Bulletin of Electrical Engineering and Informatics
Vol. 15, No. 6, December 2026, pp. 5626~5635
ISSN: 2302-9285, DOI: 10.11591/eei.v15i6.13736

Implementation of a secure digital data exchange framework based on post-quantum cryptography

Nhu-Quynh Luc[1], Duc-Huy Quach[2], Thanh-Long Le-Nguyen[3], Duy-Phuong Nguyen[1]
[1]Academy of Cryptography Techniques, Hanoi, Vietnam
[2]Center 3, Command 86, Danang, Vietnam
[3]Headquarters of Staff, 3rd Military Region, Hai Phong, Vietnam

Article Info

Article history:

Received Jun 19, 2025
Revised Sep 15, 2026
Accepted Oct 4, 2026

Keywords:

Advanced encryption standard-256
Digital data security
Falcon-1024
Kyber-1024
Quantum cryptography
Secure data exchange

ABSTRACT

Quantum computing threatens Rivest-Shamir-Adleman (RSA)- and elliptic-curve-based data exchange, motivating practical post-quantum cryptography (PQC). This study implements a cross-platform file-exchange framework using Kyber-1024 for key encapsulation mechanism (KEM)-based session-key establishment, Falcon-1024 detached signatures, and advanced encryption standard (AES)-256-cipher block chaining mode (CBC) for bulk encryption. Evaluation covered an IEEE 802.11ac Linux/Windows testbed (1,000–40,365 KiB) and controlled single-host transmission control protocol (TCP)-loopback runs up to 102,400 KiB. The three reported Linux-to-Linux wireless session-key-establishment times averaged 25.26 ms (approximately 25.3 ms); encrypting 4,352 KiB in 29.8 ms yielded 142.6 MiB/s. Inspection of captured packets revealed no plaintext file content or session-key material in the application payloads, and a forged file/signature pair was rejected. These results support implementation feasibility on commodity hardware within the tested workflow, but they do not establish authenticated key exchange, active man-in-the-middle (MITM) resistance, metadata confidentiality, or traffic-analysis resistance.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Nhu-Quynh Luc
Academy of Cryptography Techniques
141 Chien Thang, Thanh Liet, Hanoi, Vietnam
Email: quynhln@actvn.edu.vn

1. INTRODUCTION

The rapid progress of quantum computing has introduced fundamental challenges to classical public-key cryptosystems, which are based on the presumed hardness of problems such as integer factorization—utilized in Rivest-Shamir-Adleman (RSA) and the elliptic curve discrete logarithm problem, which underlies elliptic curve cryptography (ECC) [1], [2]. Large-scale quantum computers would alter these security assumptions because Shor's algorithm can solve the underlying integer-factorization and discrete-logarithm problems in polynomial time [3]. Grover's algorithm also reduces the effective search margin of symmetric ciphers [4]–[6]. For information that must remain confidential for many years, this creates a harvest-now-decrypt-later risk: data intercepted today may be stored and decrypted once cryptographically relevant quantum computers become available [7].

To proactively respond to this emerging threat, the global cryptographic research community, led in particular by the U.S. National Institute of Standards and Technology (NIST), launched the post-quantum cryptography (PQC) standardization program. NIST finalized module-lattice-based key-encapsulation mechanism (ML-KEM) in Federal Information Processing Standards (FIPS) 203 and module-lattice-based digital signature algorithm (ML-DSA) in FIPS 204 in 2024; ML-KEM is derived from CRYSTALS-Kyber,

while Falcon remains on the standardization track toward fast-Fourier transform over NTRU-lattice-based digital signature algorithm (FN-DSA) [8]–[12]. The migration to PQC is therefore not only a matter of replacing one primitive with another. A practical application must account for key establishment, origin and integrity verification, bulk-data encryption, key management, and protocol-level authentication without assigning security properties to the composition that the individual primitives do not provide. In the framework evaluated here, Kyber-1024 establishes a per-session shared secret, Falcon-1024 provides a detached signature over the transferred file, and advanced encryption standard (AES)-256-cipher block chaining mode (CBC) protects the file payload.

Previous work has explored post-quantum key exchange and the evaluation of individual PQC primitives [13], [14], but comprehensive studies that integrate Kyber and Falcon within a complete data-exchange application remain limited. Related work also covers hybrid key exchange, protocol integration, accelerated stacks, constrained devices, and heterogeneous benchmarking [15]–[27]. These studies are important for understanding primitive and protocol costs, but their platforms and measurement scopes differ substantially. Comparatively fewer studies document the complete behavior of a desktop file-exchange application that combines post-quantum key establishment, post-quantum signatures, symmetric bulk encryption, graphical operator interaction, and heterogeneous Linux/Windows endpoints while also stating the limits of the resulting security claims.

