.uns.ac.id

1. INTRODUCTION

The Internet is used daily by billions of users worldwide, making secure communication a fundamental requirement of modern digital infrastructure. Transport layer security (TLS) has become the de facto standard for securing internet communications, with more than 90% of global internet connections protected by TLS as of 2020 [1], [2]. TLS provides confidentiality, authenticity, and integrity of data exchanged over potentially malicious networks [3], [4]. These security properties are primarily established during the TLS handshake, which performs authentication and session key establishment using an authenticated key exchange (AKE) protocol before secure communication begins [5], [6].

The AKE protocol enables two communicating parties to securely establish a shared session key over an insecure channel [7], [8]. This process typically relies on public-key cryptography and digital signatures, after which the derived session key is used for symmetric encryption of data. Currently, widely deployed AKE mechanisms in TLS use classical cryptographic algorithms such as Rivest-Shamir-Adleman (RSA), elliptic curve Diffie–Hellman (ECDH), and elliptic curve digital signature algorithm (ECDSA) [9]. The security of

these algorithms is based on the computational hardness of integer factorization (IF) and the elliptic curve discrete logarithm problem (ECDLP) [8], [10].

However, the rapid development of quantum computing poses a significant threat to these classical cryptographic foundations. Grover's algorithm can compromise the adequate security of symmetric and asymmetric cryptosystems by providing a quadratic speedup for brute-force key searches, necessitating an increase in security parameters accordingly [11]. More critically, Shor's algorithm can efficiently solve IF and discrete logarithm problems, rendering RSA and elliptic curve–based cryptographic schemes fundamentally insecure in a post-quantum era [12]. Unlike Grover's algorithm, the threat posed by Shor's algorithm cannot be mitigated by merely increasing key sizes, necessitating the adoption of fundamentally new cryptographic primitives.

In response to these challenges, the National Institute of Standards and Technology (NIST) has initiated and completed a standardization process for post-quantum cryptography (PQC). As a result, CRYSTALS-Dilithium, Falcon, and SPHINCS+ have been standardized for digital signatures, while CRYSTALS-Kyber has been selected as the primary public-key encapsulation mechanism (KEM) for key establishment [13], [14]. These algorithms are designed to be resistant to both classical and quantum attacks, making them suitable candidates for future TLS and AKE implementations [14].

Several studies have explored tightly secure AKE protocols and their variants [5], [6], [15] incorporating techniques such as KEMs, digital signatures, symmetric encryption, and forward secrecy. However, most of these works remain based on classical Diffie–Hellman assumptions and have not adopted NIST-standardized PQC algorithms. Other research has investigated the integration of Kyber into AKE protocols [13], [16], [17], but without incorporating post-quantum digital signatures for authentication. Consequently, a comprehensive implementation that combines both post-quantum KEM and post-quantum digital signature algorithms within a tightly secure AKE framework remains underexplored [17].

Therefore, this study aims to address this research gap by implementing and evaluating an AKE protocol that integrates CRYSTALS-Kyber for key encapsulation and CRYSTALS-Dilithium for digital signatures. The novelty of this work lies in the combined application of NIST-standardized PQC algorithms for both key exchange and authentication within the AKE protocol, followed by a detailed performance evaluation. This contribution provides practical insights into the feasibility and efficiency of post-quantum AKE mechanisms for securing future TLS communications.

The remainder of this paper is organized as follows: section 2 discusses CRYSTALS-Kyber and CRYSTALS-Dilithium and their roles in post-quantum key exchange and digital signatures; section 3 describes the methodology used for system design, implementation, and evaluation; section 4 presents and analyzes the experimental results; and finally, section 5 concludes the paper and outlines directions for future research.

2. CRYSTAL KYBER AND CRYSTAL DILIHIUM

CRYSTALS-Kyber and CRYSTALS-Dilithium are two leading post-quantum cryptographic algorithms developed under the NIST standardization process, designed to provide secure key encapsulation and digital signature mechanisms resistant to quantum computer attacks." Federal Information Processing Standards (FIPS) are making ML-DSA which is derived from Crystal-Dilithium as standard on the FIPS-204 [9], [18] Crystal-Kyber is also derived to become standard under FIPS-203 with the new name ML-KEM [19], [20]. Both algorithms are built based on module lattices that are utilizing number-theoretic transform (NTT) which is an efficient method to calculate ring polynomials, matrix and vector multiplication [21]. The NTT features is reduction in the time complexity for polynomials multiplication from O(n^2) into O(n log n) [22].

2.1. Crystal Kyber

Crystal-Kyber is based on the hardness of solving the module-learning with error problem (MLWE) [23]. In its specifications, Kyber has 6 algorithm processes. In some of these algorithm processes, there are encoding and compression functions, but for simplification in the writing, the functions are named toRing() to convert a bitstring into a ring element in polynomial form and fromRing() to recover the bitstring value from the ring element to facilitate reading and understanding [24]. In addition, Kyber also requires additional functions such as eXtendable output function (XOF), hashing, pseudorandom function (PRF), and a key derivation function (KDF) [25]. The six processes of the algorithm are presented in Algorithm 1 to 6.

