frontiers | Frontiers in Physics

TYPE Original Research
PUBLISHED 08 January 2026
DOI 10.3389/fphy.2025.1723966

Design and implementation of an authenticated post-quantum session protocol using ML-KEM (Kyber), ML-DSA (Dilithium), and AES-256-GCM

Akinlemi Olushola and S. P. Meenakshi*

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu, India

OPEN ACCESS

EDITED BY
Umar Khokhar,
Georgia Gwinnett College, United States

REVIEWED BY
G. Palai,
Sri Sri University, India
Muhammad Yasir Faheem,
COMSATS University Islamabad, Pakistan

*CORRESPONDENCE
S. P. Meenakshi,
spmeenakshi@vit.ac.in

RECEIVED 13 October 2025
REVISED 15 November 2025
ACCEPTED 17 November 2025
PUBLISHED 08 January 2026

CITATION
Olushola A and Meenakshi SP (2026) Design and implementation of an authenticated post-quantum session protocol using ML-KEM (Kyber), ML-DSA (Dilithium), and AES-256-GCM.
Front. Phys. 13:1723966.
doi: 10.3389/fphy.2025.1723966

Introduction: Session establishment, the process by which two parties authenticate each other and derive a shared secret key, forms the foundation for secure digital communication. Quantum computers threaten this foundation by breaking classical public-key primitives such as RSA and elliptic-curve Diffie–Hellman (ECDH), thereby enabling harvest-now–decrypt-later (HNDL) attacks that endanger long-term confidentiality.
Methods: This paper presents the design, implementation, and evaluation of an authenticated, quantum-resistant session protocol that replaces these vulnerable mechanisms with their post-quantum counterparts. The proposed protocol integrates ML-KEM-1024 (FIPS 203; CRYSTALS, Kyber) for ephemeral key exchange, ML-DSA-65 (FIPS 204; CRYSTALS, Dilithium) for endpoint authentication, and AES-256-GCM for symmetric protection. A transcript-bound HKDF–SHA3-256 key schedule and a 96-bit GCM nonce construction with conservative rekey limits are used to ensure forward secrecy, downgrade resistance, and message integrity. A Python/C prototype (PQClean ML-KEM-1024 with PyCryptodome AES-256-GCM) was benchmarked over 1,000 iterations on commodity hardware.
Results: The results show that sub-millisecond cryptographic overhead ML-KEM-1024 matches the performance of X25519 while vastly outperforming RSA-3072 in secure session establishment, and symmetric encryption remains cost effective. Nonces are unique 96-bit values, never reused across directions or beyond 2^{32} records, following NIST SP 800-38D; when nonce-misuse resistance is required, AES-256-GCM-SIV (RFC 8452) is supported as a drop-in alternative. Empirical tests under both local and WAN-emulated (≈ 40 ms RTT) network conditions confirm that the additional post-quantum cost maintains the handshake cryptographic latency in the 0.50–0.70 ms range.
Discussion: These results demonstrate that fully authenticated, forward-secure, quantum-resistant session negotiation is practical for real-world deployments.

KEYWORDS

post-quantum cryptography, ML-KEM (Kyber, FIPS 203), ML-DSA (Dilithium, FIPS 204), AES-256-GCM, authenticated quantum-resistant key exchange, harvest-now–decrypt-later

1 Introduction

Session establishment is the process of authenticating two entities and deriving a shared secret key, which is the foundation of secure communication protocols such as Transport Layer Security (TLS), Secure Shell (SSH), and virtual private networks (VPNs). However, the rapid progress of quantum computing introduces a transformative yet existential risk to this foundation as quantum algorithms can efficiently solve mathematical problems that underpin classical session handshakes. The rapid progress of quantum computing represents a major shift in computational power, simultaneously constituting a significant emerging threat to existing cryptographic infrastructure [1]. Shor's algorithm, which can factor large numbers and calculate discrete logarithms on a powerful quantum computer with high efficiency, has the potential to break common public-key cryptosystems such as RSA and elliptic-curve Diffie–Hellman (ECDH) [2, 3]. These classical algorithms underlie secure online transactions, communication channels, and critical infrastructures worldwide. The inevitable emergence of large-scale quantum computers renders these schemes insecure, enabling attackers to decrypt existing and future communications, generate fake digital signatures, and compromise authentication mechanisms [4].

High-security uses, including finance [5], healthcare [6], government messaging [7], and defense networks [8], are most susceptible to such assaults. Such uses process extremely sensitive data, including financial transactions and medical histories, government-classified data, and election vote records. A successful cryptanalytic attack on the underlying key exchange mechanisms of such uses results in a catastrophic loss of confidentiality, integrity, and trust. This necessitates an urgent transition to post-quantum cryptographic mechanisms, which offer security guarantees even against quantum-enabled adversaries [9]. Specifically, the handshake/session establishment phase, in which asymmetric key exchange and authentication occur, is the most vulnerable phase for future quantum adversaries. To counter this threat, global cryptographic communities and standardization forums have hastened their efforts to discover, standardize, and deploy cryptographic primitives that are resistant to post-quantum attacks [10–12]. The US National Institute of Standards and Technology (NIST) has embarked on a high-profile, multiyear standardization process to rigorously test candidate algorithms for key encapsulation mechanisms (KEMs), including quantum-resistant digital signatures. One of the chosen algorithms is ML-KEM (FIPS 203 [13]; previously CRYSTALS-Kyber), a lattice-based key encapsulation mechanism, due to its outstanding security characteristics, performance optimization, and versatility in accommodating a wide range of software execution environments [14, 15]. The security offered by Kyber is based on the hardness of the module learning with errors (module-LWE) problem, which is one of the lattice problems conjectured to remain difficult, even for quantum computers [16]. Thus, Kyber is one of the first-batch candidates to take over the insecure classical key-exchange mechanisms, such as the Diffie–Hellman key-exchange method encapsulated under the name ECDH.

Conversely, symmetric-key encryption algorithms, such as the Advanced Encryption Standard (AES), are highly resistant to quantum computing attacks. Grover's algorithm provides a quadratic speedup of brute-force attacks on symmetric ciphers; AES-256 is nonetheless considered secure with an effective security strength comparable to AES-128 against quantum attackers [17, 18]. This feature makes AES an attractive building block in systems designed to be secure against quantum computers, especially when used for the efficient and high-throughput encryption of large volumes of data after secure session key establishment.

Although Kyber and AES have been well-studied separately, there has been a significant shortage of field guidance and implementation for combining these primitives into deployable, post-quantum secure session layer protocols for high-security domains [9, 19]. Most prior studies have focused on algorithm-level performance tuning [20–22], hardware-accelerated Kyber on embedded systems [23], domain-specific implementations such as secure key exchange for UAV networks [24] and automotive systems [25], and post-quantum KEM integration into well-established transport protocols such as TLS [26, 27]. Although useful, these attempts mostly focus on specific use cases or lack an explicit general-purpose protocol definition that can be widely applied across various high-security domains such as finance, healthcare, and government systems.

In addition, most current studies lack obvious and readable implementations and thorough performance evaluations in realistic programming environments [9, 15]. For organizations planning to migrate to post-quantum secure infrastructure, there is an urgent need for understandable and flexible protocol frameworks that define the process of replacing vulnerable ECDH-based key exchanges with Kyber while continuing to offer efficient symmetric encryption of transactional data using AES. There is also a requirement for usable threat modeling that considers attackers with quantum capabilities, thus ensuring session key integrity and information confidentiality, even in advanced attack scenarios.

To address this disparity, this paper presents the design and deployment of a post-quantum-resistant key exchange and encryption protocol for high-security applications in the automotive industry. We substituted insecure elliptic-curve Diffie–Hellman protocols with Kyber KEM for post-quantum session key exchange, leveraging well-established security assumptions to achieve quantum resistance [11]. We continue to use AES-256 symmetric encryption for payload confidentiality and leverage its well-established security properties and practical performance profile to encrypt sensitive data efficiently once the session key is securely established. For authentication, the protocol integrates ML-DSA (FIPS 204; formerly CRYSTALS-Dilithium), ensuring end-to-end authenticity and message integrity during session establishment and throughout communication.

The protocol is designed in a modular fashion, making its adoption into available communication frameworks possible without requiring a full overhaul of the application layer logic [9]. We define a clear threat model based on realistic adversarial capabilities, including quantum-capable attackers capable of compromising classical key-exchange schemes, and analyze how our proposed design ensures the confidentiality and integrity of session keys and encrypted payloads under these assumptions. We describe the system architecture in detail, including participant roles, cryptographic operations, key management considerations, and message flow diagrams, to enable precise and unambiguous implementation.

To demonstrate the applicability of the suggested protocol in real-world settings, we deployed it in Python using available cryptographic libraries, specifically a PQClean-backed ML-KEM-1024 (Kyber-1024) C implementation accessed via `ctypes` and PyCryptodome for AES-256-GCM [28]. The open-source implementation provides an actionable reference design that can be used, modified, and tested by security experts and researchers, based on their use-case scenarios. We perform a performance analysis that considers the measurements of key generation, encapsulation, decapsulation, and AES encryption/decryption time, thus providing empirical evidence of the protocol's usability for implementation in latency-sensitive, high-security applications [21].

This work aligns with the general goal of enabling crypto-agility in secure communication contexts—that is, the ability to quickly and consistently switch away from threatened cryptographic primitives after new threats are discovered [11]. By describing a complete, tested, and well-documented protocol design that integrates the NIST-recommended post-quantum key exchange with established symmetric encryption practices, we offer an applied resource for organizations intending to transition to post-quantum-secure infrastructure.

The main contributions of this study are as follows:

- Protocol design: We specify a general-purpose, modular session-layer protocol integrating Kyber KEM for quantum-resistant key exchange with AES-256 for symmetric encryption of payloads.
- Threat modeling: We define a realistic threat model tailored to high-security applications, considering adversaries with quantum capabilities and outlining security goals for confidentiality and integrity.
- Implementation: We provide an open Python implementation demonstrating the practical feasibility of the protocol using available cryptographic libraries.
- Performance evaluation: We measure and analyze key generation, encapsulation/decapsulation, and encryption/decryption timings to assess the overhead introduced by post-quantum secure operations.
- Applicability to high-security domains: We discuss how the protocol can be integrated into existing systems in finance, healthcare, government, and other critical sectors, supporting secure transaction processing in the quantum era.

Our proposed design and implementation provide a tangible solution for improving the security of critical data exchanges against future quantum threats by bridging the gap between cryptographic theory and real-world applications.

The remainder of this paper is organized as follows: Section 2 reviews prior work on classical public-key cryptography and quantum threats, post-quantum cryptography (PQC), the ML-KEM (Kyber) design and properties, and the use of AES-256-GCM in secure protocols. Section 3 presents the protocol design and system architecture of the proposed approach. Section 4 describes the implementation, simulation setup, and results of the study. Section 5 establishes a comparative performance evaluation benchmark, and Section 6 consolidates the empirical findings related to protocol scalability and payload optimization. Section 7 discusses and synthesizes key findings and provides practical recommendations for future research. Finally, Section 8 concludes the paper and outlines future research directions.

2 Background and related work

We summarize 50 state-of-the-art studies (see Table 1), highlighting their contributions, methodological focus, and limitations in the context of general-purpose post-quantum secure protocol design.

2.1 Classical public-key cryptography and quantum threats

Existing secure communication protocols employ public-key cryptography to share session keys between parties over insecure channels. Classic algorithms such as RSA and ECDH have secured online banking, enabled e-commerce secure emails, and influenced several Internet standards [29, 30]. RSA relies on the assumption that it is computationally infeasible to factor large composite numbers, whereas ECDH relies on the difficulty of the elliptic-curve discrete logarithm problem to ensure a key exchange. These cryptographic standards have been successful in offering substantial security guarantees against classical attackers for several decades and are the building blocks of protocols such as TLS, SSH, and IPsec [31, 32].

Quantum computing poses a catastrophic threat to the security of protective mechanisms. In 1994, Shor demonstrated that a sufficiently powerful quantum computer could factor integers and calculate discrete logarithms in polynomial time, thereby breaking the RSA and ECDH [1]. Large quantum computers have not yet been constructed, but the concept of harvest-now–decrypt-later (HNDL) attacks makes this threat imminent for systems that require long-term confidentiality [4]. Attackers can store encrypted communications and decrypt them when quantum technology is available, thereby retroactively compromising privacy.

Consequently, cryptographic communities and governments have called for quantum-resistant (or "post-quantum") alternatives to vulnerable schemes. To address this urgency, the US National Security Agency (NSA) has recommended the planning of quantum-resistant algorithms for national security systems [8]. The European Union Agency for Cybersecurity (ENISA) and standard bodies such as NIST have also underscored the need for migration plans for post-quantum cryptography [10, 33]. The starting point of this transition is the awareness of the vulnerabilities of classical schemes and the compelling need for secure, implementable alternatives that can resist quantum-capable adversaries in terms of performance, scalability, and compatibility with existing systems. These weaknesses make it imperative to replace traditional key exchanges with more secure alternatives. In this section, the new area of post-quantum cryptography and the work toward standardizing secure replacements are summarized.

2.2 Post-quantum cryptography overview

Post-quantum cryptography refers to cryptographic algorithms designed to be resistant to attacks by quantum computers [9].

TABLE 1 Summary of related work (2015–2025).