Addressing this gap, the present work proposes a secure digital data exchange framework built upon PQC. The implementation combines Kyber-1024 key-encapsulation mechanism (KEM)-based session-key establishment, Falcon-1024 detached file signatures, and AES-256-CBC encryption in a Linux/Windows application. The study evaluates application-level latency, encryption throughput, file-signing and verification time, cross-platform transmission behavior, and two implementation-level security checks. It also separates file verification from authenticated key exchange so that the claims remain consistent with the implemented protocol.

The remainder of this paper is organized as follows. Section 2 combines the cryptographic design rationale, related technical context, end-to-end algorithm, implementation environment, and operational interfaces of the proposed framework. Section 3 presents the experimental results and discusses performance, comparative scope, security observations, limitations, and future development directions. Section 4 concludes the study and summarizes the principal findings and remaining work.

2. SYSTEM DESIGN AND IMPLEMENTATION

2.1. Cryptographic design rationale and security boundary

Kyber represents a module-lattice–based KEM whose security guarantees are mathematically anchored in the computational hardness of the module learning with errors (MLWE) problem [10]. NIST's FIPS 203 standardizes ML-KEM, which is derived from Kyber, while FIPS 204 standardizes ML-DSA [8], [11], [12]. Falcon, also lattice-based, is built upon the NTRU framework and is distinguished by its strong security proofs and compact signature size, making it well-suited for applications where bandwidth and memory are constrained [9]. Falcon is expected to be standardized separately as FN-DSA [8]. The present implementation retains Kyber-1024 and Falcon-1024 because the objective is to evaluate the existing application pipeline rather than to claim direct conformance between the pre-standardized algorithm names and the finalized NIST standards.

A central design consideration is the distinction between successful KEM execution and authenticated key exchange. Hybrid KEM and TLS/SSH research shows that a secure primitive does not, by itself, authenticate the peer or the complete protocol transcript [15], [16], [28], [29]. In the present framework, the receiver provides a Kyber encapsulation key ek_R; the sender computes (ct_K, K_S)=Encaps(ek_R), and the receiver derives K_R=Decaps(dk_R, ct_K). Successful execution gives K_S=K_R, but neither ek_R nor the Kyber transcript is cryptographically bound to the Falcon identity. Accordingly, the workflow is described as KEM-based session-key establishment rather than authenticated key exchange.

In the implemented workflow, Falcon-1024 is used independently to verify the recovered file. The sender computes a detached signature S=Falcon.Sign(sk_S, F), and the receiver verifies S over the recovered file F′ using the configured Falcon public key pk_S. This provides file-origin and integrity verification relative to the configured key, but it does not authenticate the Kyber encapsulation key or provide protocol-wide non-repudiation without a documented trust, revocation, and key-governance process. Symmetric encryption remains indispensable for safeguarding bulk payload confidentiality. Unlike public-key cryptosystems, symmetric cryptography retains a substantial security margin against generic quantum search attacks. Grover's asymptotic search algorithm reduces the effective security margin of symmetric primitives by a square-root factor, mapping the bit-security of a 256-bit key to a classically equivalent 128-bit threshold [4]. AES-256-CBC is therefore used for bulk confidentiality with a fresh unpredictable 16-byte initialization

vector (IV) and public-key cryptography standards #7 (PKCS#7) padding. CBC provides no built-in integrity, which is an important limitation of the current composition and motivates migration to authenticated encryption in future work [6].

Prior implementations of quantum-resistant paradigms, such as NewHope variants [13] or protocol-oriented post-quantum key-exchange prototypes [15], have frequently optimized singular primitives in isolation. The present composition differs from primitive-only benchmarking because the measured path includes application logic, file handling, cryptographic-library calls, socket transfer, and graphical workflow overhead. It also differs from transport layer security (TLS)/QUIC integrations because the application defines its own transfer envelope and local key configuration rather than inheriting an authenticated channel from a mature transport protocol. For this reason, comparisons with accelerated, constrained-device, and protocol-level studies [17]–[26] are used only to place the measurements in context, not as direct speed rankings.

2.2. Experimental environment and end-to-end workflow

To evaluate the operational viability of the proposed framework in a heterogeneous network environment, the system was deployed across two distinct physical endpoints connected via an IEEE 802.11ac wireless routing interface. The experimental infrastructure isolates the workstation roles into a Linux-based source node and a Windows-based destination node. The original testbed used Ubuntu 22.04 and Windows 10 endpoints on Intel Core i5-9300H hardware. Its archived measurements cover 1,000–40,365 KiB files and are kept separate from the controlled same-host transmission control protocol (TCP)-loopback re-verification data.