Algorithm 1. Kyber PKE key generation

Input: $randomness\ d \in \{0,1\}^{256}$

Process:

1. Expand $(\rho, \sigma) := G(d)$

2. Generate matrix $A \in R_q^{k \times k}$ using ρ
3. Samples $S \in R_q^k$ using Centered Binomial Distribution using σ
4. Samples $e \in R_q^k$ using Centered Binomial Distribution using σ
5. Calculate $t = As + e$

Output: $Public\ Key\ pk := (t, \rho)\ and\ Private\ Key\ sk := s$

Algorithm 2. Kyber PKE encryption

Input: $Public\ Key\ pk := (t, \rho), message\ (m), random\ rings\ (r)$

Process:

1. Generate matrix $A \in R_q^{k \times k}$ using ρ
2. Samples $y \in R_q^k$ using Centered Binomial Distribution with r
3. Samples $e_1 \in R_q^k$ using Centered Binomial Distribution with r
4. Samples $e_2 \in R_q^k$ using Centered Binomial Distribution with r
5. Calculate $u = (A^T y) + e_1$
6. Calculate $v = (t^T y) + e_2 + toRing(m)$

Output: $ciphertext\ c = (u, v)$

Algorithm 3. Kyber PKE decryption

Input: $Secret\ Key\ sk := s, ciphertext\ c = (u, v)$

Process:

1. Calculate $m* = v - s^T u$
2. Calculate $m = fromRing(m*)$

Output: $message\ m$

Algorithm 4. Kyber KEM key generation

Input: $randomness\ d\ and\ randomness\ z$

Process:

1. Calculate $(pk, sk') := Kyber.PKE.KeyGen(d)$
2. Calculate $sk := (sk'||pk||H(pk)||z)$

Output: $Public\ Key\ pk\ and\ Private\ key\ sk := (sk'||pk||H(pk)||z)$

Algorithm 5. Kyber KEM encapsulation

Input: $Public\ Key\ pk$

Process:

1. Generate message $m \in \{0,1\}^{256}$
2. Calculate $(\acute{K}, r) := G(m||H(pk))$
3. Calculate $c = PKE.enc(pk, H(m))$
4. Calculate $K := KDF(\acute{K}||H(c))$,

Output: $encapsulated\ c, shared\ secret\ K$

Algorithm 6. Kyber KEM decapsulation

Input: $secret\ key\ sk\ and\ encapsulated\ c$

Process:

1. Calculate $m' = Kyber.PKE.Decryption(sk, c)$
2. Calculate $(\acute{K}', r') := G(m'||H(pk))$
3. Calculate $c' = Kyber.PKE.Encryption(pk, m', r')$
4. $if\ c = c', then\ K := KDF(K'||H(c))$
5. $if\ c \neq c', then\ K := KDF(z||H(c))$

Output: $shared\ secret\ K$

2.2. Crystal-Dilithium

Dillitihum is a digital signature scheme based on lattice modules that has been standardized by NIST [26]. Dilithium is designed to be simple to implement securely, minimize the size of public keys and signatures, modular, and conservative on parameters [14], [22]. Other than that, Dilithium is designed to be efficient to run and secure against lattice reduction attack [27]. This effectiveness is accomplished by meticulously organizing the instructions and combining the multiplications and reductions throughout the NTT [6].

The security of this algorithm is based on the difficulty of obtaining the shortest vector from a lattice [22] Dilithium are utilizing calculation functions like ExpandA, ExpandS, ExpandMask, Power2Round, HighBits, SampleInBall, MakeHint, and UseHint [22]. The simplified design of the algorithm used in Dilithium is shown in Algorithms 7 to 9.

Algorithm 7. Dilithium key generation

Input: No Input

Process:

1. Generate random $seed\ \xi\ \in\ \{0,1\}^{256}$
2. Calculate $(\rho, \rho', K)\ \in\ \{0,1\}^{256} \times \{0,1\}^{512} \times \{0,1\}^{256} :=\ H(\xi)$
3. Generate matrix $A\ \in\ R_q^{k \times l} :=\ ExpandA(\rho)$
4. Samples $(s_1, s_2) \in\ R_q^l \times\ R_q^k := ExpandS(\rho')$
5. Calculate $t\ =\ As_1\ + s_2$
6. Calculate $(t_1, t_0) :=\ Power2Round(t, d)$
7. Calculate $tr\ \in\ \{0,1\}^{256} :=\ H(\rho||t_1)$

Output: $Public\ Key\ vk :=\ (\rho, t_1)\ and\ Private\ Key\ ssk :=\ (\rho, K, tr, s_1, s_2. t_0)$

Algorithm 8. Dilithium signing

Input: $Private\ key\ ssk\ =\ (\rho, K, tr, s_1, s_2. t_0)\ and\ messages\ M\ \in\ \{0,1\}^*$

Process:

1. Generate matrix $A\ \in\ R_q^{k \times l} :=\ ExpandA(\rho)$
2. Generate $\mu\ \in\ \{0,1\}^{512} :=\ H(tr||M)$
3. Calculate $\rho''\ \in\ \{0,1\}^{512}\ =\ H(K||\mu)$
4. Initiate $\kappa := 0, and\ (z, h) :=\perp$
5. ***while*** $(z, h)\ =\perp$ ***do***
6. $y\ \in\ R_q^l :=\ ExpandMask(\rho'', \kappa)$
7. $w :=\ Ay$
8. $w_1 :=\ HighBits(w)$
9. $\bar{c} := H(\mu||w_1)$
10. $c\ \in\ R_q :=\ SampleInBall(c)$
11. $z :=\ y\ + cs_1$
12. $r_0 :=\ LowBits_q(w - cs_2, 2_{\gamma 2})$
13. ***if*** $||z||_\infty\ \geq\ \gamma_1\ \ -\ \beta\ or\ ||r_0||\ \geq \gamma_2 - \beta$
14. ***then*** $(z, h)\ =\perp$
15. ***else***
16. $h := MakeHint(-ct_0, w - cs_2 + ct_0, 2_{\gamma 2})$
17. ***if*** $||ct_0||_\infty\ \geq\ \gamma_2\ \ or\ the\ number\ of\ 1's\ in\ h\ i\ greater\ than\ \omega$
18. ***then*** $(z, h)\ =\perp$
19. $\kappa :=\ \kappa\ + l$