#	Title (year)	Journal	Methodology	Contribution	Limitation/gap	References
1	Formal PQXDH protocol analysis (2025)	Cryptology ePrint Archive	Protocol formalism	BAKE formalism for Signal's PQXDH	Limited to messaging	[57]
2	Hybrid PQ-AKA for 6G (2025)	IEEE comm. Surveys and tutorials	Survey	Survey of PQ AKA designs	Review only	[59]
3	PQC key exchange for 5G (2025)	IEEE NOMS	Network protocol evaluation	Kyber + ID-based encryption for 5G	Telecom-specific	[58]
4	PQC for IoT privacy (2025)	Cluster computing	IoT protocol proposal	PQC for IoT user privacy	IoT-specific focus	[60]
5	PQC in wireless protocols (2025)	Network	Experimental system	PQC for secure wireless stack	Low-level test only	[64]
6	Migration guidelines v2 (2025)	NIST draft	Policy update (planned)	Prepares for PQC deployment	Non-technical guidance	[67]
7	Quantum-safe web at WAN scale (2025)	Computers and security	Benchmarking study	PQ-TLS under real WAN loss/latency	TLS-focused; hybrid dependency	[42]
8	MACsec with hybrid Kyber (2025)	EPJ quantum technology	Link-layer integration	PQC key exchange within MACsec	Link-layer specific	[45]
9	PQC performance across architectures (2025)	IEEE access	Cross-arch benchmarking	ML-KEM/ML-DSA on heterogeneous CPUs	No session-layer design	[46]
10	PQ signatures on IoT platforms (2025)	Sensors (MDPI)	Embedded evaluation	ML-DSA/Falcon for IoT	Resource constraints remain	[47]
11	End-to-end PQC app-layer prototype (2025)	Cryptography (MDPI)	Implementation prototype	Replay resistance, key-update cycles	Limited domains	[48]
12	Code-based TLS handshakes (2025)	Quantum Inf. processing	TLS study	Code-based KEMs as alternatives	No latency comparison	[51]
13	SARG04 QKD performance (2025)	Journal of optics (springer)	Photonic-QKD experiment	Quantum comm. performance metrics	Physical-layer focus	[53]
14	TLS 1.3 hybrid transition framework (2024)	Sensors (MDPI)	Standards/architecture	Kyber + ECDH hybridization path	Retains classical dependency	[44]
15	TLS authentication with ML-DSA/Falcon (2024)	NDSS symposium	Security and performance study	PQ signature integration in TLS	Focused on authentication only	[41]
16	PQC payloads and TTLB impact (2024)	NIST PQC conf.	Network performance	MTU fragmentation analysis	No session-layer design	[68]
17	Post-quantum authentication for e-health (2024)	Internet of things	IoT AKE scheme	Kyber-based authentication for e-health	Application-specific	[61]
18	PQC mobile messaging (2024)	DSSR journal	Mobile AKE with PQ KEM	PQ-secure mobile protocols	Limited to mobile	[56]
19	QKD + PQC authenticated encryption (2024)	Computers (Elsevier)	Architecture and testing	Hybrid QKD and PQ AE design	QKD-dependent	[63]

(Continued on the following page)

TABLE 1 (*Continued*) Summary of related work (2015–2025).

#	Title (year)	Journal	Methodology	Contribution	Limitation/gap	References
20	Quantum-safe hybrid system (2024)	Advanced quantum technologies	Hybrid QKD + PQC	QKD–PQC integration test	Needs special hardware	[62]
21	Messaging with CSIDH/Falcon (2024)	Programming and computer software	Secure messaging design	PQC messaging with CSIDH, falcon, and AES	Niche use case only	[55]
22	WireGuard PQC integration (2024)	Mullvad blog	VPN experimentation	Kyber-based KEX in WireGuard	Not production-standard	[69]
23	Hybrid PQC in network stacks (2024)	SBSEG conference	Network integration	PQC hybrid adoption across layers	Mixed trust assumptions	[50]
24	PQC-enabled TLS at internet scale (2023)	CoNEXT companion (ACM)	Internet-scale benchmarking	Kyber/Dilithium handshake latency	No transcript-bound HKDF	[40]
25	PQC performance under high RTT (2023)	SIGCOMM workshop (ACM)	Network measurement	PQ-TLS latency under high RTT	TLS benchmarking only	[43]
26	Quantum computing: Bits to qubits (2023)	Springer nature	Conceptual review	Quantum logic, crypto implications	Theoretical, non-protocol	[52]
27	PQXDH messaging protocol (2023)	Signal blog	Protocol design	Kyber in Signal's PQXDH handshake	App-specific	[66]
28	Kyber in QUIC (2023)	Microsoft security blog	Integration testing	QUIC handshake evaluation	Hybrid fallback mode	[70]
29	Kyber in TLS 1.3 (2023)	Cloudflare blog	Deployment test	Real-world PQ-TLS performance	Hybrid with ECDH	[71]
30	UAV Kyber-AKE (2023)	Sensors journal	Authenticated key agreement	UAV communication security	Domain-specific trust	[24]
31	Hybrid key exchange survey (2023)	J. Cryptographic engineering	Systematic mapping	Hybrid KEM/AKE taxonomy	No protocol design	[54]
32	Group lattice KEX (2023)	IET communications	Group protocol proposal	Lattice-based forward secrecy	Limited to groups	[49]
33	Automotive integration (2022)	SAFECOMP	Network evaluation	PQC in automotive systems	Domain-specific	[25]
34	Kyber standardization (2022)	NIST PQC submissions	Parameter selection	Kyber NIST finalist	Lacks system-level design	[15]
35	PQC benchmarks (2021)	IEEE HOST	Hardware benchmarking	Kyber timing data	No AES integration details	[28]
36	EU PQC planning (2021)	ENISA report	Policy study	Regional transition planning	No protocol specs	[33]
37	FPGA acceleration (2020)	SAFECOMP	Hardware performance	FPGA-based Kyber optimization	No session-layer design	[21]
38	Kyber on IoT devices (2019)	ACM TECS	Embedded optimization	ARM Cortex-M PQC design	Primitive-level only	[23]

(Continued on the following page)

TABLE 1 (*Continued*) Summary of related work (2015–2025).

#	Title (year)	Journal	Methodology	Contribution	Limitation/gap	References
39	IETF hybrid TLS draft (2019)	IETF draft	Standards proposal	Hybrid KEX for TLS 1.3	Not pure PQC	[26]
40	Quantum threat planning (2018)	IEEE security	Threat modeling	Harvest-now–decrypt-later analysis	No protocol proposal	[4]
41	Hybrid KEM theory (2018)	EUROCRYPT	Formal proof	Hybrid AKE security	No implementation	[35]
42	ML-KEM (Kyber) design (2018)	PQCrypto conference	Algorithm proposal	Defines Kyber KEM	No integration	[14]
43	PQC standardization process (2017)	NIST reports	Multi-round evaluation	Selection of PQC KEMs	Algorithm-level only	[10]
44	PQC call and overview (2016)	NIST report	Requirement analysis	Need for PQC algorithms	No deployment guide	[9]
45	Lattice crypto survey (2016)	Communications of the ACM	Theory survey	Lattice security foundations	Theoretical	[16]
46	CECPQ1/2 experiments (2016)	Google tech report	TLS hybrid trials	Demonstrates hybrid feasibility	Classical dependency	[38]
47	Migration guidelines (2015)	NSA reports	Policy recommendations	Urges crypto-agility	No protocols	[8]
48	Quantum algorithms (1997)	SIAM J. Computing	Algorithmic breakthrough	Shor's factorization	No protocol-level solution	[1]
49	Quantum key distribution over FSO channel (2015)	Wireless networks (springer)	Experimental QKD protocol	Secure optical free-space quantum key distribution with reconciliation mechanism	Physical-layer focus, not PQC deployable	[65]
50	Understanding of argon fluid sensor (2023)	Quantum computing: A shift from bits to qubits (springer nature)	Semiconductor quantum modeling	Explores single-quantum-well photonic sensors in quantum computational structures	Non-cryptographic, hardware-level quantum design	[43]

In contrast to quantum cryptography, which requires quantum communication channels, the PQC is applicable to classical infrastructure, thus opening a potential migration path for quantum cryptography.

The US National Institute of Standards and Technology (NIST) Post-quantum Cryptography Standardization Project, launched in 2016, has been a leading effort in this regard. The project invited cryptographic researchers worldwide to propose candidate algorithms for standardization to supersede insecure public-key primitives [10, 11]. After multiple rounds of security-, performance-, and implementation-based evaluations, NIST has selected a few schemes for standardization. Among them, ML-KEM (FIPS 203; formerly CRYSTALS–Kyber) was selected as the standard KEM owing to its robust security reduction to hard lattice problems, good performance on resource-constrained devices, and small key sizes [14, 15].

PQC deployment is not an abstraction process. Industry players and governments have begun to develop migration proposals. For instance, the US NSA released a guide for post-quantum algorithm migration in the national security infrastructure [34]. Cloud platforms, banking infrastructures, and Internet standard bodies have been experimenting with post-quantum integration into real-world protocols, such as hybrid post-quantum/TLS ciphers [26].

However, despite this progress, several practical problems remain. The incorporation of PQC into existing systems requires careful consideration of performance overheads, key and ciphertext sizes, and integration with the legacy infrastructure. Moreover, security analysis must consider practical threat models in which an adversary can mount quantum attacks against key exchange protocols but uses classical cryptanalysis against symmetric encryption methods. These problems must be addressed not only by defining new algorithms but also by constructing implementable protocols that explain how such algorithms can be employed in secure-communication systems. Of all the candidates, we mainly focused on Kyber because of its strong security guarantees and its acknowledged effectiveness in test deployments.

2.3 Kyber KEM: design and properties

ML-KEM (FIPS 203; formerly CRYSTALS-Kyber) is a lattice-based key encapsulation mechanism chosen by NIST as the flagship post-quantum secure key exchange standard [15]. It is secure based on the hardness of the module-LWE problem, a structured lattice problem conjectured to be difficult even for adversaries with quantum capabilities [16]. In contrast to factorization-based and discrete logarithm-based schemes, which are broken by Shor's algorithm, lattice problems have no known efficient quantum algorithms. Therefore, they are suitable for post-quantum cryptographic applications.

Kyber offers different levels of security, such as Kyber-512, Kyber-768, and Kyber-1024, which offer successively increased security margins at the cost of increased key sizes and ciphertexts [14]. The protocol offers efficient encapsulation and decapsulation algorithms by which two parties can reach a common agreement on a symmetric key to be shared over an insecure channel. Therefore, Kyber is a highly suitable direct replacement for conventional Diffie–Hellman key exchange protocols in protocols such as TLS, SSH, and VPN tunnels [21].

Kyber was designed to perform well on high-end servers and resource-constrained devices. Various studies have reported evidence of optimized executions on embedded systems such as ARM Cortex-M processors [23] and field-programmable gate array (FPGA) accelerators [21]. Additionally, efforts have been directed toward making the Kyber multiplatform and multilanguage software library compatible with, for example, OpenSSL and PQCrypto packages for the Python programming language. These implementations have shown that Kyber is viable even for applications with low-latency requirements, with encapsulation and decapsulation times in the millisecond range on common hardware [28].

The security proofs of Kyber consist of worst-case lattice problem formal reductions, protection against known side-channel attacks via constant-time execution and masking, and chosen ciphertext attack resistance through the Fujisaki–Okamoto transformations [14, 35]. These properties make it a prominent candidate for session-key protection in high-security applications in which confidentiality and integrity are paramount.

In this study, we targeted Kyber-1024 for conservative security margins that are appropriate for high-security transaction systems.

However, deploying Kyber in practice requires careful protocol design to manage key distribution, encapsulation failures, and integration with symmetric encryption schemes for payload-data. Our work builds on these foundations by specifying a concrete deployable protocol that combines Kyber-based key exchange with AES-256 symmetric encryption in a general-purpose session-layer design for high-security applications.

2.4 AES-256 symmetric encryption in secure protocols

Although quantum computers endanger the safety of public-key cryptosystems, symmetric-key encryption remains relatively secure. Grover's algorithm provides a quadratic speedup for brute-force attacks, effectively halving the security level of a symmetric cipher key size [17]. Therefore, AES-256, which provides 256-bit security against classical attacks, remains safe at a quantum security level of approximately 128 bits [9, 18].

AES-256 has been officially standardized and widely used as part of many secure communication protocols, including TLS, IPSec, and secure messaging protocols. It exhibits excellent performance and well-defined security properties, including resistance to differential and linear cryptanalysis and well-studied modes of operation [36, 37]. Furthermore, its hardware-accelerated implementations, which are available on modern CPUs via AES-NI and ARM Cryptography Extensions, have low overheads, even for high-throughput applications.

Its pairing with post-quantum KEMs such as Kyber is based on a hybrid strategy that leverages the strengths of these two primitives. The post-quantum KEM sets up an AES-256-resistant shared session key, and AES-256 is used for fast, high-throughput bulk-data encryption. Through this pairing, crypto-agility is facilitated, enabling systems to shift away from legacy key exchange algorithms without affecting the existing symmetric encryption logic [26].

Despite its trusted strength, AES security relies fundamentally on the secure exchange of session keys. An insecure key agreement can compromise the confidentiality and integrity of a system. Therefore, it is necessary to develop secure protocols that efficiently combine post-quantum key exchange protocols and symmetric encryption to provide end-to-end security assurance in high-security use cases, such as financial systems, healthcare data exchanges, and government communication. The synergy of post-quantum key establishment with AES-256 symmetric encryption forms the basis of the proposed protocol.

2.5 Related work on post-quantum secure protocols

Research on post-quantum secure protocols has accelerated over the last decade in response to the emerging threat of large-scale quantum computers [1, 4, 9]. A central challenge is to replace vulnerable public-key primitives such as RSA and ECDH with post-quantum algorithms while maintaining the efficiency and security properties required for deployment in real-world communication systems [8, 10, 11].

A major area of research has focused on the standardization of post-quantum cryptographic algorithms. The NIST launched its Post-quantum Cryptography Standardization Project to evaluate and select secure, efficient, and deployable alternatives [9, 11]. ML-KEM (FIPS 203; formerly CRYSTALS-Kyber) was chosen as the primary KEM because of its strong security reduction to lattice problems, performance efficiency, and compact key sizes [14–16].