The loopback environment used Windows 10 build 19045 with Python 3.14.5 and Ubuntu 26.04 long-term support (LTS) under Windows Subsystem for Linux 2 (WSL2) kernel 6.18.33.2-microsoft-standard-WSL2 with Python 3.14.4 on an Intel Core Ultra 7 155U. PQClean reference non-Advanced Vector Extensions 2 (AVX2) code was compiled through Python C Foreign Function Interface (cffi 2.0) using Microsoft Visual C++ (MSVC)/O2 on Windows and GNU Compiler Collection (GCC)-O3-std=c99 on Linux. AES-CBC used PyCryptodome 3.23.0 with PKCS#7 padding, fresh 16-byte IVs, and the default use_aesni=True setting on an advanced encryption standard new instructions (AES-NI)-capable processor. The graphical application used PyQt5 5.15.11 and SQLite 3. Each loopback operating-system/file-size configuration was executed independently 10 times, and the reported loopback values are arithmetic means.

Algorithm 1 expresses the complete sender/receiver sequence in input-output form. Presenting the workflow as an algorithm makes the order of key establishment, signing, encryption, transmission, decryption, and verification explicit while also showing the point at which the current authentication boundary arises.

Algorithm 1. End-to-end secure file-exchange workflow
Input: file F; recipient R; receiver Kyber key material (ek_R, dk_R) or KeyGen capability; sender Falcon signing key sk_S; configured Falcon verification key pk_S.
Output: verified file F′ stored at the receiver, or rejection.
Step 1. Load the recipient/contact metadata and configured Falcon verification key; the sender selects recipient R and file F.
Step 2. The receiver loads or generates the Kyber key pair (ek_R, dk_R) and makes ek_R available to the sender.
Step 3. The sender computes (ct_K, K_S) = Encaps(ek_R). The receiver computes K_R = Decaps(dk_R, ct_K); successful decapsulation gives K_S = K_R.
Step 4. The sender computes the detached signature S = Falcon.Sign(sk_S, F).
Step 5. The sender samples a fresh 16-byte IV and computes C = AES-256-CBC.Encrypt(K_S, IV, PKCS#7(F)).
Step 6. The sender transmits the application envelope {ct_K, IV, C, S}. The receiver decapsulates, decrypts, and removes the padding to recover F′.
Step 7. The receiver evaluates Falcon.Verify(pk_S, F′, S). It stores F′ only if verification succeeds; otherwise, the transfer is rejected.

The sequence keeps the three cryptographic roles separate: Kyber supplies the session secret, Falcon validates the recovered file, and AES-CBC encrypts the bulk payload. The transmitted envelope contains the Kyber ciphertext, IV, encrypted file, and detached signature. The receiver does not store the file until signature verification succeeds. This ordering supports the implemented file-verification behavior, but because the Kyber public key and transcript are not signed or otherwise identity-bound, the algorithm must not be interpreted as a complete authenticated key-exchange protocol.

2.3. Cross-platform application and operational functions

The system features a cross-platform graphical user interface (GUI) designed to present complex post-quantum cryptographic operations through an intuitive operational control dashboard. The software architecture separates real-time communication pipelines from local administrative and cryptographic management utilities.

To initiate a secure transfer, the operator selects a target recipient from the contact directory dropdown menu, which automatically binds and displays the active cryptographic context (e.g., testpk2.pem). After the receiver accepts the request, the application performs Kyber-based session-secret establishment, signs the selected file with Falcon, encrypts it with AES-256-CBC, and transmits the resulting envelope. A key-change warning is displayed when the configured key material changes; this is an operator warning rather than certificate-based authentication. The receiver interface operates on an automated, multi-threaded background listener stack. It accepts or rejects incoming requests, decrypts the received ciphertext, verifies the detached Falcon signature, and stores the file only after successful verification.

Figure 1 relates the cryptographic workflow to the actions visible to the operator. Figure 1(a) shows the sender interface, Figure 1(b) shows the receiver interface, and Figure 1(c) shows the warning raised when stored key information changes. Figure 1(d) illustrates the connection-request notification, while Figure 1(e) presents the file-transmission request notification. The warning assists local key management, but it is not a substitute for public key infrastructure (PKI), certificate validation, or cryptographic binding between the Kyber and Falcon identities.