Output: $Signature\ \boldsymbol{\sigma}: = (z, h, \bar{c})$

Algorithm 9. Dilithium verification

Input: $Public\ Key\ vk\ =\ (\rho, t_1)\ , message\ M, and\ signature\ \sigma :=\ (z, h, \bar{c}\ )$

Process:

1. Generate matrix $A\ \in\ R_q^{k \times l} :=\ ExpandA(\rho)$
2. Generate $\mu\ \in\ \{0,1\}^{512} := H(H(\rho||t_1)||M)$
3. Calculate $c := SampleInBall(\bar{c}\ )$
4. $w'_1 :=\ UseHint(h, Az - ct_1 \cdot 2^d, 2\gamma_2)$
5. $\bar{c}\ ' := H(\mu||w'_1)$
6. ***if*** $\bar{c}\ '\ =\ \bar{c}\ and\ ||z||_\infty\ <\ \gamma_1 \beta\ and\ the\ number\ of\ 1's\ in\ h\ \leq \omega$
7. $VerificationStatus\ =\ Valid$
8. ***else***
9. $VerificationStatus\ =\ Valid$

Output: $VerificationStatus$

3. METHOD

The following schematic diagram illustrates the overall workflow of the research process, which consists of five main stages: system design and planning, system implementation, system testing, performance measurement, and performance analysis, all of which are essential to ensure the development of a reliable and effective system.

3.1. System design and planning

The standard NIST post-quantum algorithms, Kyber and Dilithium have been implemented. Pre-quantum algorithms like RSA, ECDSA, and ECDH are utilized for comparison. The Brainpool curve is used by both ECDH and ECDSA, with ECDH serving as the key agreement rather than KEM.

Several equivalent or nearly equivalent security levels are used to obtain a comparable comparison. Here is the algorithm strength table based on security levels for RSA and elliptic curve cryptography (ECC) [23] and Kyber and Dilithium [28], [29]. It can be depicted in Table 1.

Table 1. Algorithm security level

No	Security bits	RSA	ECDH/ECDSA	Kyber	Dilithium
1	128	3072	256≤f≤383	Kyber512	Dilithium2
2	192	7680	384≤f≤511	Kyber768	Dilithium3
3	256	15360	f≥512	Kyber1024	Dilithium5

Algorithm variants are denoted using the format keyExchange_digitalSignature_level. For example, Kyber_Dilithium_1 refers to the post-quantum algorithm variant operating at the first security level. The proposed system is organized into two main stages: i) digital signature key pair generation, which is used to show and represent participant identities and ii) execution of the AKE protocol, during which authentication and secure session key establishment are performed.

a. Digital signature key generation

The process of generating the digital signature will be carried out according to the diagram in Figure 1.

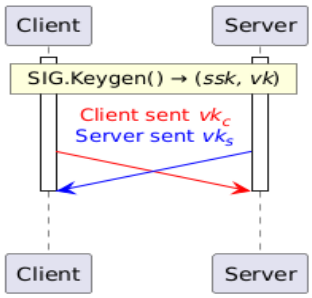

Figure 1. Digital signature key generation and exchange

Each party will generate a key pair (vk, ssk), which is obtained by running Algorithm 7. Next, the client will exchange public keys with the server. In the TLS protocol, public keys between parties are distributed by involving a trusted third party using certificates obtained from the public key infrastructure (PKI) [30]. However, in this study, both parties exchanged keys directly through the API.

b. AKE protocol execution

The AKE protocol consists of three stages: keygen-sign, verify-encapsulate-sign, and verify-decapsulate, as shown in Figure 2.

The AKE protocol begins when the client generates a KEM public–private key pair (pk, sk). The client then signs the public key (pk) using its digital signature private key (ssk_c) producing a signature σ_A. The pair (pk, σ_A) is terminated to the server for verification. If the verification fails, the server returns an error and the protocol is terminated.

Upon successful verification, the server generates a shared secret key K and encapsulates it into a ciphertext c. The server signs c with its digital signature private key to produce a second signature σ_B and sends (c, σ_B) back to the client. The client verifies σ_B if the verification is invalid, the protocol terminates with an error. Otherwise, the client decapsulates c to recover the shared secret key K.