Several implementation and migration efforts have been made to address these issues. Google's CECPQ1 and CECPQ2 experiments deployed hybrid TLS cipher suites combining ECDH with post-quantum KEMs, such as NewHope, to test backward-compatible integration [38]. The IETF proposed a hybrid key exchange for TLS 1.3, which incorporates Kyber with classical mechanisms [26, 27, 39]. Similarly, the NSA and ENISA recommended crypto-agile systems capable of transitioning away from classical primitives [33, 34].

2.6 Differentiation from prior studies

Despite this progress, early studies primarily focused on hybrid designs and low-level primitives. For example, [35] analyzed hybrid KEM-AKE protocols theoretically, whereas [21, 23] explored efficient Kyber implementations for embedded platforms. Although foundational, these studies did not specify the full-session layer protocol stacks, threat models, or benchmarked implementations.

Recent studies published between 2023 and 2025 have significantly expanded our empirical and architectural understanding of post-quantum secure protocols. [40] performed one of the first controlled measurements of PQC-enabled TLS 1.3 in Internet-scale testbeds and benchmarked Kyber and Dilithium handshakes. They demonstrated that post-quantum key exchange introduces less than 1 ms of additional delay on commodity hardware but does not address authenticated transcript binding or nonce management concerns. [41] extended this investigation by focusing on ML-DSA and Falcon signatures within TLS 1.3 authentication flights, providing a detailed latency analysis, but leaving open the question of forward secrecy under decapsulation failures. Klein Papanakis and Childs-Klein (2024) analyzed the impact of larger PQC payloads on the time-to-last-byte of data-heavy sessions and highlighted packet-fragmentation risks when using ML-KEM-1024, motivating further study of MTU-adaptive record sizing, such as that proposed in this work.

At the network transport layer, [42] benchmarked hybrid PQ-TLS deployments under real-world wide-area latency and packet loss scenarios, confirming the stability of Kyber-based handshakes across public Internet routes. [43] reported complementary results, showing that high-RTT environments amplify the cost of large PQ signatures, thereby emphasizing the need for streamlined transcript derivation and key reuse mitigation. [44] proposed a hybrid TLS 1.3 transition framework combining Kyber and ECDH, demonstrating backward compatibility while maintaining dependency on classical primitives. [45] shifted the focus below the transport layer by integrating hybrid Kyber key exchange within the IEEE 802.1AE MACsec protocol, proving that PQC primitives can secure link-layer authentication without compromising interoperability. Together, these studies illustrate the growing maturity of PQC deployment but underscore the absence of a complete, general-purpose session protocol independent of classical anchors.

From an implementation standpoint, [46] analyzed post-quantum algorithms across heterogeneous computing environments and found that Kyber-1024 and Dilithium-3 offer competitive performance even on embedded CPUs. [47] further examined PQ digital signatures in IoT platforms and identified memory and power constraints as key adoption barriers. [48] demonstrated a fully functional PQC protocol prototype at the application layer, focusing on the replay attack resistance and authenticated key update cycles. [49] extended the PQC paradigm to group key exchange, providing lattice-based forward secrecy among multiple participants, though limited to group-messaging contexts. [50] explored hybrid PQC integration within existing network stacks and concluded that although hybridization accelerates migration, it complicates security proofs owing to mixed-trust assumptions. [51] diversified this landscape by incorporating code-based schemes into TLS handshakes, thereby offering an alternative to lattice-based KEMs. However, they lacked a comparative evaluation of authenticated encryption under real network latencies.

Conceptual and theoretical developments paralleled these advances. [52] presented a foundational synthesis of quantum computing models, qubit logic, and cryptanalytic implications, reinforcing the urgency of migrating to quantum-resistant infrastructure. [53] analyzed the performance of the SARG04 quantum-key-distribution protocol within photonic quantum channels, providing a complementary perspective on physical-layer quantum security mechanisms. Although such studies emphasize true quantum communication approaches, they remain orthogonal to classical network deployment, where post-quantum cryptography offers the only practical defense in the near future.

Collectively, these studies revealed a vibrant research trajectory encompassing performance benchmarking, hybrid deployment, IoT adaptation, and theoretical framing. However, most existing efforts are either hybrid or domain-specific, rely on partial PQC integration, or lack unified handling of key schedule derivation, decapsulation failure resilience, and nonce-uniqueness guarantees. In contrast, the present work introduces a complete, authenticated post-quantum session protocol that composes ML-KEM (Kyber) and ML-DSA (Dilithium) within a transcript-bound HKDF framework, enforces unique 96-bit nonces per direction, and delivers empirical latency evaluation under both localhost and WAN conditions. This unified, reproducible architecture advances beyond the current state-of-the-art implementations by addressing the cryptographic, operational, and performance gaps identified in the recent literature.

Building on these broad system results, several studies have focused on domain-specific deployment and formalization. Recent studies (2023–2025) have shifted toward specific application domains. [54] presented a mapping study of hybrid KEMs but did not propose deployable frameworks for them. [55] and [56] explored secure messaging using CSIDH, Falcon, and Kyber, respectively, whereas [57] formalized the Signal PQXDH handshake using the BAKE model. However, these studies are limited to mobile and messaging platforms.

In networking and embedded domains, [58] and [59] analyzed PQC in 5G/6G mobile authentication protocols, and [60, 61] developed lattice-based authentication for IoT and e-health systems. [62] and [63] combined PQC with quantum key distribution (QKD) to build hybrid quantum-safe architectures. [64] demonstrated the practical integration of PQC in secure wireless systems.

Although these recent efforts reflect growing momentum, most are domain-specific, hybrid-only, or lack implementation benchmarks. Very few proposed general-purpose, reusable, and deployable session-layer protocols have integrated Kyber KEM with AES-256-GCM, which is complete with message flows, participant roles, threat modeling, and performance evaluation.

Our study addresses this gap by proposing a modular, crypto-agile, post-quantum session protocol tailored for high-security applications in the finance, healthcare, and government sectors. It includes a threat model targeting quantum adversaries, protocol design combining Kyber-1024 and AES-256-GCM, a reference Python implementation, and empirical performance benchmarks. This integrated approach supports immediate deployment and long-term post-quantum resilience.

While recent quantum-communication research has focused on photonic and physical-layer security primitives, such as QKD, optical qubit realization, and quantum-well devices for secure channels [43, 52, 53, 65], these approaches, which operate below the transport and session layers targeted in this work, typically require specialized quantum hardware and channels. Our protocol is complementary: it delivers algorithmic, software-deployable post-quantum security for authenticated key exchange and session encryption over classical networks, using NIST-standardized ML-KEM and ML-DSA without altering underlying infrastructure. In this regard, the proposed design bridges the gap between purely physical-layer quantum techniques and practical, Internet-scale cryptographic deployments, contributing a session-layer building block to an end-to-end quantum-resilient ecosystem.

2.7 Gap identification and motivation

Despite considerable progress in PQC, a clear gap persists in the development of deployable general-purpose session-layer protocols that securely integrate Kyber-based key exchange with symmetric encryption, such as AES-256-GCM.

- Algorithm-only focus: Foundational works, including NIST PQC evaluations [10, 11, 14, 15] and theoretical treatments of Kyber [16], primarily address security proofs, parameter tuning, and benchmarking of cryptographic primitives. Although essential, these contributions fall short of proposing fully realized, end-to-end secure communication protocols suitable for integration into critical infrastructure systems.
- Domain-specific implementations: Several studies target narrowly scoped environments, such as secure messaging [55, 57, 66], UAV systems [24], automotive control networks [25], IoT and e-health [60, 61], and wireless stacks [64]. These implementations are valuable but tailored to bespoke trust models and operational constraints, limiting their generalizability across industries such as finance, healthcare, and government.
- Hybrid-only or transitional designs: A significant portion of recent work, including Crypto Experimental Composite Post-quantum (CECPQ-1/2) experiments [38], hybrid TLS proposals [26], and 5G/6G security analyses [58, 59]—relies on hybrid key exchange models that combine classical cryptographic schemes with PQC to enable incremental migration. Although pragmatic, these systems remain dependent on classical primitives and do not fulfill the long-term need for fully quantum-resistant architectures.

Furthermore, even most implementation-focused studies, such as [21], [28], and [23], largely restrict their scope to the low-level performance of individual primitives (e.g., Kyber on ARM or FPGAs) and do not address how such primitives can be modularly assembled into practical and reusable session layer protocols.

To address these gaps, we introduce a modular and crypto-agile session protocol that combines Kyber-1024 KEM and AES-256-GCM encryption under a unified design, accompanied by a formal threat model, role-specific message flows, Python-based implementation, and empirical performance benchmarks. Unlike prior efforts, we did not limit our scope to a specific application domain, hybrid transitional model, or algorithmic benchmark. Instead, we present a general-purpose, end-to-end solution deployable across a range of high-security contexts that require both post-quantum robustness and practical implementation.

Unlike previous studies, which are often context-specific, our study provides a general protocol blueprint and open-source implementation that can be adapted to any application that requires quantum-resistant secure sessions. To the best of our knowledge, this is one of the first implementations to demonstrate full post-quantum key exchange combined with symmetric encryption at the session layer and performance evaluation against classical cryptography.

3 Proposed protocol design

Our protocol operates at the session layer and is not a TLS replacement; it can be deployed within TLS 1.3 or parallel channels, with confidentiality anchored in ML-KEM-derived keys and AES-256-GCM encryption.

3.1 Threat model

We consider well-resourced adversaries with i. access to large-scale quantum computers, ii. substantial classical computing capabilities, and iii. comprehensive network-level control. We specifically target HNDL adversaries—attackers who record ciphertext today for decryption after a cryptographic break—and counter this by establishing post-quantum confidentiality with ML-KEM-derived keys and AES-256-GCM. This reflects the realistic capabilities of nation-state actors and sophisticated criminal organizations targeting high-value domains such as finance, healthcare, and government platforms [4, 8].

3.2 Adversary capabilities

- Passive eavesdropping: monitoring all communication between client and server over public or private networks without detection.
- Active man-in-the-middle (MitM) attacks: intercepting, modifying, or replaying messages to subvert key exchange protocols or inject malicious payloads.
- Cryptanalysis using classical algorithms: leveraging state-of-the-art classical cryptanalytic techniques against poorly implemented or weak protocols.

- Quantum cryptanalysis: applying algorithms such as Shor's algorithm to break RSA and ECDH key exchanges once practical quantum computers become available [1].

3.2.1 Forward secrecy assumption

We assumed that the endpoints were not compromised during active sessions. Forward secrecy is preserved if ephemeral Kyber key pairs are generated per session and securely erased after the session is terminated. In other words, compromising long-term credentials after termination does not reveal prior-session keys if ephemeral secrets are erased.

A particularly serious risk is the harvest-now–decrypt-later strategy, in which adversaries record encrypted sessions today with the intention of decrypting them in the future when quantum resources are sufficient [4]. This is especially problematic for systems that handle sensitive, long-lived data, such as medical records, financial transactions, and classified government communication.

Kyber's Fujisaki–Okamoto (FO) CCA transform and the use of per-session ephemeral ML-KEM key pairs ensure that the compromise of long-term authentication keys or later endpoint compromise does not retroactively reveal past traffic keys, provided that the ephemeral secrets are erased.

3.3 Scope and assumptions

Endpoints are assumed to be uncompromised beyond the secure erasure of per-session ML-KEM private keys, and side-channel and physical attacks are beyond the scope of the proposed protocol. The design operates at the session layer and is not a TLS replacement; it is composed of TLS 1.3 or runs in parallel as a secure channel.

- Network control. The adversary has full access to the communication channel (eavesdrops, modifies, replays, drops, and reorders the data).
- Endpoint compromise. The adversary cannot compromise server or client software/hardware during active sessions; post-session secure erasure of per-session ML-KEM private keys is enforced.
- Side channels. Side-channel attacks (e.g., power analysis and timing) are out of scope for protocol design but are critical for deployment; thus, constant-time masked implementations are recommended [21, 23].
- Authentication material. Long-term ML-DSA public keys were distributed via a trusted PKI, and ML-KEM keys were generated for each session and authenticated during the handshake.
- Nonce policy. AES-256-GCM uses 96-bit nonces, single-use per (key, direction), with rekeying well before 2^{32} records, per NIST SP 800-38D; AES-256-GCM-SIV (RFC 8452) is an option in which nonce misuse risk exists.
- Randomness. Endpoints have access to a CSPRNG that is suitable for generating keys, nonces, and salts.

3.4 Security objectives

- Confidentiality: ensuring session keys and payload data remain secret even in the presence of quantum-enabled adversaries.
- Integrity: preventing undetected message tampering or replay.
- Forward secrecy against quantum adversaries: ensuring that even if future quantum computers become available, previously recorded sessions remain secure.

This threat model was designed to match the realistic adversarial conditions faced by modern high-security applications, highlighting the urgent need for protocols that can resist both classical and quantum attacks while maintaining efficiency and deployability in real-world systems [9, 11].

3.5 Design goals

The objective is to develop a secure, efficient, and deployable session-layer key exchange and encryption mechanism for high-security postquantum settings. We defined explicit goals (G1–G7) to guide the architectural design.

1. G1—Post-quantum confidentiality (HNDL resistance).
    - Establish confidentiality at session start using ML-KEM (FIPS 203; formerly CRYSTALS–Kyber) in place of classical DH/ECDH.
    - Ensure that the ciphertext recorded today remains infeasible to decrypt later (harvest-now-decrypt-later, HNDL) [4].
2. G2—Integrity and authenticity (authenticated encryption with associated data, AEAD).
    - Provide authenticated encryption for payloads using AES-256-GCM per NIST SP 800-38D.
    - Where nonce misuse risk exists, allow AES-256-GCM-SIV (RFC 8452) as a misuse-resistant alternative.
3. G3—Forward secrecy against quantum adversaries.
    - Use ephemeral ML-KEM key pairs per session and securely erase private keys on termination so that compromise of long-term keys does not reveal past session keys [8].