Figure 1. Cross-platform graphical interface of the proposed framework; (a) sender interface, (b) receiver interface, (c) key-change warning, (d) connection-request notification, and (e) file-transmission request notification

Beyond active network communication channels, the framework encapsulates separate cryptographic administrative modules to manage identities, offline validation, and key lifecycles. The current implementation stores contact metadata and configured Falcon public keys locally, supports manual signing

and verification, and generates Kyber or Falcon key pairs as local .pem files. These utilities support the prototype workflow but do not constitute a PKI, certificate-revocation service, or independent trust-management system.

Figure 2 shows the local functions used to prepare and maintain the experimental environment. Figure 2(a) illustrates the contact management interface, where contact records associate peers with configured key material. Figure 2(b) shows the manual signing and verification interface, which allows the Falcon signature operations to be performed independently of network transmission. Figure 2(c) presents the notification displayed after successful manual signing. Figure 2(d) shows the key-generation interface for creating Kyber and Falcon key pairs, while Figure 2(e) displays the generated key files. Because these files are managed locally, trust establishment and revocation remain outside the present prototype.

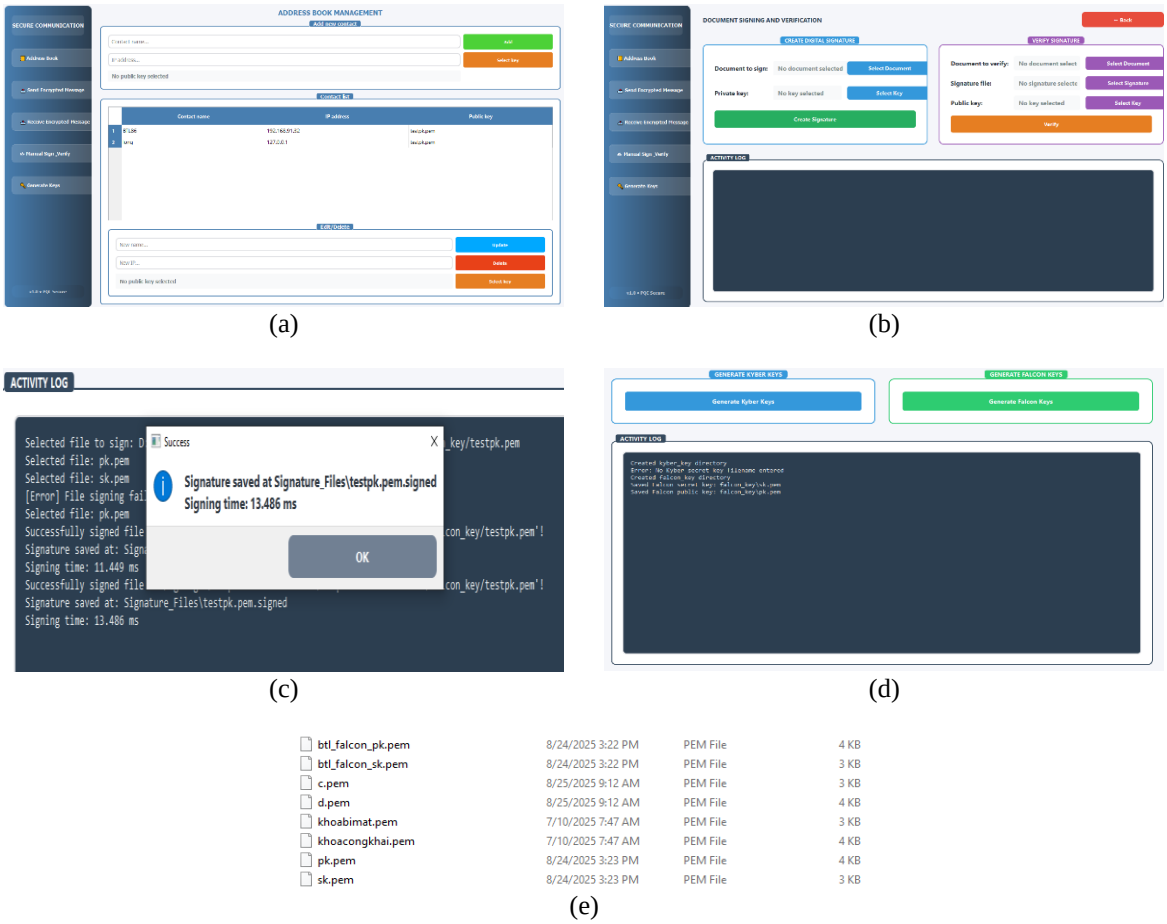


Figure 2. Administrative and cryptographic-lifecycle functions; (a) contact management, (b) manual signing and verification, (c) successful manual-signing notification, (d) key-generation interface, and (e) generated key files

3. RESULTS AND DISCUSSION

3.1. Performance results and interpretation

To assess the system's practical performance, the proposed framework was deployed in a real-world network environment involving both Linux and Windows platforms. Table 1 summarizes two evaluation settings. The original IEEE 802.11ac dataset contains Linux-to-Linux and Linux-to-Windows measurements for files from 1,000 to 40,365 KiB, whereas the controlled re-verification dataset contains same-host Windows and WSL2 TCP-loopback measurements up to 102,400 KiB. Each loopback operating-system/file-size configuration was run independently 10 times and is reported as an arithmetic mean (n=10). The retained aggregate JavaScript Object Notation (JSON) files do not contain the individual observations required for dispersion statistics, and the original wireless logs likewise do not preserve complete per-run dispersion data. The two datasets are therefore discussed separately.