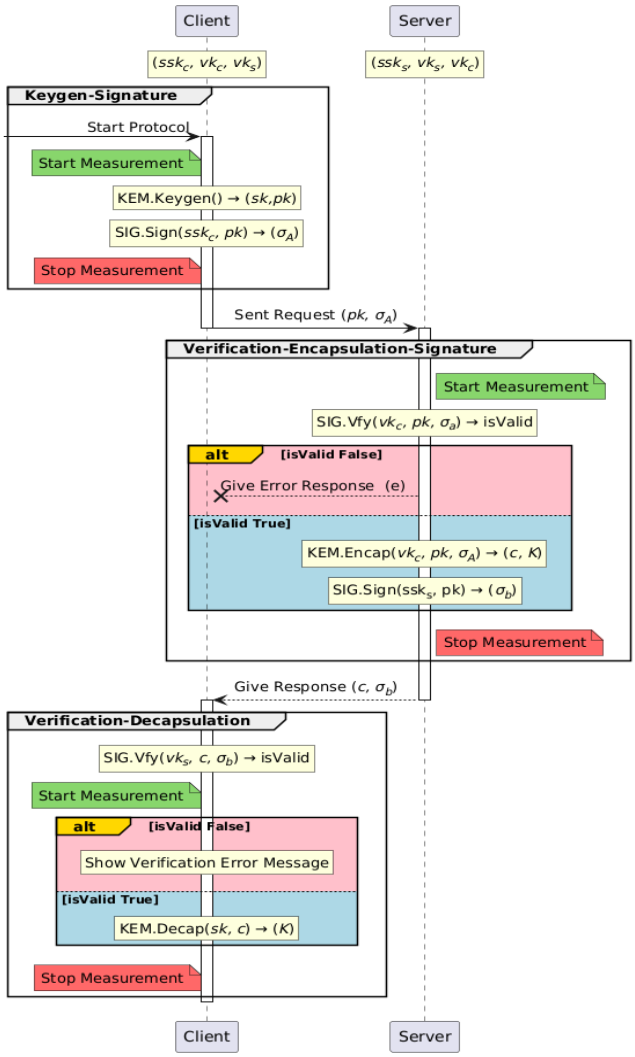

Figure 2. AKE protocol

3.2. System implementation

The proposed AKE system was implemented using Python 3.12 on a Linux-based platform. The cryptographic operations for CRYSTALS-Kyber and CRYSTALS-Dilithium were implemented using a post-quantum cryptographic library compliant with NIST specifications. The system architecture follows a client–server model, where FastAPI is used to expose RESTful HTTP endpoints for key exchange and authentication operations.

Each phase of the AKE protocol—key generation, signing, verification, encapsulation, and decapsulation—is executed through dedicated API endpoints. The entire execution workflow is automated using Bash scripts, which sequentially invoke the API endpoints to ensure consistent execution order and repeatability. This design enables the isolated measurement of each cryptographic operation, as well as the end-to-end execution of AKE.

3.3. System testing

System testing was conducted in two stages. First, functional testing verified the correctness of the digital signature and the KEM key pair generation, storage, and loading processes. Second, integration testing evaluated the complete execution of the AKE protocol, encompassing authentication, key exchange, and the establishment of a shared secret between communicating parties.

All experiments were performed on a test environment with fixed hardware and software configurations to ensure consistency. The testing environment consisted of a single machine running the client and server components locally to eliminate network latency effects. The system was evaluated under identical conditions for all algorithm variants and security levels (128, 192, and 256 bits).

3.4. Performance measurement and analysis

Performance evaluation focused on execution time and communication overhead. Measurements were conducted for the following stages:

a. Digital signature key pair generation,
b. KEM key pair generation and public key signing (KeyGen–Sign),
c. Public key verification, shared secret generation, and encapsulation signing (verify–encapsulation–sign),
d. Verification and decapsulation of encapsulated data (verify–decapsulation),
e. End-to-end execution of the AKE protocol,
f. Output the byte size of cryptographic messages.

Each experiment was executed multiple times (N iterations), and the reported execution time represents the average value across all runs to minimize measurement variance. Execution time was measured using high-resolution system timers, while communication overhead was calculated by recording the byte size of all transmitted cryptographic artifacts. The collected results were analyzed comparatively across algorithm variants and security levels to assess performance trade-offs.

4. RESULTS AND DISCUSSION

This section presents the experimental results and discusses the performance of the proposed AKE protocol using the Kyber–Dilithium post-quantum cryptographic variant in comparison with classical ECDH–ECDSA and ECDH–RSA schemes. The evaluation focuses on execution time and output byte size across three security levels: 128, 192, and 256 bits.

4.1. Key encapsulation mechanism and digital signature key pair generation

Dilithium consistently achieves the fastest key-generation performance among the evaluated algorithms, requiring only 0.0245–0.0599 ms across all security levels. In comparison, ECDSA operates in the sub-second range. In contrast, RSA exhibits substantially higher latency, increasing from approximately 130 ms at the lowest security level to nearly 50 s at the highest, reflecting severe scalability limitations. These results underscore the significant computational overhead of RSA relative to elliptic curve–based ECDSA and lattice-based Dilithium. At the same time, Dilithium's consistently low latency highlights its superior efficiency and scalability as a practical post-quantum alternative to conventional public-key cryptography, as summarized in Table 2.

4.2. Key generation and sign phase

The post-quantum Kyber–Dilithium hybrid consistently achieves the highest efficiency in combined key-generation and signing operations, with execution times remaining below 0.17 ms and increasing only modestly from 0.0896 ms at security level 1 to 0.1696 ms at level 3. By comparison, the ECDH–ECDSA hybrid operates in the sub-3 ms range, whereas the ECDH–RSA combination exhibits severe scalability limitations, with latency escalating from 1.0843 ms to 178.0964 ms at the highest security level. At level 3, ECDH–RSA is more than three orders of magnitude slower than Kyber–Dilithium, highlighting the impracticality of RSA-based hybrids for high-security, performance-sensitive environments. These findings demonstrate that post-quantum hybrid schemes can deliver both long-term security and substantial computational advantages, as summarized in Table 3.