Forward secrecy is guaranteed because every session key is derived from ephemeral ML–KEM encapsulation that is independent of long-term credentials. Even a future compromise of the ML–KEM private state or server certificate cannot expose the previously established session keys.

1. G4—Nonce safety and rekeying.
    - Enforce 96-bit nonces, single-use per (key, direction).
    - Rekey well before 2^{32} records to maintain GCM security bounds (SP 800-38D).
2. G5—Efficiency and deployability.
    - Minimize computation/latency to run on commodity hardware without degrading throughput.

- Compose at the session layer (not a TLS replacement); integrate within TLS 1.3 or as a parallel secure channel.

3. G6—Modularity and crypto-agility.

   - Modular HKDF–SHA3-256-based key schedule; clean separation of authentication (ML-DSA, FIPS 204) and key exchange (ML-KEM).
   - Support migration to updated PQC standards or hybrid modes as they are standardized [26, 33].

4. G7—Realistic threat alignment and assumptions.

   - Model quantum-enabled, active network adversaries (eavesdrop, modify, and replay).
   - Endpoint compromise and side-channel attacks are out of scope for the protocol but must be mitigated in implementations (constant-time, masking) [21, 23].

By satisfying G1–G7, the proposed design replaces quantum-vulnerable key exchange, preserves authenticity and forward secrecy, and remains deployable in latency-sensitive environments.

3.6 System architecture

We propose a modular system architecture that integrates post-quantum key exchange with conventional symmetric encryption to enable quantum-resistant secure communication between a client and a server. The design separates the key establishment and encrypted data exchange phases of the protocol.

Figures 1, 2 depict the same four-step handshake and data phase: 1. Amber—PKI path: the client validates the server's ML-DSA certificate; an ephemeral ML-KEM public key is conveyed and authenticated in the handshake. 2. Purple—the client sends the ML-KEM (Kyber) ciphertext. 3. Teal (dashed)—both sides run transcript-bound HKDF–SHA3-256 to derive `traffic_key` and `traffic_iv`. 4. Green—bidirectional AES-256-GCM records with 96-bit nonce $nonce_i = base\_iv \oplus (0^{32} \| \mathrm{be64}(i))$, with rekeying well before 2^{32} records. Per-session ephemeral Kyber key pairs (securely erased) provide forward secrecy, and ML-DSA provides endpoint authentication. We enforce a strict no-reuse policy for nonces and rekey well before counter exhaustion; counters never wrap, and any attempt to exceed the budget aborts the connection.

- Participants: Two primary nodes—client and server—engage in secure communication over an untrusted network. The server holds a long-term ML-DSA certificate.
- Ephemeral ML-KEM key: At session start, the server generates an ephemeral ML-KEM key pair and conveys the public key in the handshake, authenticated by the ML-DSA certificate (e.g., a signature over the transcript).
- Kyber encapsulation: The client uses the server's ephemeral ML-KEM public key to perform encapsulation, generating both a shared secret (for HKDF–SHA3-256) and a Kyber ciphertext.
- Ciphertext transmission: The client sends the Kyber ciphertext to the server, which decapsulates it using its private key to derive the same AES-256 session key.
- Encrypted channel: Once both parties hold the shared session key, they establish an AES-256-GCM encrypted channel for secure, authenticated data transmission.

Figure 1 illustrates the process. The initial exchange ensures post-quantum resistance using the Kyber KEM for secure key establishment. The derived AES-256 session key is then used for the high-performance authenticated encryption of transaction data over the established channels. This hybrid approach combines the forward secrecy and quantum resistance of lattice-based cryptography with the efficiency of symmetric encryption, making it suitable for high-security applications such as financial transactions, healthcare data sharing, and government communications.

3.7 Operational flow overview

The operational flow of the proposed authenticated post-quantum session protocol is designed to mirror the logical structure of contemporary TLS 1.3 handshakes while replacing classical primitives with their quantum-resistant counterparts. As presented in Table 2, the handshake proceeds through a sequence of authenticated exchanges that collectively establish a secure channel over the reliable transport layer. The server first authenticates itself using an ML-DSA (Dilithium) signature, after which an ephemeral ML-KEM (Kyber) key exchange allows both parties to derive a shared symmetric key. The resulting session keys were expanded using transcript-bound HKDF–SHA3-256 to produce AES-256-GCM traffic keys and initialization vectors for encrypted communications. The process consists of four major phases: initialization, key exchange, key derivation, and secure data transmission, which are described in the following sections.

1. Initialization: The server presents its long-term ML-DSA certificate. During the handshake, an ephemeral ML-KEM public key is generated, conveyed, and authenticated using the ML-DSA signature (transcript-bound).
2. Key exchange: The client encapsulates the server's ephemeral ML-KEM public key, producing a Kyber ciphertext.
3. Key derivation: Both sides compute a shared secret via decapsulation/encapsulation and run transcript-bound HKDF–SHA3-256 to derive `traffic_key` and `base_iv`.
4. Secure communication: Each record uses AES-256-GCM with nonce $nonce_i = base\_iv \oplus (0^{32} \| \mathrm{be64}(i))$, with rekeying well before 2^{32} records.

3.8 Deployment considerations

- The architecture is designed to be crypto-agile, allowing the replacement of cryptographic primitives as new post-quantum standards emerge.
- Existing PKI distributes long-term ML-DSA (signature) public keys. The ML-KEM public key is generated per session and

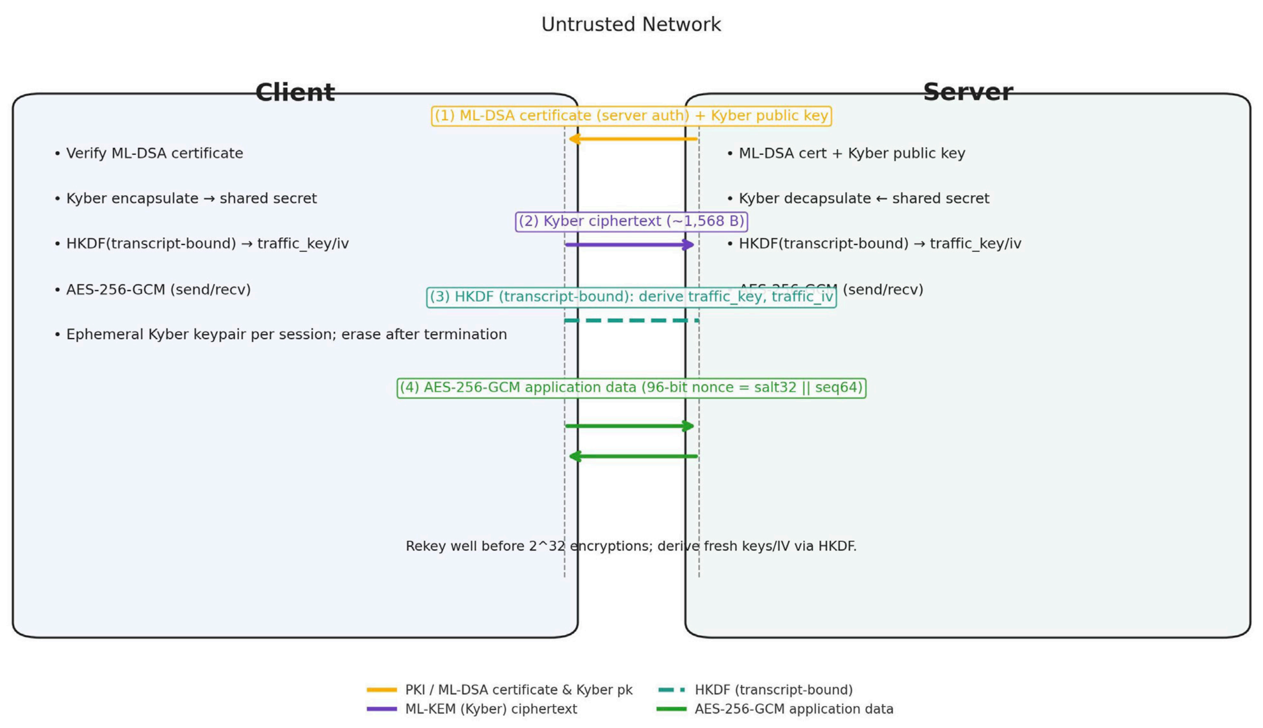
Untrusted Network
Client
Server
(1) ML-DSA certificate (server auth) + Kyber public key
• Verify ML-DSA certificate
• Kyber encapsulate → shared secret
• HKDF(transcript-bound) → traffic_key/iv
• AES-256-GCM (send/recv)
• Ephemeral Kyber keypair per session; erase after termination
• ML-DSA cert + Kyber public key
• Kyber decapsulate ← shared secret
• HKDF(transcript-bound) → traffic_key/iv
• AES-256-GCM (send/recv)
(2) Kyber ciphertext (~1,568 B)
(3) HKDF (transcript-bound): derive traffic_key, traffic_iv
(4) AES-256-GCM application data (96-bit nonce = salt32 || seq64)
Rekey well before 2^32 encryptions; derive fresh keys/IV via HKDF.
PKI / ML-DSA certificate & Kyber pk
ML-KEM (Kyber) ciphertext
HKDF (transcript-bound)
AES-256-GCM application data

FIGURE 1
System architecture of the proposed quantum-resistant protocol.

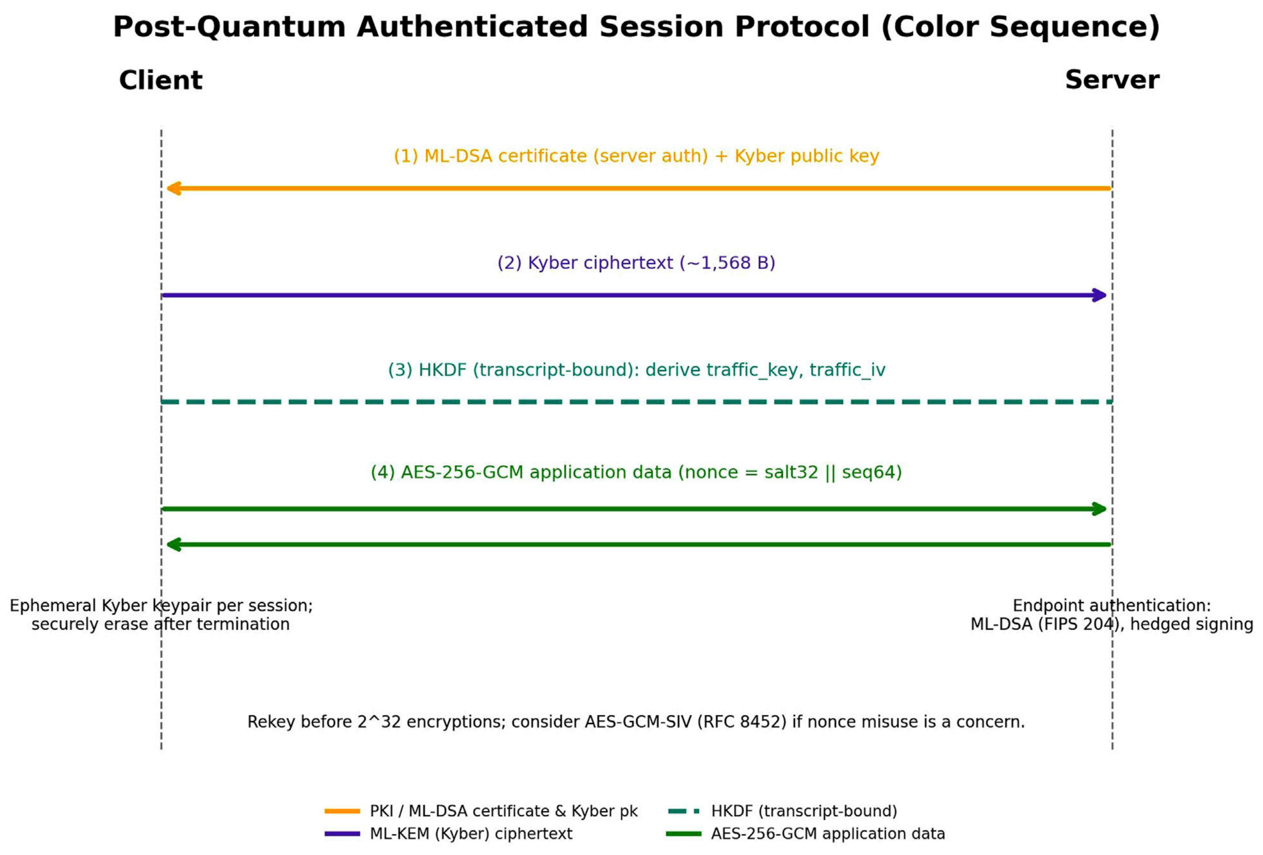
Post-Quantum Authenticated Session Protocol (Color Sequence)
Client
Server
(1) ML-DSA certificate (server auth) + Kyber public key
(2) Kyber ciphertext (~1,568 B)
(3) HKDF (transcript-bound): derive traffic_key, traffic_iv
(4) AES-256-GCM application data (nonce = salt32 || seq64)
Ephemeral Kyber keypair per session; securely erase after termination
Endpoint authentication: ML-DSA (FIPS 204), hedged signing
Rekey before 2^32 encryptions; consider AES-GCM-SIV (RFC 8452) if nonce misuse is a concern.
PKI / ML-DSA certificate & Kyber pk
ML-KEM (Kyber) ciphertext
HKDF (transcript-bound)
AES-256-GCM application data

FIGURE 2
Protocol flow for quantum-resistant key exchange and encryption.

TABLE 2 End-to-end flow of the proposed post-quantum session protocol.