The following key performance indicators were recorded and analyzed: i) session-key-establishment time using Kyber-1024; ii) encryption and decryption time using AES-256 (CBC mode); iii) total file transmission duration; and iv) manual digital signing and verification time using Falcon-1024. Separate manual Falcon measurements yielded signing/verification times of 31.812/8.029 ms for 1 MiB, 83.338/83.447 ms for 10 MiB, and 783.000/637.917 ms for 100 MiB. Because the retained records do not include the repetition count or individual observations, these values are treated as descriptive application-level results.

Table 1. Application-level performance results

Experimental environment	File size (KiB)	Session-key establishment (ms)	Encryption (ms)	Transmission (ms)	Decryption (ms)
Linux vs Linux: original IEEE 802.11ac	1000	25.03	7.64	251.40	9.03
	4352	25.30	29.80	488.61	16.43
	40365	25.45	200.79	263718.32	198.24
Linux vs Windows: original IEEE 802.11ac	1000	11.75	–	195.74	3.97
	4352	23.80	–	488.61	12.90
	40365	298.79	–	78367.32	137.89
Windows vs Windows: single-host TCP loopback	1000	1.68	3.89	0.69	3.19
	4352	2.86	10.11	1.06	9.01
	40365	2.70	79.45	10.42	88.83
	102400	1.93	201.14	19.78	212.76
Linux vs Linux: WSL2 single-host TCP loopback	1000	3.15	7.82	0.35	1.71
	4352	1.00	7.74	9.44	6.41
	40365	1.33	113.63	207.14	79.53
	102400	1.45	259.57	1467.81	208.81

The three Linux-to-Linux wireless session-key-establishment measurements (25.03, 25.30, and 25.45 ms) give a mean of 25.26 ms (approximately 25.3 ms). This value characterizes the application-level Kyber-based session-key-establishment time in the original physical-network testbed. Only Kyber-1024 was implemented, so the result is not used to infer the performance of Kyber-768 or to claim an advantage over elliptic curve Diffie–Hellman (ECDH) or optimized Kyber implementations. Results from algorithm-level and platform-focused studies [18], [20], [25], [26] are useful as context, but their measurement scope excludes much of the application, I/O, and network overhead included here.

For the 4,352 KiB sample, 29.8 ms of AES-CBC encryption corresponds to approximately 142.6 MiB/s. PyCryptodome used its default use_aesni=True setting on an AES-NI-capable processor, so the measured value reflects hardware-assisted AES execution. The timing also includes Python allocation, padding, memory copying, and other application-level processing. It therefore characterizes the implemented encryption path rather than the stand-alone performance of the AES primitive, and the contribution of AES-NI cannot be isolated because a matched use_aesni=False run was not recorded.

Transmission time increased with file size in the tested runs, but the archived wireless measurements also show substantial variation. The available records do not separate the effects of wireless conditions, real-time scanning, socket and file-system behavior, buffering, scheduling, or background load. The loopback data are therefore used only to re-check the implementation under a controlled local transport, not as evidence of physical-network scalability. Link bandwidth was not varied as an independent factor, so the present measurements do not support a quantitative bandwidth penalty.

Falcon application-level signing/verification increased from 31.812/8.029 ms at 1 MiB to 783.000/637.917 ms at 100 MiB. These measurements include file I/O, hashing, memory handling, and framework overhead, which were not timed separately. Both operations remained below one second for the 100 MiB sample in the archived manual test. Within the limits of these records, the result shows that the signing and verification path is practical at the tested file sizes, but it should not be read as a primitive-level Falcon benchmark.

3.2. Comparative scope and security evaluation

Kyber-1024, Falcon-1024, and AES-256-CBC perform different functions in the implemented workflow: session-secret establishment, file verification, and bulk confidentiality. Their combination is useful for evaluating a complete file-transfer application, but the security boundary remains important. The Falcon signature verifies the recovered file, while the Kyber encapsulation key and transcript are not bound to that signature. AES-CBC also provides no built-in integrity. The current construction therefore should not be treated as a formally analyzed authenticated-key-exchange or authenticated-encryption protocol.