Table 2. Key generation average time

Algorithm	Average time (ms)
Dilithium_1	0.0245
Dilithium_2	0.0377
Dilithium_3	0.0599
ECDSA_1	0.2738
ECDSA_2	0.6302
ECDSA_3	0.8998
RSA_1	130.0487
RSA_2	4980.0524
RSA_3	49936.7783

Table 3. Keygen-sign average time

Algorithm	Average time (ms)
Kyber_Dilithium_1	0.0896
Kyber_Dilithium_2	0.1435
Kyber_Dilithium_3	0.1696
ECDH_ECDSA_1	0.7249
ECDH_ECDSA_2	1.536
ECDH_ECDSA_3	2.445
ECDH_RSA_1	1.0843
ECDH_RSA_2	35.3117
ECDH_RSA_3	178.0964

4.3. Verification-encapsulation-signature

The post-quantum Kyber_Dilithium hybrid consistently achieves the best performance among the evaluated schemes, requiring only 0.13 ms at the lowest security level—approximately an order of magnitude faster than the classical alternatives—and exhibiting strong scalability, with execution time increasing only modestly as security requirements rise. In comparison, ECDH_ECDSA shows predictable and moderate

performance degradation, whereas ECDH_RSA suffers from severe scalability limitations, becoming nearly 700 times slower than Kyber_Dilithium at the highest security level. These results highlight the substantial computational overhead of RSA-based hybrids at elevated security settings and position Kyber_Dilithium as an efficient, scalable, and future-ready cryptographic solution, as summarized in Table 4.

4.4. Verification and decapsulation phase

The Kyber_Dilithium hybrid demonstrates the fastest combined verification and decapsulation performance among the evaluated schemes, with execution times remaining below 0.08 ms and scaling smoothly from 0.0398 ms at security level 1 to 0.0786 ms at level 3. In contrast, the classical ECDH_RSA and ECDH_ECDSA hybrids operate in the millisecond range, with 0.3814–1.961 ms and 0.6007–1.99 ms, respectively, and both converge to nearly identical performance at the highest security level. Overall, Kyber_Dilithium is approximately 15 times faster than the closest classical alternative at level 1 and about 25 times faster at level 3, underscoring the superior efficiency and scalability of post-quantum hybrid schemes for verification and decapsulation operations, as summarized in Table 5.

Table 4. Verification-encapsulation-signature average time

Algorithm	Average time (ms)
Kyber_Dilithium_1	0.1344
Kyber_Dilithium_2	0.2538
Kyber_Dilithium_3	0.2635
ECDH_ECDSA_1	1.4742
ECDH_ECDSA_2	2.9092
ECDH_ECDSA_3	4.7692
ECDH_RSA_1	1.5314
ECDH_RSA_2	35.323
ECDH_RSA_3	182.889

Table 5. Verification-decapsulation average time

Algorithm	Average time (ms)
Kyber_Dilithium_1	0.0398
Kyber_Dilithium_2	0.0574
Kyber_Dilithium_3	0.0786
ECDH_ECDSA_1	0.6007
ECDH_ECDSA_2	1.3052
ECDH_ECDSA_3	1.99
ECDH_RSA_1	0.3814
ECDH_RSA_2	0.9451
ECDH_RSA_3	1.961

4.5. Overall authenticated key exchange protocol performance

Figure 3 compares the average runtime of four different hybrid cryptographic AKE protocols. The combinations tested are Kyber_Dilithium, ECDH_ECDSA, ECDH_RSA, and a standalone Kyber_Dilithium structure labeled "10.76" through "355.00," which represent the average time in milliseconds for each protocol to complete. The data shows that the Kyber_Dilithium hybrid protocol is the most computationally expensive, taking 393.68 ms, followed closely by the Kyber-only variant at 271.83 ms. In contrast, the classical ECDH_ECDSA and ECDH_RSA hybrids are significantly faster, with average times of 52.47 ms and 20.83 ms, respectively.

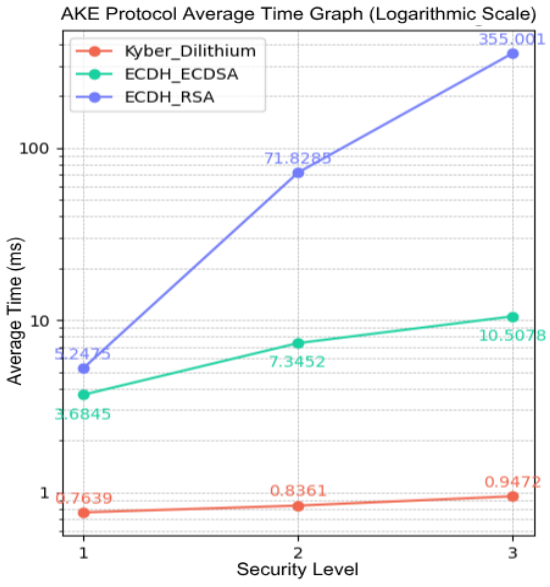

Figure 3. AKE protocol average time

The performance gap highlights the substantial computational overhead introduced by post-quantum cryptographic algorithms compared to their classical elliptic curve counterparts. While the Kyber_Dilithium combination offers strong security against both classical and quantum threats, its runtime is nearly an order of magnitude slower than ECDH_RSA. This suggests that for performance-sensitive applications, a hybrid approach using ECDH may still be preferable. However, as quantum computing advances, adopting slower but quantum-resistant algorithms like Kyber and Dilithium will likely become necessary, despite current speed trade-offs.