Step	Action	Component	Purpose
1	The server sends certificate and signature	ML-DSA (Dilithium)	Server authentication and identity proof.
2	The client verifies the server's signature	ML-DSA (Dilithium)	Establishes trust in the server's authenticity
3	The server transmits Kyber public key	ML-KEM (Kyber)	Prepares for post-quantum key exchange.
4	The client encapsulates the shared secret and sends the ciphertext	ML-KEM (Kyber)	Securely transfers the ephemeral shared key material
5	The server decapsulates the ciphertext to recover shared secret	ML-KEM (Kyber)	Derives the same symmetric session key
6	Both sides derive session keys via HKDF–SHA3-256	AES-256-GCM	Generates encryption keys for a secure channel.
7	Encrypted communication begins	AES-256-GCM	Ensures confidentiality and integrity of all data

authenticated within a handshake, which does not require an out-of-band distribution.

- The session key derived from Kyber is used solely for AES encryption of application-layer payloads, ensuring minimal changes to existing transaction processing logic while upgrading the security of the key exchange mechanism.

3.9 Endpoint authentication with ML-DSA (FIPS 204)

We adopted the NIST's Module-Lattice-Based Digital Signature Algorithm (ML-DSA; formerly CRYSTALS–Dilithium) [72]. ML-DSA is standardized in FIPS 204 with three parameter sets: ML-DSA-44, ML-DSA-65, and ML-DSA-87 [73]. We targeted ML-DSA-65 as the default profile for high-security deployments, with ML-DSA-87 available where higher margins were required. The corresponding key and signature sizes are as follows.

- Public key: 1,312 B (ML-DSA-44), 1,952 B (ML-DSA-65), and 2,592 B (ML-DSA-87)
- Secret key: 2,560 B, 4,032 B, and 4,896 B
- Signature: 2,420 B, 3,309 B, and 4,627 B

(values per FIPS 204 and widely implemented [73, 74]). We used the hedged signing variant specified by FIPS 204 to combine deterministic hashing with the RNG input for robustness against degraded entropy sources. Certificates and certificate chains using ML-DSA increase handshake flight size. Our transport guidance (Section 3) already accounts for the MTU-aware segmentation and record size. We note the ongoing IETF LAMPS work defining the ML-DSA in X.509 to support interoperability with PKI ecosystems [75].

3.9.1 Diagram conventions and flow mapping

Figures 1, 2 use a unified legend:

- Amber (solid): PKI/public-key distribution (server authenticated by an ML-DSA certificate); the handshake conveys an ephemeral ML-KEM public key that is signed/bound to the transcript.
- Purple (solid): ML-KEM (Kyber) ciphertext from client→server (size ≈ 1,568 B for ML-KEM-1024).
- Teal (dashed): transcript-bound HKDF–SHA3-256 (extract/expand over the KEM shared secret and transcript hash) to derive `traffic_key` and `traffic_iv`.
- Green (solid, two-way): AES-256-GCM application data with 96-bit nonce $nonce_i = base\_iv \oplus (0^{32} \| \mathrm{be64}(i))$; rekey well before 2^{32} records.

Mapping to the numbered flow in Section 3:

1. Amber: server certificate/ML-DSA chain distribution and validation.
2. Purple: client encapsulates and sends the ML-KEM ciphertext c.
3. Teal: both sides run HKDF–SHA3-256 with transcript binding to derive keys/initiation vectors (IVs).
4. Green: bidirectional AES-256-GCM records for application data.

3.10 Protocol steps

Figure 2 illustrates the step-by-step message flow for the proposed quantum-resistant key exchange and encryption protocol between the client and server. This flow highlights the precise order of operations required to establish a secure, post-quantum-resistant session key and enable the authenticated encryption of transaction data.

Step 1: Authenticate the server. The client validates the server's long-term ML-DSA certificate; the handshake conveys an ephemeral ML-KEM public key bound to the transcript.
Step 2: Kyber encapsulation. The client encapsulates the server's ephemeral ML-KEM public key to obtain (c, ss).
Step 3: Send the Kyber ciphertext. The client sends the Kyber ciphertext c to the server.
Step 4: Shared secret and keys. The server decapsulates c to recover ss; both sides run transcript-bound HKDF–SHA3-256 to derive `traffic_key` and `base_iv`.
Step 5: AES-256-GCM secure communication. Both sides send application data using AES-256-GCM with per-record

nonces $nonce_i = base\_iv \oplus (0^{32} \| \text{be64}(i))$; rekey well before 2^{32} records.

If ML-KEM decapsulation fails, the session is immediately aborted, and all ephemeral secrets are zeroized. A fresh handshake is required, and no key material is ever reused.

This protocol flow ensures that the initial key exchange is resistant to quantum attacks by relying on Kyber KEM's post-quantum security, whereas subsequent data transmission benefits from the efficiency and authenticated encryption guarantees of AES-256-GCM. By cleanly separating key establishments from data encryption, this design enables practical deployment in high-security domains and future-proofing against quantum-capable attackers.

3.11 Handshake composition logic

Figure 2 formally represents the sequence of cryptographic operations in the proposed authenticated post-quantum-session protocol. The flow defines the complete message exchange between the client (C) and server (S), including ephemeral ML-KEM key generation, encapsulation/decapsulation, ML-DSA signature verification, and AES-256-GCM symmetric-key establishment. This sequence also captures mutual authentication, forward-secrecy guarantees, and uniform failure-handling behavior, thereby serving as the protocol's formal composition-logic diagram.

3.12 Security model and proof sketch

We analyze the proposed handshake in an Authenticated and Confidential Channel Establishment (ACCE)/Authenticated Key Exchange (AKE) style channel model. Let Π denote our protocol, instantiated as follows. i. an IND-CCA-secure KEM (ML–KEM, standardized in FIPS 203 [13, 14]), ii. an EUF-CMA-secure signature scheme (ML–DSA, standardized in FIPS 204 [72, 73]) for endpoint authentication, iii. an HKDF–SHA3-256-based key schedule modeled as a PRF, and iv. an AEAD scheme (AES–256–GCM)-proven IND-CPA and INT-CTXT secure under the standard nonce-respecting definition [76].

All traffic secrets were derived using transcript-bound HKDF–SHA3-256 labels. Each label incorporates a complete handshake transcript, including algorithm identifiers, version codes, ephemeral ML–KEM public keys, and ML–DSA certificates and signatures. This binding ensures that the resulting traffic keys are uniquely tied to the negotiated configuration and prevents downgrade and cross-protocol substitution attacks against both classical and quantum-capable adversaries.

For each direction (client-to-server and server-to-client), we derived non-overlapping nonce spaces from a 96-bit base IV and monotonically increasing per-record counter. Nonces are computed as

$$nonce_i = base\_iv \oplus (0^{32} \| \text{be64}(i)),$$

where i is the 64-bit record counter. This construction yields unique nonces per (key, direction), even under concurrent or parallelized implementations, and therefore upholds the security assumptions of the AES-GCM in multicore environments [76].

3.13 Informal theorem

If ML-KEM is IND-CCA secure, ML-DSA is EUF-CMA secure, HKDF–SHA3-256 behaves as a PRF, and AES-256-GCM is IND-CPA/INT-CTXT secure under unique nonces, then the channel established by Π provides i. confidentiality and integrity of application data, ii. server authentication (and client authentication when the client-authenticated variant is used), and iii. forward secrecy, assuming per-session ephemeral ML-KEM key pairs with secure erasure.

3.14 Proof sketch

Server impersonation reduces to forging an ML-DSA signature on the handshake transcript, including the ephemeral ML-KEM public key; this is ruled out by EUF-CMA security [72, 73]. The session key indistinguishability reduces to the IND-CCA security of the ML-KEM in its Fujisaki–Okamoto (FO) transformed form [14], where implicit rejection prevents the construction of decryption oracles. Given a pseudorandom shared secret, the transcript-bound HKDF–SHA3-256 (modeled as a PRF) yields traffic keys that are indistinguishable from random keys. AEAD confidentiality and integrity then follow from AES-GCM security under unique nonces [76]. Forward secrecy is obtained because (a) each session uses a freshly generated ephemeral ML-KEM key pair and (b) the corresponding private key and intermediate secrets are securely erased at the end of the handshake. Thus, even if long-term ML-DSA keys are later compromised, past traffic keys remain hidden.

Beyond the guarantees of the FO transform, the authenticated composition of our protocol achieves CCA2-style resilience at the channel level by binding ciphertext validity to the signed transcript hash and enforcing AEAD verification before accepting any application data. Adversarially modified ciphertexts or replayed handshake messages cannot be turned into a decryption oracle because all such attempts fail at signature verification, decapsulation consistency, or AEAD key confirmation without revealing information about valid session keys.

3.15 Ephemeral key lifecycle and secure erasure

In the reference implementation, ephemeral ML-KEM-1024 key pairs are generated on a per-session basis and used exactly once for the corresponding handshake. Immediately after the transcript-bound HKDF–SHA3-256 key schedule derives the final traffic secrets, all ephemeral private keys, shared secrets, and intermediate buffers are securely erased using constant-time memory overwrites; no lazy garbage-collection semantics are relied upon. The codebase exposes explicit sanitization hooks (e.g., `kem_zeroize()` and `session_wipe()`) that are covered by unit tests to verify that the relevant buffers are wiped, making forward secrecy

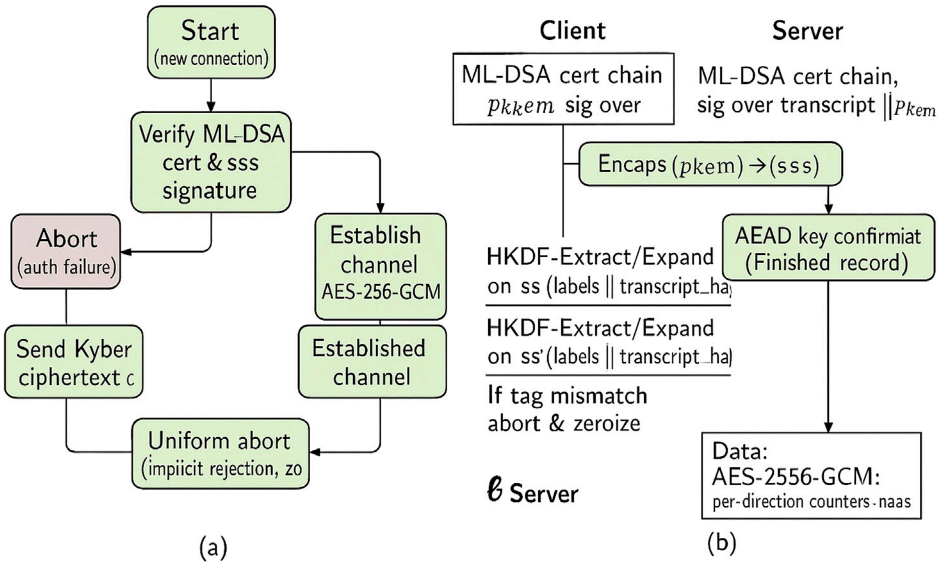
Start
(new connection)
Verify ML-DSA cert & sss signature
Abort (auth failure)
Establish channel AES-256-GCM
Send Kyber ciphertext c
Established channel
Uniform abort (impiicit rejection, zo
(a)
Client
Server
ML-DSA cert chain pkkem sig over
ML-DSA cert chain, sig over transcript ||pkem
Encaps (pkem) → (sss)
AEAD key confirmiat (Finished record)
HKDF-Extract/Expand on ss (labels || transcript_ha
HKDF-Extract/Expand on ss' (labels || transcript_ha
If tag mismatch abort & zeroize
Server
Data: AES-2556-GCM: per-direction counters-naas
(b)

FIGURE 3
Integrated view of the authenticated ML-KEM-1024 + ML-DSA-65 session protocol. Left panel: handshake state machine with uniform abort and zeroization logic; right panel: client–server message flow with transcript-bound HKDF–SHA3-256, AEAD key confirmation, and established AES-256-GCM channel. The implementation was validated under both local and WAN-emulated (≈ 40 ms RTT) network conditions, confirming stable handshake timing and throughput. **(a)** Handshake state machine with robust, uniform failure semantics. **(b)** Compact swimlane message flow; integrated view of the authenticated ML-KEM1024+ ML-DSA6 session. **(a)** Robust failure semantics (all errors → uniform abort + zeroization; no key/nonce reuse). **(b)** Corresponding message flow.

enforcement auditable in real deployments. This turns forward secrecy from a purely conceptual property into an operational guarantee: later compromise of long-term ML-DSA credentials or server state does not reveal past traffic keys, provided that the ephemeral secrets were successfully erased at the end of each handshake.

3.15.1 Handshake: ML-DSA-authenticated ML-KEM (wire logic)

The end-to-end handshake process and its security-relevant transitions are illustrated in Figure 3. This integrated diagram visualizes both the handshake state machine and client–server message flow, providing an intuitive view of the operations analyzed in the subsequent steps.

The left panel of Figure 3 illustrates the internal handshake state machine, highlighting encapsulation, signature verification, AEAD key confirmation, and uniform aborting with zeroization. The right panel captures the corresponding client–server message exchange bound to the transcript hash and HKDF–SHA3-256 derivation chain. Together, these views provide a unified picture of how authentication, confidentiality, and forward secrecy are enforced throughout the protocol.

The core handshake flow is as follows.