To contextualize the observed performance, the results were compared with data from prior peer-reviewed studies. These include experimental benchmarks from published studies on post-quantum

cryptographic implementations, covering both prototype systems and standardized testbeds. Table 2 therefore compares studies by platform and evaluation scope rather than by raw speed alone. Lawo *et al*. [17] evaluated hardware-accelerated stacking on an NVIDIA BlueField-2 data processing unit (DPU); studies [19]–[26] cover constrained devices, heterogeneous processors, and TLS/QUIC environments; and the earlier data-protection prototype [30] used a different workflow and Advanced RISC Machines (ARM) platform. These differences explain why their timings are not directly interchangeable with the application-level measurements in this work.

Table 2. Comparison with peer-reviewed PQC studies by evaluation scope

Study and scope	Algorithms and platform	Finding and comparison boundary	Ref.
Proposed system: application-level file exchange	Kyber-1024, Falcon-1024, AES-256-CBC; Linux/Windows, Intel Core i5-9300H	25.3 ms wireless mean; 142.6 MiB/s for the 4,352 KiB AES sample. Application-level measurements: session-key timing includes network interaction, while AES timing includes application/I/O overhead.	This work
Lawo *et al*.: accelerated network stack	Falcon/Kyber and Dilithium/Kyber; NVIDIA BlueField-2 DPU	Demonstrates hardware-accelerated PQC stacking; not comparable with a consumer Python application.	[17]
Recent heterogeneous, constrained, and protocol studies [19]–[26]	Falcon/Kyber and alternative PQC; varied hardware and TLS/QUIC testbeds	Performance is strongly platform- and scope-dependent; results are contextual, not direct application-level comparisons.	[19]–[26]
Earlier data-protection prototype: application prototype	PQC data protection; ARM Cortex-A72	16 ms key establishment; 50 ms signing. Different workflow and measurement scope.	[30]

The implementation was also checked at the application-payload and file-verification levels. Wireshark/Scapy inspection did not reveal plaintext file content or session-key material in the captured application payloads, and a forged or mismatched Falcon file/signature pair was rejected. These tests confirm the behavior observed in the implemented path, but they do not evaluate network-metadata confidentiality, identity binding of the Kyber transcript, active man-in-the-middle (MITM) resistance, or traffic-analysis resistance.

Figure 3 illustrates the two security checks used in the experiment. Figure 3(a) shows ciphertext in the captured application payload rather than visible file plaintext, and Figure 3(b) shows rejection of a file/signature pair that fails Falcon verification under the configured public key. The figure supports these two implementation-level observations only; it is not evidence of complete protocol authentication or resistance to traffic analysis.

Together, these measurements show that the intended Kyber-Falcon-AES file-transfer workflow operated on the tested Linux/Windows platforms and that the receiver rejected data that failed the configured Falcon verification step. The measurements and packet checks remain limited to the implemented workflow and test environments; they do not establish authenticated peer identity, active MITM resistance, metadata confidentiality, or behavior at larger deployment scales.



(a) (b)

Figure 3. Application-level security checks; (a) captured application payload showing ciphertext with no visible plaintext file content and (b) rejection of a forged or mismatched Falcon signature

3.3. Limitations and future work

The main limitations follow directly from the design and the available measurements. The Kyber transcript is not bound to the configured Falcon identity, and AES-CBC has no built-in integrity; the present tests therefore do not establish authenticated key exchange or active MITM resistance. Network metadata remains visible, and traffic-analysis resistance was not evaluated. Performance evidence is also limited by incomplete per-run dispersion data, two-node wireless and same-host loopback testing, and the absence of a matched AES-NI-disabled control. Public keys are managed locally rather than through a PKI or revocation service.

Future work should address these points by binding peer identity and transcript data in an authenticated hybrid design, replacing CBC with an authenticated-encryption mode, and integrating the framework with established secure transports such as transport layer security version 1.3 (TLS 1.3), secure shell (SSH), or virtual private networks (VPNs). Further evaluation should retain per-run data and extend to controlled hardware-acceleration tests, concurrent or multi-recipient transfers, and broader network tests. Standards-oriented validation should focus on FIPS 203 and, for signatures, ML-DSA/FIPS 204 or FN-DSA after FIPS 206 is finalized.