4.6. Output by the size analysis

Figures 4 and 5 compare the output byte sizes of key pairs, ciphertexts, and digital signatures. As expected, the Kyber KEM produces significantly larger public keys, secret keys, and ciphertexts than ECDH, with sizes approximately an order of magnitude larger. Similarly, Dilithium signatures and keys are larger than those of ECDSA, though they grow more moderately than RSA signatures at higher security levels.

Overall, the Kyber–Dilithium variant incurs higher communication overhead than classical schemes, particularly when compared to ECDH–ECDSA. However, when compared to ECDH–RSA, the Dilithium secret signing key size at higher security levels is smaller, while other components remain larger. These results indicate a clear trade-off between communication cost and computational efficiency.

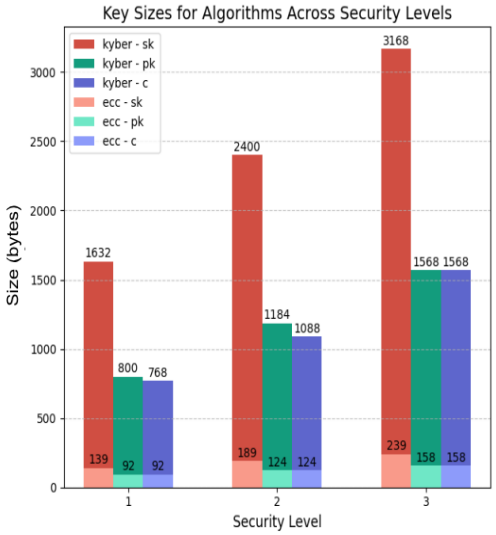

Figure 4. KEM algorithm bytes size

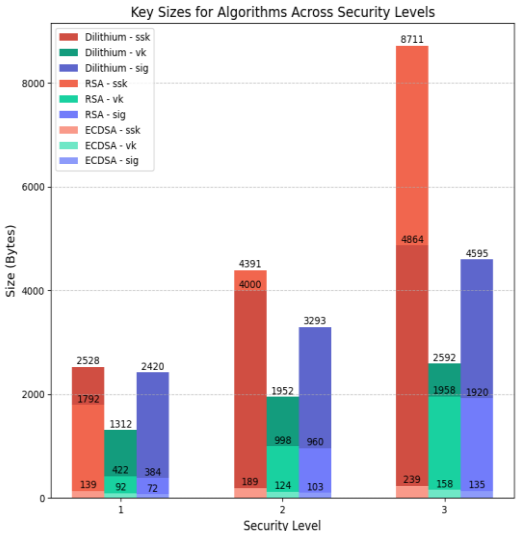

Figure 5. Digital signatures bytes size

4.7. Discussion and pratical implementation

The experimental results reveal a fundamental trade-off between execution time and communication overhead. While the Kyber–Dilithium variant introduces larger message sizes, it significantly reduces execution time and scales efficiently across higher security levels. This characteristic is particularly advantageous for post-quantum TLS deployment, where handshake latency is a critical factor.

Compared with recent post-quantum TLS and AKE studies, these findings align with prior observations that lattice-based schemes increase bandwidth usage but offer superior computational performance. Unlike previous works that focus solely on KEM integration or retain classical authentication mechanisms, this study evaluates a fully post-quantum AKE protocol by combining Kyber for key encapsulation and Dilithium for authentication. Consequently, the results provide empirical evidence supporting the feasibility of deploying a complete post-quantum AKE design in future secure communication systems.

5. CONCLUSION

Experimental results show that the Kyber–Dilithium variant introduces higher communication overhead in terms of byte size compared to classical ECDH–ECDSA and ECDH–RSA schemes; however, it consistently achieves lower average execution time across security levels of 128, 192, and 256 bits. These findings demonstrate the feasibility and practical efficiency of integrating NIST-standardized post-quantum algorithms for both key encapsulation and authentication within an AKE protocol. While the evaluation is

limited to execution time and message size in a software-based environment, the results indicate that Kyber–Dilithium is a promising alternative for post-quantum secure key exchange. Future work will focus on deployment in TLS-based systems, evaluation on resource-constrained devices, and broader security and performance analyses.

ACKNOWLEDGMENTS

The authors would like to sincerely thank Universitas Sebelas Maret for the continuous support and research opportunities. Special thanks are also extended to the research team for their valuable contributions, dedication, and collaboration throughout this.

FUNDING INFORMATION

This work was supported by the Non-State Budget (Non-APBN) Research Grant of Universitas Sebelas Maret under the Fundamental Research Scheme for the 2025 Fiscal Year, Grant Number 369/UN27.22/PT.01.03/2025, dated March 24, 2025.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Bambang Harjito	✓	✓		✓	✓	✓		✓	✓			✓	✓	✓
Muhammad Defaroyan	✓	✓	✓		✓			✓	✓					
Fajar Muslim		✓	✓		✓				✓			✓		
Ery Permana Yudha		✓				✓				✓			✓	
Endra Pratama		✓				✓				✓		✓		

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

DATA AVAILABILITY

No new data were generated or analyzed during this study; therefore, data sharing is not applicable to this article.