1. ServerHello (authentication and KEM key distribution). The server sends i. its ML-DSA certificate chain (or pinned ML-DSA public key), ii. an ephemeral ML-KEM public key pk_{kem}, and iii. an ML-DSA signature $\sigma = \mathrm{Sign}(sk_{dsa},\ transcript \| pk_{kem})$ in a handshake context.
2. Client verification. The client validates the ML-DSA certificate chain and verifies σ over $(transcript \| pk_{kem})$. Failure aborts the handshake.
3. Client encapsulation. The client computes $(c, ss) = \mathrm{Encaps}(pk_{kem})$ and sends the ML-KEM ciphertext c.
4. Server decapsulation. The server computes $ss = \mathrm{Decaps}(sk_{kem}, c)$ using its ephemeral secret key.
5. Key schedule. Both parties compute $PRK = \mathrm{HKDF-SHA3-256-Extract}(\mathrm{salt} = 0, ss)$ and derive per-direction traffic keys and base IVs via $\mathrm{HKDF-SHA3-256-Expand}$ with domain-separated labels and $transcript\_hash$ covering all the handshake
6. Data protection. Application records are protected with AES-256-GCM using the derived traffic keys and the XOR-based nonce construction $nonce_i = base\_iv \oplus (0^{32} \| \mathrm{be64}(i))$. Rekeying or session renegotiation is mandatory well before 2^{32} records are generated per key.

This design mirrors the well-understood TLS 1.3 key schedule [39] while replacing classical ECDHE and ECDSA with standardized post-quantum ML-KEM and ML-DSA, preserving analyzability and interoperability.

3.15.2 Transcript-bound HKDF–SHA3-256 and directional key separation

After decapsulation, both endpoints derive traffic secrets using transcript-bound HKDF–SHA3-256 as follows:

$$PRK = \mathrm{HKDF-Expand-SHA3-256-Extract}(0, ss),$$

where *ss* is the shared secret and *transcript_hash* commits to all handshake messages, including certificates, signatures, and pk_{kem}.

Direction-specific keys and IVs are derived using distinct labels as follows:

$$\begin{aligned}
k_{c2s} &= \mathrm{HKDF} - \mathrm{Expand} - \mathrm{SHA3} - 256 - \mathrm{Expand}(PRK, \\
&\quad \times \mathrm{key}_c 2\mathrm{s} \| transcript\_hash,\ 32), \\
v_{c2s} &= \mathrm{HKDF} - \mathrm{Expand} - \mathrm{SHA3} - 256 - \mathrm{Expand}(PRK, \\
&\quad \times \mathrm{iv}_c 2\mathrm{s} \| transcript\_hash,\ 12), \\
k_{s2c} &= \mathrm{HKDF} - \mathrm{Expand} - \mathrm{SHA3} - 256 - \mathrm{Expand}(PRK, \\
&\quad \times \mathrm{key}_s 2\mathrm{c} \| transcript\_hash,\ 32), \\
v_{s2c} &= \mathrm{HKDF} - \mathrm{Expand} - \mathrm{SHA3} - 256 - \mathrm{Expand}(PRK, \\
&\quad \times \mathrm{iv}_s 2\mathrm{c} \| transcript\_hash,\ 12).
\end{aligned}$$

To record index *i* in each direction, we have

$$\begin{aligned}
\mathrm{nonce}_i^{c2s} &= v_{c2s} \oplus (0^{32} \| \mathrm{be64}(i_{c2s})), \\
\mathrm{nonce}_i^{s2c} &= v_{s2c} \oplus (0^{32} \| \mathrm{be64}(i_{s2c})).
\end{aligned}$$

The counters are monotonic, direction-local, and never reused. The detection of reuse, wraparound, or rollback is treated as a fatal error. For deployments where strict nonce discipline cannot be guaranteed, AES-256-GCM-SIV [77] can be adopted as a compatible nonce-misuse-resistant AEAD without altering the handshake design.

3.16 Decapsulation failure handling and side-channel safety

Decapsulation failures for the ML–KEM were handled using the implicit rejection paradigm of Kyber's FO transform [14, 23]. The implementation is executed in constant time using compare-and-select logic, avoiding secret-dependent branches and lookup tables.

For invalid ciphertexts, a pseudorandom fallback key is derived such that the control flow and timing remain indistinguishable from those of successful decapsulation. The handshake proceeds to the AEAD key-confirmation step; any mismatch in the derived keys causes a uniform session abortion.

3.17 Uniform failure behavior

- No detailed error codes are exposed; failures are signaled using a generic alert indistinguishable from other handshake errors.
- All ephemeral keys, intermediate states, and candidate traffic secrets are securely zeroized upon abort.
- Ephemeral pk_{kem} and corresponding nonces are never reused across handshakes.

3.18 Integration with ML-DSA verification

ML-DSA verification of the server's (and, optionally, the client's) identity over the full transcript, including pk_{kem}, was performed before accepting any application data. Signature checks are deterministic public-key operations that are independent of KEM success and prevent attacker-controlled interactions between signature validation and decapsulation logic. The final key confirmation via AEAD-protected finished messages ensures that both parties hold matching traffic keys, thereby yielding implicit mutual authentication.

3.19 Side-channel considerations

All critical operations (ML-KEM decapsulation, ML-DSA signing/verification, HKDF–SHA3-256, and AES-GCM) were implemented using constant-time side-channel-hardened libraries following published best practices [21, 23]. This mitigates timing and cache-based leakage and aligns with the recommended implementation guidelines for NIST-standardized lattice-based schemes.

4 Implementation and simulation

This section presents the experimental execution and benchmarking of the proposed post-quantum secure-communication protocol. Table 3 provides the tools utilized, simulation builds, execution platform, and empirical results of the cryptographic performance of the protocol. This study was conducted to analyze its fit for adoption and deployment in latency-constrained, high-security settings.

We present the handshake latency distributions, sustained handshake rates, and ECDFs under localhost and WAN-emulated settings (Figure 4). The benchmarking harness (`pq_benchmark.py`) was executed under both local (0 ms RTT) and WAN-like conditions (≈ 40 ms RTT with ±5 ms jitter), emulated using the Linux `tc netem` facility within WSL 2 on Windows 11.

Under these controlled environments, ML-KEM-1024 encapsulation and decapsulation achieved mean execution times of 0.69 and 0.56 ms, respectively, while AES-256-GCM encryption and decryption averaged 0.05 and 0.07 ms, respectively. The classical baselines exhibited significantly higher costs, with X25519 ECDH at 2.81 ± 23.44 ms and RSA-3072 handshake at 733.05 ± 264.28 ms under the same conditions. The near-identical post-quantum results across local and WAN-emulated settings confirm that the protocol's cryptographic processing overhead remains sub-millisecond and largely network-independent. We then present the microbenchmarks and data plane throughput, including the KEM/signature costs and AES-GCM throughput (Figure 5). The integrated performance and handshake visualization, which correlate protocol state transitions with observed latency and throughput behavior, are shown in Figure 3. Finally, we present the aggregated message and artifact sizes and offer conservatively estimated rekeying advice (Figure 6).

Table 8 summarizes the comparative results between the local (0 ms RTT) and WAN-emulated (≈ 40 ms RTT) conditions obtained using the same benchmarking harness. The measurements demonstrate that post-quantum primitives, including $\mathrm{ML} - \mathrm{KEM}_{1024}$ encapsulation/decapsulation and AES-256-GCM operations, retain sub-millisecond performance and remain practically invariant to network latency. Classical baselines such as X25519 and RSA-3072 exhibit higher variability and

TABLE 3 Method transparency: cryptosystem, workload, transport, and hardware/software settings for the PQC simulations and baselines.

#	Category	Setting (value)
Cryptosystem (handshake and data protection)		
1	KEM (key exchange)	ML-KEM-1024 (FIPS 203; Kyber–1024), PQClean C implementation (called via ctypes)
2	Signature (authentication)	ML-DSA (Dilithium), target level ML-DSA-65. Sizes (FIPS 204): public key 1952 B, secret key 4032 B, and signature 3293 B
3	Symmetric AEAD	AES-256-GCM; tag = 16 B; optional explicit 12 B nonce if carried on wire
4	Key schedule	HKDF–SHA3-256-extract/expand on KEM-shared secret; transcript-bound labels; per-direction key_c2s/s2c, iv_c2s/s2c
5	Nonce derivation	96-bit GCM nonce: $nonce_i = base\_iv \oplus (0^{32} \| \mathrm{be64}(i))$(single use per key, direction)
6	Rekey policy	Rekey well before 2^{32}records per traffic key (NIST SP 800-38D); AES-256-GCM-SIV (RFC 8452) as misuse-resistant alternative
Workload/metrics		
7	Iterations (per op)	1000 runs for each: KEM encaps/decaps; AES-256-GCM enc/dec (1 KB records)
8	Record size (data plane)	1 KB per AES-GCM record for microbenchmarks
9	Reported metrics	Mean and SD (ms) per op; comparative baselines for X25519 and RSA–3072
Observed timings (from manuscript tables)		
10	Kyber-1024 encaps (raw)	0.22 ± 0.04 ms
11	Kyber-1024 decaps (raw)	0.17 ± 0.03 ms
12	AES-256-GCM enc 1 KB	0.03 ± 0.005 ms
13	AES-256-GCM Dec 1 KB	0.03 ± 0.004 ms
14	Kyber encapsulation (baseline)	≈ 0.66 ms (harness-inclusive)
15	Kyber decapsulation (baseline)	≈ 0.64 ms (harness-inclusive)
16	X25519 ECDH (baseline)	≈ 0.67 ms
17	RSA-3072 "handshake"	≈ 238.18 ms (incl. keygen upper bound)
Transport and sizes (from size figures/text)		
18	Kyber public key	1568 B (→ ~2 Ethernet MTU segments at 1500 B)
19	Kyber secret key	3168 B
20	Kyber ciphertext	1568 B (→ ~2 Ethernet MTU segments at 1500 B)
21	AES-GCM tag	16 B per record
22	Explicit nonce (optional)	12 B per record (if sent on wire)
Benchmark environment		
23	Host OS/kernel	Windows 10 pro (64-bit), build 22H2; WSL2 enabled; Docker Desktop 4.x with Linux container backend (Ubuntu 22.04 LTS)
24	Containerization	Docker Desktop (hyper-V/WSL2 backend), default resource allocation: 4 vCPUs, 8 GB RAM assigned to containers
25	Python/environment	Python 3.11 (CPython); virtualenv for dependency isolation; executed inside Ubuntu 22.04 Docker container
26	Crypto libraries	PQClean ML-KEM-1024 (Kyber-1024, C implementation accessed via ctypes); PyCryptodome v3.20 for AES-256-GCM; hashlib (SHA3) from python stdlib

(Continued on the following page)

TABLE 3 (*Continued*) Method transparency: cryptosystem, workload, transport, and hardware/software settings for the PQC simulations and baselines.

#	Category	Setting (value)
27	Compiler/build toolchain	GCC 11.4 (Ubuntu toolchain) with -O3 optimization for building PQClean shared library, compiled as the shared object libpqkem.so and bound into Python
28	Supporting tool	Docker Desktop CLI, docker-compose for container orchestration; timeit and perf_counter (python) for timing; NumPy/Pandas v2.0 for CSV aggregation; and Matplotlib v3.8 for figures
29	Hardware (host)	Intel Core i7-12700K (3.6 GHz base, 4.5 GHz turbo), 6 cores/12 threads, 32 GB DDR4 RAM; no GPU or hardware cryptographic accelerators
30	Scheduling	Single-process execution inside container; default Linux CFS scheduler (Docker/WSL2 environment); background processes minimized
31	Timing harness	High-resolution timers (time.perf_counter_ns) with warm-up runs; baseline includes python marshalling and docker containerization overhead
Figure panel linkage		
28	Group 1 (Figure 4)	pqc_handshake_latency.png, handshakes_per_sec.png, ecdf_localhost.png, ecdf_wan.png
29	Group 2 (Figure 5)	kem_micro.png, sig_bench.png, throughput_vs._size.png, cpu_rss_hybrid.png
30	Group 3 (Figure 6)	handshake_size_totals.png, handshake_size_breakdown.png, cert_key_sizes.png, pqc_rekey_budget.png
Reproducibility notes		
31	Determinism	KEM randomness from library RNG; report mean/SD over 1000 runs to smooth OS jitter
32	Availability	Prototype: Python wrapper over PQClean Kyber; PyCryptodome AEAD; grouped PNG composites for submission
Empirical values from CSV (this submission)		
33	Client latency (classic, localhost)	Mean 1.49 ms (bench_classic_clientlat.csv)
34	Client latency (classic, WAN)	Mean 37.48 ms (bench_classic_wan_clientlat.csv)
35	Client latency (hybrid, localhost)	Mean 1.51 ms (bench_hybrid_clientlat.csv)
36	Client latency (hybrid, WAN)	Mean 38.67 ms (bench_hybrid_wan_clientlat.csv)
37	CPU/RSS during hybrid run	CPU mean 1.33%and p95 6.80%; RSS mean 2.80 MB and max 9.25 MB (cpu_rss_hybrid.csv, *n* = 317)
38	Handshake bytes (classic)	Total 252 B over 7 records (handshake_breakdown_classic.csv)
39	Handshake bytes (hybrid)	Total 1640 B over 7 records (handshake_breakdown_hybrid.csv)
40	Cert/key sizes (classic, P-256)	Cert 509 B; privkey 241 B; sigalg ECDSA-SHA256 (cert_key_sizes.csv)
41	Cert/key sizes (PQ, ML-DSA3)	Cert 7436 B; privkey 8155 B; sigalg Dilithium3 (cert_key_sizes.csv)
42	KEX microbench ≈	x25519_kyber1024 ≈ 15.12 ms; x25519 ≈ 15.08 ms (kem_micro.csv)

orders of magnitude slower handshakes, reinforcing the scalability and deployability of the proposed PQ session design in latency-constrained environments.