4. CONCLUSION

This study presents and evaluates a cross-platform file-exchange implementation that combines Kyber-1024 KEM-based session-key establishment, Falcon-1024 detached file signatures, and AES-256-CBC encryption. In the original Linux-to-Linux wireless tests, session-key establishment averaged 25.26 ms (approximately 25.3 ms), and encryption of the 4,352 KiB sample reached approximately 142.6 MiB/s under the measured AES-NI-enabled configuration. Packet inspection found no plaintext file content or session-key material in the captured application payloads, and the receiver rejected a forged file/signature pair. These results support the feasibility of the implemented workflow on the tested commodity platforms. They also define its current boundary: the Kyber transcript is not identity-bound, CBC provides no built-in integrity, and the experiment does not establish active MITM or traffic-analysis resistance. The next development step is to strengthen protocol authentication and authenticated encryption while extending reproducible testing beyond the present two-node and loopback environments.

ACKNOWLEDGMENTS

The authors thank the Academy of Cryptography Techniques for its support.

FUNDING INFORMATION

No specific research grant or contract was received from public, commercial, or not-for-profit funding agencies.

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
Nhu-Quynh Luc	✓	✓								✓		✓		
Duc-Huy Quach			✓	✓		✓			✓					
Thanh-Long Le-Nguyen						✓		✓		✓	✓			
Duy-Phuong Nguyen				✓	✓		✓			✓				

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 no conflict of interest.

DATA AVAILABILITY

The study data are not publicly available; contact the corresponding author for inquiries.

REFERENCES

[1] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978, doi: 10.1145/359340.359342.

[2] Z. Zhang and Y. Zhao, "Enhanced Elliptic Curve Cryptography (EECC)," *Procedia Computer Science*, vol. 247, no. C, pp. 1324–1330, 2024, doi: 10.1016/j.procs.2024.10.158.

[3] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997, doi: 10.1137/S0097539795293172.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Annual ACM Symposium on Theory of Computing*, 1996, pp. 212–219, doi: 10.1145/237814.237866.

[5] A. Ambainis, A. Bačkurs, N. Nahimovs, and A. Rivosh, "Grover's algorithm with errors," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7721 LNCS, pp. 180–189, 2013, doi: 10.1007/978-3-642-36046-6_17.

[6] J. Daemen and V. Rijmen, *The Design of Rijndael*. in Information Security and Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, doi: 10.1007/978-3-662-04722-4.

[7] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?," *IEEE Security and Privacy*, vol. 16, no. 5, pp. 38–41, 2018, doi: 10.1109/MSP.2018.3761723.

[8] G. Alagic *et al.*, "Status report on the fourth round of the NIST post-quantum cryptography standardization process," Mar. 2025, doi: 10.6028/NIST.IR.8545.

[9] P. -A. Fouque *et al.*, "Falcon : Fast-Fourier Lattice-based Compact Signatures over NTRU," Falcon-sign.info. [Online]. Available: https://falcon-sign.info/falcon.pdf.

[10] J. Bos *et al.*, "CRYSTALS - Kyber: A CCA-Secure Module-Lattice-Based KEM," in *Proceedings - 3rd IEEE European Symposium on Security and Privacy, EURO S and P 2018*, 2018, pp. 353–367, doi: 10.1109/EuroSP.2018.00032.

[11] G. M. Raimondo, L. E. Locascio, and K. M. Stine, "Module-lattice-based key-encapsulation mechanism standard," Gaithersburg, MD, USA, Aug. 2024, doi: 10.6028/NIST.FIPS.203.

[12] G. M. Raimondo and L. E. Locascio, "Module-lattice-based digital signature standard," Aug. 2024, doi: 10.6028/NIST.FIPS.204.

[13] E. Alkim, L. Ducas, T. Poppelmann, and P. Schwabe, "Post-Quantum Key Exchange—A New Hope," in *SEC'16: Proceedings of the 25th USENIX Conference on Security Symposium*, 2016, pp. 327–343.

[14] D. Moody *et al.*, "Status report on the second round of the NIST post-quantum cryptography standardization process," Gaithersburg, MD, Jul. 2020, doi: 10.6028/NIST.IR.8309.

[15] D. Stebila and M. Mosca, "Post-quantum Key Exchange for the Internet and the Open Quantum Safe Project," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10532 LNCS, pp. 14–37, 2017, doi: 10.1007/978-3-319-69453-5_2.

[16] E. Crockett, C. Paquin, and D. Stebila, "Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH," Archive, Cryptology ePrint. [Online]. Available: https://eprint.iacr.org/2019/858.

[17] D. C. Lawo *et al.*, "Falcon/Kyber and Dilithium/Kyber Network Stack on Nvidia's Data Processing Unit Platform," *IEEE Access*, vol. 12, pp. 38048–38056, 2024, doi: 10.1109/ACCESS.2024.3374629.