REFERENCES

[1] H. Lee, D. Kim, and Y. Kwon, "TLS 1.3 in practice: How TLS 1.3 contributes to the internet," in *Proceedings of the Web Conference 2021*, 2021, pp. 70–79, doi: 10.1145/3442381.34500.

[2] K. Souvatzidaki and K. Limniotis, "Post-Quantum Key Exchange in TLS 1.3: Further Analysis on Performance of New Cryptographic Standards," *Cryptography*, vol. 9, no. 4, p. 73, Nov. 2025, doi: 10.3390/cryptography9040073.

[3] N. Alnahawi, J. Müller, J. Oupický, and A. Wiesmaier, "A Comprehensive Survey on Post-Quantum TLS," *IACR Communications in Cryptology*, Jul. 2024, doi: 10.62056/ahee0iuc.

[4] R. Gonzalez and T. Wiggers, "KEMTLS vs. Post-quantum TLS: Performance on Embedded Systems," in *International Conference on Security, Privacy, and Applied Cryptography Engineering*, Cham: Springer Nature Switzerland, 2022, 2022, pp. 99–117, doi: 10.1007/978-3-031-22829-2_6.

[5] C. Bader, D. Hofheinz, T. Jager, E. Kiltz, and Y. Li, "Tightly-secure authenticated key exchange," in *Theory of Cryptography Conference*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 629–658, doi: 10.1007/978-3-662-46494-6_26.

[6] S. Han *et al.*, "Authenticated Key Exchange and Signatures with Tight Security in the Standard Model," in *Annual International Cryptology Conference*, Cham: Springer International Publishing, 2021, pp. 670–700, doi: 10.1007/978-3-030-84259-8_23.

[7] R. Niederhagen and M. Waidner, "Practical Post-Quantum Cryptography," *Fraunhofer Institute for Secure Information Technology SIT*, 2017, Available: http://publica.fraunhofer.de/documents/N-481797.html.

[8] M. Abbasi, F. Cardoso, P. Váz, J. Silva, and P. Martins, "A Practical Performance Benchmark of Post-Quantum Cryptography Across Heterogeneous Computing Environments," *Cryptography*, vol. 9, no. 2, pp. 1–27, May 2025, doi: 10.3390/cryptography9020032.
[9] Q. D. Truong, P. N. Duong, and H. Lee, "Efficient Low-Latency Hardware Architecture for Module-Lattice-Based Digital Signature Standard," *IEEE Access*, vol. 12, pp. 32395–32407, 2024, doi: 10.1109/ACCESS.2024.3370470.
[10] D. Sikeridis, P. Kampanakis, and M. Devetsikiotis, "Post-Quantum Authentication in TLS 1.3: A Performance Study," *Cryptology ePrint Archive*, 2020, doi: 10.14722/ndss.2020.24203.
[11] A. Zafar and S. S. Iqbal, "Integrating code-based post-quantum cryptography into SSL TLS protocols through an interoperable hybrid framework," *Discover Computing*, vol. 28, no. 1, p. 202, Sep. 2025, doi: 10.1007/s10791-025-09735-7.
[12] 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, doi: 10.1109/ACCESS.2020.3013250.
[13] M. Barbosa *et al.*, "EasyPQC: Verifying Post-Quantum Cryptography," *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2564–2586, 2021, doi: 10.1145/3460120.3484567.
[14] E. D. Demir, B. Bilgin, and M. C. Onbasli, "Performance Analysis and Industry Deployment of Post-Quantum Cryptography Algorithms," *arXiv preprint*, 2025, doi: 10.48550/arXiv.2503.12952.
[15] J. Pan, B. Wagner, and R. Zeng, "Lattice-Based Authenticated Key Exchange with Tight Security," in *Annual International Cryptology Conference*, Cham: Springer Nature Switzerland, 2023, pp. 616–647, doi: 10.1007/978-3-031-38554-4_20.
[16] J. Bos *et al.*, "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM," in *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, IEEE, Apr. 2018, pp. 353–367, doi: 10.1109/EuroSP.2018.00032.
[17] A. Astarloa, J. Lázaro, and J. I. Gárate, "CRYSTALS-Dilithium post-quantum cyber-secure SoC for wired communications in critical systems," *Internet of Things (The Netherlands)*, vol. 33, p. 101656, Sep. 2025, doi: 10.1016/j.iot.2025.101656.
[18] A. Scrivano, "A Comparative Study of Classical and Post-Quantum Cryptographic Algorithms in the Era of Quantum Computing," *arXiv preprint*, 2025, doi: 10.48550/arXiv.2508.00832.
[19] L. Ducas, T. Lepoint, V Lyubashevsky, P. Schwabe, G. Seiler, and D. Stehle, "Crystals–dilithium: Digital signatures from module lattices," *IACR Transactions on Cryptographic Hardware and Embedded Systems (TCHES)*, no. 1, pp. 238–268, 2018, doi: 10.13154/tches.v2018.i1.238-268.
[20] M. Almutairi and F. T. Sheldon, "Resilience of Post-Quantum Cryptography in Lightweight IoT Protocols: A Systematic Review," *Eng*, vol. 6, no. 12, p. 346, Dec. 2025, doi: 10.3390/eng6120346.
[21] T. H. Nguyen, B. Kieu-Do-Nguyen, C. K. Pham, and T. T. Hoang, "High-Speed NTT Accelerator for CRYSTAL-Kyber and CRYSTAL-Dilithium," *IEEE Access*, vol. 12, pp. 34918–34930, 2024, doi: 10.1109/ACCESS.2024.3371581.
[22] S. Bai *et al.*, "CRYSTALS-Dilithium Algorithm Specifications and Supporting Documentation," *NIST Proposal*, pp. 1–38, 2021, Available: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210208.pdf.
[23] Y. Xing and S. Li, "A compact hardware implementation of cca-secure key exchange mechanism crystals-kyber on fpga," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, no. 2, pp. 328–356, Feb. 2021, doi: 10.46586/tches.v2021.i2.328-356.
[24] M. Richter, M. Bertram, J. Seidensticker, and A. Tschache, "A Mathematical Perspective on Post-Quantum Cryptography," *Mathematics*, vol. 10, no. 15, pp. 1–33, Jul. 2022, doi: 10.3390/math10152579.
[25] R. Avanzi *et al.*, "CRYSTALS-KYBER Algorithm Specifications and Supporting Documentation (Version 3.2)," *NIST PQC Round*, vol. 2, no. 4, pp. 1–43, 2021, Available: https://www.pq-crystals.org/kyber/data/kyber-specification-round3-20210804.pdf.
[26] R. Avanzi *et al.*, *Module-Lattice-Based Digital Signature Standard*, NIST FIPS PUB 204, National Institute of Standards and Technology, Gaithersburg, MD, USA, Aug. 2024, doi: 10.6028/NIST.FIPS.204.
[27] E. Alkim, J. W. Bos, L. Ducas, and T. Lepoint, "Post-Quantum Key Exchange—A New Hope," in *25th USENIX security symposium (USENIX Security 16)*, 2016, pp. 327–343.
[28] T. M. Fernandez-Carames, "From Pre-Quantum to Post-Quantum IoT Security: A Survey on Quantum-Resistant Cryptosystems for the Internet of Things," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6457–6480, Jul. 2020, doi: 10.1109/JIOT.2019.2958788.
[29] F. R. Ghashghaei, Y. Ahmed, N. Elmrabit, and M. Yousefi, "Enhancing the Security of Classical Communication with Post-Quantum Authenticated-Encryption Schemes for the Quantum Key Distribution," *Computers*, vol. 13, no. 7, pp. 1–25, Jul. 2024, doi: 10.3390/computers13070163.
[30] K. Gjøsteen and T. Jager, "Practical and tightly-secure digital signatures and authenticated key exchange," in *Annual International Cryptology Conference*, vol. 10992 LNCS, Cham: Springer International Publishing, 2018, pp. 95–125, doi: 10.1007/978-3-319-96881-0_4.