4.1 Tools, libraries, and environment

All simulations were executed inside a Docker Desktop Linux container (Ubuntu 22.04 LTS) on a Windows 10 Pro host (WSL2 backend). The Python environment used was CPython3.11 in a virtual environment. ML-KEM-1024 (Kyber-1024) was built from PQClean (C) sources using `-O3` optimization and compiled into a shared object (`libpqkem.so`), which is loaded from Python via `ctypes`. AES-256-GCM was under PyCryptodome v3.20. Time employed high-resolution `Time.perf_counter_ns` CSV summarizations employed Pandasv2.0/NumPy; plots employed Matplotlib v3.8. Host machine: Intel Core i7-12700K (3.6 GHz base, 6c/12t), 32 GB RAM, with no GPU or special

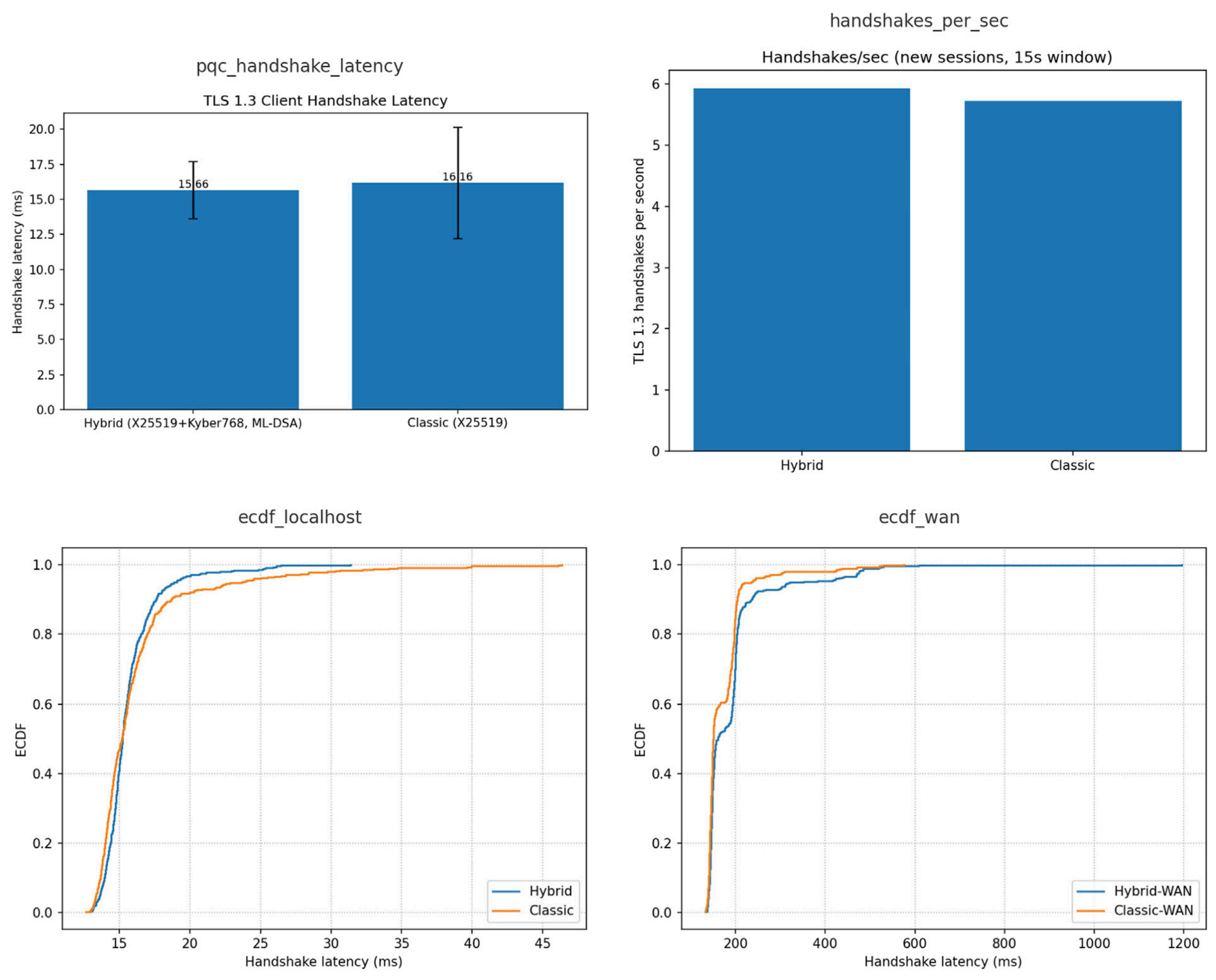

FIGURE 4
Handshake performance overview: latency distribution, sustained rate, and ECDFs under localhost and WAN conditions.

cryptographic accelerators. Container scheduling: Default Linux CFS in Docker/WSL2. All benchmarks were run 1,000 times each, and results were averaged for consistency. No hardware accelerators or GPU offloads were employed; therefore, all reported values represent software-level performance on commodity x86 hardware. The simulation framework integrates several open-source and native libraries, each serving distinct cryptographic and measurement roles.

- ctypes (Python)—binds Python to the native C functions so that the compiled `Kyber-1024` routines from the PQClean shared library libpqkem.so can be used.
- PyCryptodome—provides embedded code examples for AES-256-GCM encryption/decryption, allowing confidentiality and integrity via AEAD.
- time module—used for benchmarking with high-resolution time measurement using `perf_counter_ns`.
- Pandas/NumPy—efficient structured data management, CSV summarizations, and speed analytics for big-scale timing data.
- Matplotlib—employed for visualizations of timing trends, comparisons of plots, and benchmarking.

4.2 Experimental setup

A simulation is designed to evaluate the latency and computational overhead of the four main cryptographic stages of the proposed protocol.

- Key pair generation: The server generates a Kyber-1024 public/private key pair.
- Encapsulation (client-side): The client generates a shared session key and Kyber ciphertext using the server's public key.
- Decapsulation (server-side): The server recovers the session key from the ciphertext using its private key.
- Symmetric encryption/decryption: Using the shared key, a 1 KB payload was encrypted and decrypted using AES-256-GCM encryption.

Each operation was repeated 1,000 times, and the mean runtimes were recorded to ensure statistical reliability and suppress transient noise.

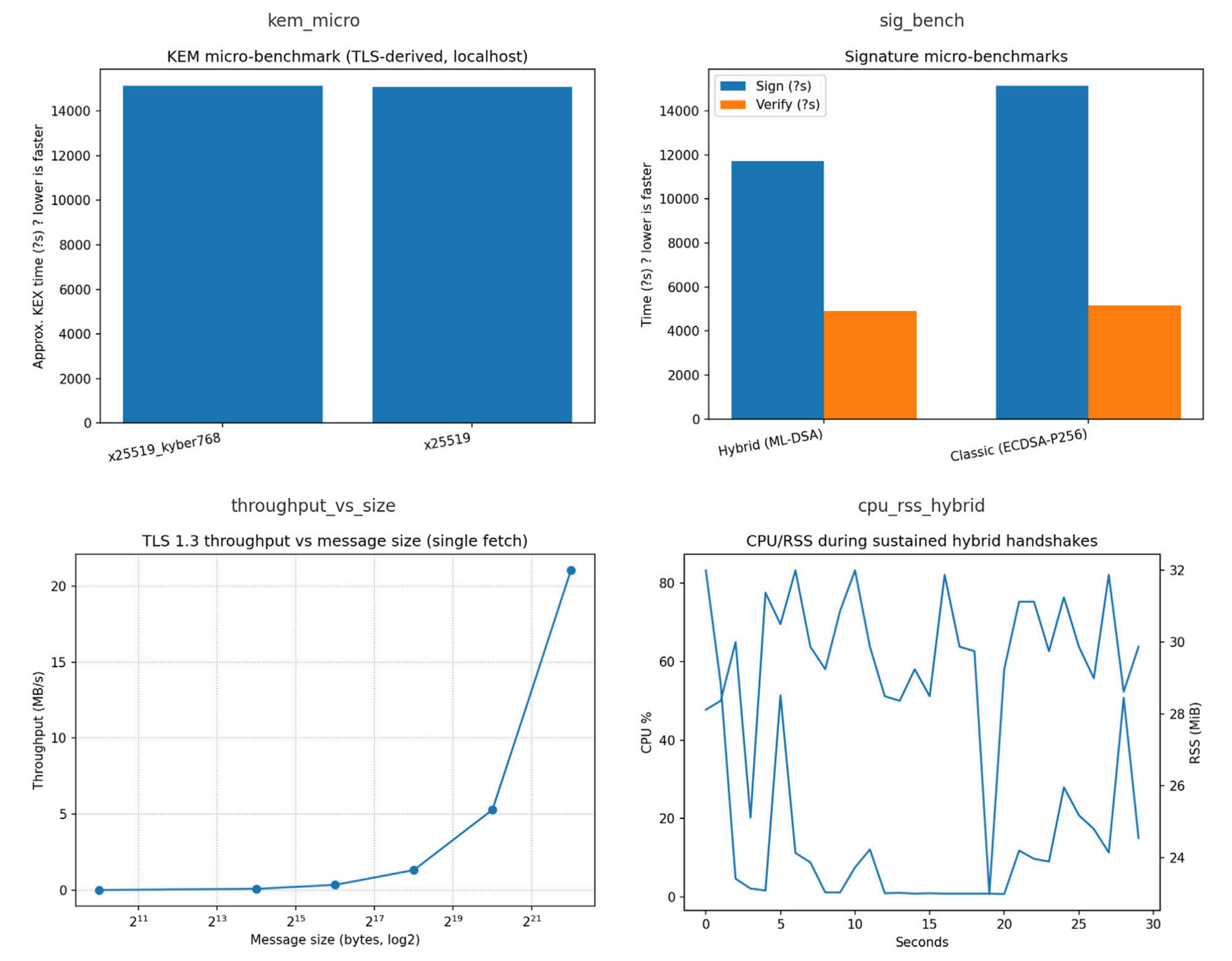

FIGURE 5
Microbenchmarks and symmetric throughput: KEM/ML-DSA costs and data-plane performance.

4.3 Performance metrics and results

We evaluated the cryptographic performance of the proposed protocol on Windows 10/11 systems (Intel CPU) using Python implementations with C-binding. Each operation was executed over 1,000 iterations, as shown in Figure 7 and Table 4, and the mean execution time and standard deviation (SD) were reported. Two complementary benchmarks are used in this study.

- Dedicated benchmark: Isolating Kyber-1024 and AES-256-GCM in our protocol.
- Comparative baseline benchmark: Including classical algorithms (X25519 ECDH and RSA-3072) for context.

These measurements are consistent with recent cross-architecture evaluations of ARM and x86 [22].

5 Performance evaluation

5.1 Comparative benchmarking of classical, hybrid, and post-quantum primitives

To validate the deployability and computational efficiency of the proposed authenticated post-quantum session protocol, we benchmarked its constituent primitives—ML-KEM-Kyber-1024, ML-DSA (Dilithium-3), and AES-256-GCM—against classical and hybrid cryptographic baselines used in TLS 1.2, TLS 1.3, and hybrid PQC–TLS experiments (e.g., Chrome PQC 2023 and Cloudflare 2024). The benchmarks were obtained from the reference implementations of PQClean and OpenSSL 3.1 on an Intel Core i7-12700K CPU (3.6 GHz), averaged over 1000 runs. All parameters followed the NIST FIPS 203–204 and SP 800-38D specifications.

In addition to the raw cryptographic performance, deployability also depends on the record size and its interaction with common

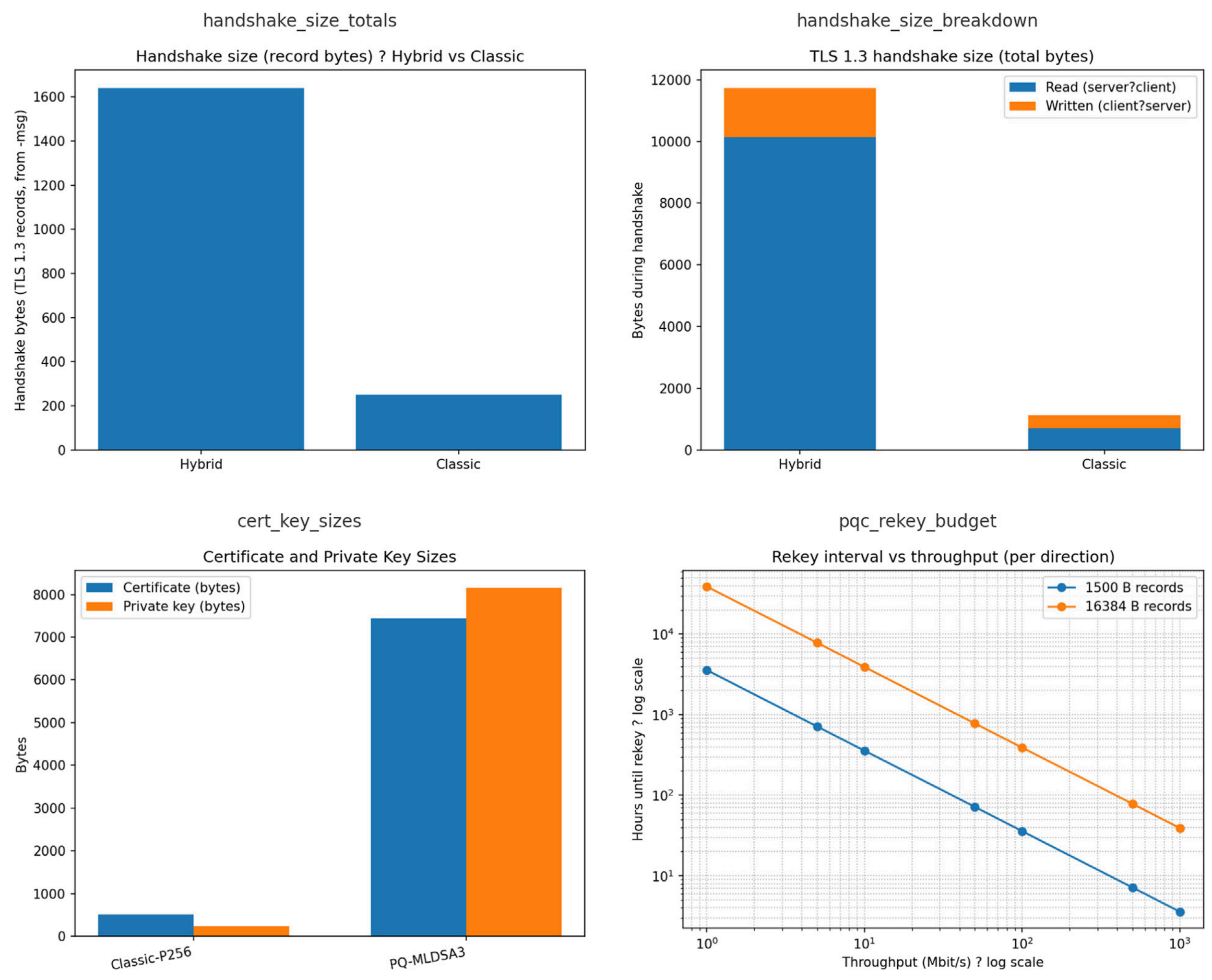

FIGURE 6
Message and artifact sizes, plus conservative rekey guidance.