[18] M. Kumar, "Post-quantum cryptography Algorithm's standardization and performance analysis," *Array*, vol. 15, p. 100242, 2022, doi: 10.1016/j.array.2022.100242.

[19] M. Vidaković and K. Miličević, "Performance and Applicability of Post-Quantum Digital Signature Algorithms in Resource-Constrained Environments," *Algorithms*, vol. 16, no. 11, p. 518, 2023, doi: 10.3390/a16110518.

[20] 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, p. 32, 2025, doi: 10.3390/cryptography9020032.

[21] M. Raavi, Q. Khan, S. Wuthier, P. Chandramouli, Y. Balytskyi, and S. Y. Chang, "Security and Performance Analyses of Post-Quantum Digital Signature Algorithms and Their TLS and PKI Integrations †," *Cryptography*, vol. 9, no. 2, p. 38, 2025, doi: 10.3390/cryptography9020038.

[22] J. A. Montenegro, R. Rios, and J. Lopez-Cerezo, "A performance evaluation framework for post-quantum TLS," *Future Generation Computer Systems*, vol. 175, p. 108062, 2026, doi: 10.1016/j.future.2025.108062.

[23] J. A. Montenegro, R. Rios, and J. Bonilla, "Comparative analysis of post-quantum handshake performance in QUIC and TLS protocols," *Computer Networks*, vol. 275, p. 111957, 2026, doi: 10.1016/j.comnet.2025.111957.

[24] M. A. Khan, F. Noor, S. Javaid, and J. Żywiołek, "Implementation and performance of post-quantum cryptography for resource constrained consumer electronics," *Discover Internet of Things*, vol. 5, no. 1, p. 139, 2025, doi: 10.1007/s43926-025-00238-x.

[25] D. L. Lucaciu and D. E. Popescu, "Cross-Platform Performance and Security Evaluation of Post-Quantum Cryptographic Algorithms on Resource-Constrained Devices," *Applied Sciences (Switzerland)*, vol. 16, no. 12, p. 5781, 2026, doi: 10.3390/app16125781.

[26] N. Nagy *et al.*, "Module-Lattice-Based Key-Encapsulation Mechanism Performance Measurements," *Sci*, vol. 7, no. 3, p. 91, 2025, doi: 10.3390/sci7030091.

[27] D. J. Bernstein and T. Lange, "Post-quantum cryptography," *Nature*, vol. 549, no. 7671, pp. 188–194, 2017, doi: 10.1038/nature23461.

[28] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 11505 LNCS, 2019, pp. 206–226, doi: 10.1007/978-3-030-25510-7_12.

[29] E. Rescorla, "RFC 8446: Internet-Draft - The Transport Layer Security (TLS) Protocol Version 1.3," Aug. 2018, doi: 10.17487/RFC8446.

[30] T. T. Nguyen, D. H. Quach, T. T. Dao, and N. Q. Luc, "Building and Developing a Data Protection Service System Based on Postquantum Cryptography Solutions," *Programming and Computer Software*, vol. 51, no. 5, pp. 314–326, 2025, doi: 10.1134/S0361768825700197.

BIOGRAPHIES OF AUTHORS



Nhu-Quynh Luc Ph.D. was born in Nam Dinh, Vietnam, in 1983. He received a bachelor's degree in mathematics from Vietnam National University, University of Science, in 2006 and a master's degree in cryptographic engineering from the Academy of Cryptography Techniques, Vietnam. He holds a Ph.D. in electronic materials science. His research has included the mathematical foundations of elliptic-curve and post-quantum cryptography and their cryptographic applications. His current interests include organic materials and microelectronic fabrication. He can be contacted at email: quynhln@actvn.edu.vn or lucnhuquynh69@gmail.com.



Duc-Huy Quach currently serves as an information security engineer at Center 3, Command 86. He received an engineering degree in cryptographic engineering from the Academy of Cryptography Techniques in 2023. His research interests include post-quantum security, onboard system integration, and comprehensive information-technology application solutions. He can be contacted at email: quachhuy.job@gmail.com.



Thanh-Long Le-Nguyen received an engineering degree in cryptographic engineering from the Academy of Cryptography Techniques, Vietnam, in 2024. He currently works at the General Staff Department of Military Region 3, Vietnam. His research interests include post-quantum cryptography, information security, cybersecurity, quantum-resistant protocols, and secure communication systems. He can be contacted at email: long7a168@gmail.com.



Duy-Phuong Nguyen received a master's degree in cryptographic engineering from the Academy of Cryptography Techniques, Vietnam. His research focuses on electronic-system design for the internet of things and hardware-based encryption. He can be contacted at email: phuongladecduy@gmail.com.