BIOGRAPHIES OF AUTHORS

Bambang Harjito is a professor at lecturer in the Department of Informatics, Faculty of Information Technology and Data Science, Universitas Sebelas Maret, Surakarta, Indonesia. He earned his Master's degree in Computer Science from James Cook University in 2000 and his Ph.D. in Information Systems from Curtin University, Perth, Australia, in 2014. His research focuses on data protection, privacy protection, information hiding, cryptography, and cybersecurity. He is also an active member of the Association of Indonesian Computer and Informatics Colleges (APTIKOM). He can be contacted at email: bambang_harjito@staff.uns.ac.id.

Muhammad Defaroyan is a Cloud Engineer in PT Cloud Ace Integra, South Jakarta, Indonesia. Before that he worked as a Technical Support Engineer at Domainesia, Yogyakarta, Indonesia. He earned his bachelor's degree in Informatics from the Sebelas Maret University in 2025. His research focuses on cryptography, cybersecurity, distributed systems, and cloud computing. He can be contacted at email: defaroyan@gmail.com.

Fajar Muslim is a Lecturer in the Department of Informatics, Faculty of Information Technology and Data Science, Sebelas Maret University, Surakarta, Indonesia. He earned his Bachelor's degree in Informatics Engineering from the Bandung Institute of Technology in 2021 and his Master's degree in Informatics from the Bandung Institute of Technology in 2022. His research focuses on text processing utilizing machine learning. He can be contacted at email: fajar.muslim@staff.uns.ac.id.

Ery Permana Yudha is a Lecturer in the Department of Informatics, Faculty of Information Technology and Data Science, Sebelas Maret University, Surakarta, Indonesia. He earned his Bachelor of Computer Science degree from the Sepuluh November Institute of Technology in 2018 and his Master of Computer Science degree from the Sepuluh November Institute of Technology in 2021. His research focuses on image processing utilizing machine learning. He can be contacted at email: erypermana@staff.uns.ac.id.

Endra Pratama is a Lecturer in the Department of Informatics, Faculty of Information Technology and Data Science, Sebelas Maret University, Surakarta, Indonesia. He earned his Bachelor's degree in Mathematics from Sebelas Maret University in 2016 and his Master's degree in Computer Science from Gadjah Mada University in 2019. His research focuses on data utilization for business domains. He can be contacted at email: endra.p@staff.uns.ac.id.