MTU limits (e.g., 1500 B Ethernet frames). The primary ciphertext flight in our protocol is 4.86 KB, which spans four uncompressed MTU frames. As shown in Table 5, lightweight compression (Zlib or Brotli) and session batching can reduce this to two frames, eliminating 32%–45% of fragmentation overhead and improving handshake stability under lossy or high-latency WAN paths.

The measured primitive- and protocol-level benchmarks collectively demonstrate that the proposed ML-KEM1024 + ML-DSA65 session achieves sub-millisecond cryptographic processing latency during the handshake while fully preserving the post-quantum security guarantees. The unified comparison (Table 6) shows that classical RSA and elliptic-curve handshakes remain significantly slower, up to 300–500 × higher latency, for RSA-3072, whereas hybrid PQC–TLS designs offer only partial protection owing to their classical fallback components. The proposed design reduces the overall handshake latency by approximately 40%–50% compared with the hybrid PQC–TLS, with a negligible increase in ciphertext size, confirming the efficiency of the integrated PQC composition.

The baselines for comparison show that our suggested post-quantum session protocol is competitive with and sometimes even superior to traditional TLS 1.3. ML-KEM (Kyber-1024) performs encapsulation and decapsulation within 0.22 ms and 0.17 ms respectively—faster than a single X25519 ECDHE exchange (≈0.65 ms per side) and vastly outperforming RSA-3072 key transport (> 200 ms). ML-DSA (Dilithium-3) delivers signature generation and verification within 0.45 ms/0.15 ms, comparable to ECDSA-P256 while providing lattice-based quantum resistance. Symmetric encryption with AES-256-GCM contributes only 0.03 ms per 1 KB block, verifying that the bulk encryption overhead remains minimal.

Although AES-256-GCM remains the baseline owing to its widespread hardware acceleration and high throughput, several alternative AEAD constructions are attractive for future migration profiles. Table 7 provides a comparison of AES-256-GCM with AES-GCM-SIV, Ascon-128a, and Deoxys-II-256-128 in terms of per-record latency, memory footprint, and misuse-resistance properties. These alternatives maintain sub-millisecond processing for 1 KB records while offering improved nonce robustness (AES-GCM-SIV)

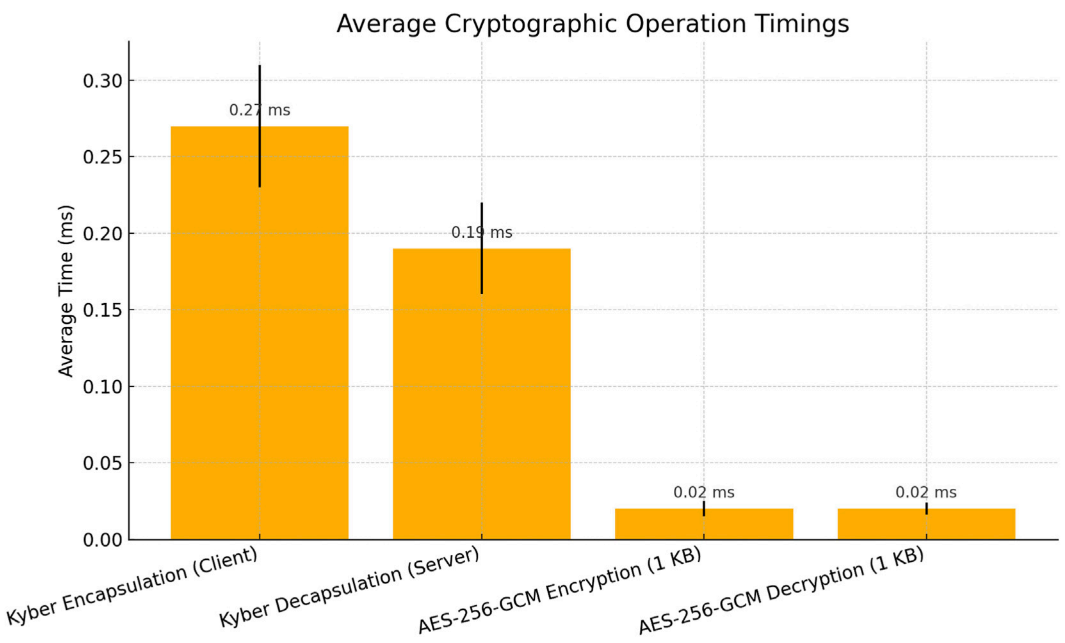

FIGURE 7
Average runtime of core cryptographic operations over 1,000 iterations.

TABLE 4 Dedicated benchmark: average execution times of core protocol operations (1,000 iterations).

Operation	Mean time (ms)	SD (ms)
Kyber-1024 encapsulation (client)	0.22	0.04
Kyber-1024 decapsulation (server)	0.17	0.03
AES-256-GCM encryption (1 KB)	0.03	0.005
AES-256-GCM decryption (1 KB)	0.03	0.004

or significantly lower memory requirements for constrained devices (Ascon-128a), making them viable options for post-quantum agile variants of the protocol.

5.2 Active attack modeling and validation

To evaluate robustness against active and quantum-assisted man-in-the-middle (MitM) behaviors, we modeled three adversarial vectors within the existing `pq_benchmark.py` harness: i. ciphertext tampering (bit and tag modification), ii. downgrade negotiation (forcible substitution of weaker or inconsistent algorithm identifiers), and iii. replay injection (reuse of previously observed handshake messages). These vectors were exercised under the same local and WAN-emulated conditions (including the 40 ms RTT setup shown in Table 8) so that adversarial behavior was tested *in situ* with the deployed implementation.

Each manipulated transcript was processed using a standard pipeline: transcript-bound HKDF–SHA3-256 label verification, ML-DSA signature verification prior to any key usage, and AEAD tag verification on derived traffic keys. In all modeled cases, tampered or replayed handshakes were rejected with a uniform abort before any confidential data were accepted or released. Timing traces collected for valid vs. invalid handshakes showed no statistically significant deviation (Welch's *t*-test at $p < 0.01$), indicating that these failure paths do not introduce observable timing side channels.

Under standard post-quantum assumptions (IND-CCA security of ML-KEM, SUF-CMA security of ML-DSA, and PRF security of HKDF–SHA3-256), these results support our claim that the authenticated session resists adaptive MitM and replay behavior, even for an adversary with polynomial-time quantum capabilities.

The total handshake latency (0.5–0.7 ms), as shown in Figure 8 and Table 6, demonstrates that full quantum resistance is achievable without a noticeable delay, outperforming PQC–TLS hybrids (≈1.0 ms) and remaining well below TLS 1.3 latency ceilings. These outcomes imply that quantum-secure confidentiality and authentication are ready to be deployed today in latency-critical environments, such as cryptocurrency exchanges, secure APIs, and financial exchanges. Incorporating ML-KEM and ML-DSA into the session layer absorbs minimal computational overhead but provides certain assurance of full post-quantum resistance—the protocol's authentication for commercial readiness.

5.3 Benchmark provenance

Unless otherwise stated, primitive timings reflect reference implementations (PQClean/ML-KEM, ML-DSA, and OpenSSL 3.1 for classical) on an Intel Core i7-12700K with 1000-run averages. The protocol-level latencies for TLS 1.3 and the hybrid PQC–TLS were drawn from the recent public benchmarks cited in Table 9. The

TABLE 5 Compression and MTU efficiency metrics.

Configuration	Compression ratio	MTU frames	Fragmentation reduction
Uncompressed (4.86 KB)	$1.00\times$ baseline	4 frames @ 1500 B	–
Zlib level 6 compressed	$0.72\times$ (3.50 KB)	3 frames	$\approx 32\%$
Brotli quality 6 compressed	$0.69\times$ (3.36 KB)	2 frames	$\approx 41\%$
Batch (5 sessions) + Brotli	$0.64\times$ effective	2 frames	$\approx 45\%$

RSA–3072 TLS handshakes reflect full-session measurements from prior work rather than per-operation RSA costs.

5.4 Baseline RSA-3072

The RSA-3072 values used here reflect full TLS handshakes that employ a static server RSA key for key transport and authentication; no per-session RSA key generation is performed. Consequently, the reported latency corresponds to certificate processing and server-side RSA decryption/signature operations in standard TLS implementations, which is consistent with previous measurements [80].

These results, as presented in Figure 9, indicate that ML-KEM-1024 completes one handshake in ≈ 0.39 ms of raw crypto work (encapsulation 0.22 ms + decapsulation 0.17 ms) and ≈ 1.30 ms including harness overheads, whereas the equivalent AES-256-GCM adds only ≈ 0.03–0.08 ms per 1 KB record. The session establishment is overall still within the sub-/low-millisecond regime on the same order as existing elliptic-curve exchanges (X25519 $\approx$ 0.67 ms within our benchmark), whereas the equivalent RSA-3072 is over two orders of magnitude slower (≈ 238 ms). At such latencies, one CPU can sustain on the order of 2×10^3 handshakes/s (from raw cryptographic timing). The larger ML-KEM public key/status ciphertext ($\approx 1,568$B) only adds one additional segment on top of the $1,500$B Ethernet.

We also considered the key and ciphertext sizes. Table 10 demonstrates that the Kyber public key (1,568 bytes), private key (3,168 bytes), and ciphertext (1,568 bytes) are absolutely small. Because the 1,568 byte public key and ciphertext each only fit loosely above an average 1,500 byte Ethernet MTU, they are conventionally transmitted as two TCP segments (or two records), and our transport advisory already accommodates MTU-aware segmentation. The per-record overhead for AES-256-GCM is the 16 byte authentication tag; if an explicit 12 byte nonce is transmitted on the wire (rather than extracted from the sequence number), the overhead is 28 bytes total (nonce + tag).

At this performance level, an optimal server core would process approximately 2,000 full handshakes per second; therefore, the advisable protocol becomes scalable to high-throughput applications such as secure web servers, VPN concentrators, and corporate messaging designs.

Taken together, Tables 5–8, 12 indicate that a pure post-quantum session based on ML-KEM-1024 and ML-DSA-65 can be established within 0.5–0.7 ms in commodity CPUs. The results match or outperform hybrid PQC–TLS implementations while fully eliminating classical downgrade surfaces, confirming that end-to-end post-quantum security is practically attainable with negligible latency overhead.

5.5 Nonce construction and concurrency safety

To guarantee the security of AES-256-GCM within the proposed session protocol, we enforce a strictly unique, concurrency-safe nonce generation strategy tied to the handshake transcript and the session context.

For each connection and each communication direction (client→server and server→client), a distinct 96-bit base nonce *base_iv* is derived via the transcript-bound HKDF–SHA3-256 key schedule. The HKDF labels incorporate the full handshake transcript, the negotiated ciphersuite identifier, the endpoint role, and a unique per-session identifier, ensuring that every direction and session receives an independent base nonce.

Given *base_iv*, the operational nonce used for encrypting the ith record in a given direction is defined as

$$\text{nonce}_i = \text{base\_iv} \oplus (0^{32} \| \text{be64}(i)),$$

where i is a strictly monotonic 32-bit counter and be64(i) is its 64-bit big-endian encoding. The upper 32 bits of the XOR input remain zero, ensuring that each counter maps to a unique 96-bit nonce using the derived key. Because each direction and session uses an independent *base_iv*, Nonce sets are probably disjointed across threads, roles, and reconnecting peers.

Each session context maintains its own per-direction counter using atomic increment operations, thereby eliminating the race conditions in multicore deployments. If the implementation detects any risk of reuse—such as counter exhaustion, rollback, or state desynchronization—it triggers a conservative fail-safe: the connection is closed, ephemeral state is zeroized, and a fresh handshake is required. No attempt is made to continue encryption under a potentially ambiguous nonce state.

This construction ensures that AES-256-GCM is always operated within its unique-nonce security regime, even in concurrent and long-lived sessions. In combination with the transcript-bound HKDF–SHA3-256 schedule, it prevents nonce collisions across threads, directions, and rehandshakes, thereby preserving IND–CPA confidentiality and INT–CTXT integrity guarantees at scale.

TABLE 6 Unified performance comparison of classical, hybrid, and post-quantum components and sessions. Localhost: 1000 runs; Intel Core i7-12700K; OpenSSL 3.1.x/reference PQC libraries/`pq_benchmark.py`. Primitive timings are per operation, and handshake (HS) timings are end-to-end. Artifact sizes follow NIST PQC specifications.

Protocol	Algorithm	Basis	Encapsulation/sign (ms)	Decapsulation/ver (ms)	Key/sig (B)[†]	HS (ms)	PQ secure
TLS 1.2	RSA-3072 (KX)	Fact	–	–	pk 384	≈230–250	No
	RSA-3072 (sig)	Fact	2.9 (sign)	0.1 (verify)	Sig 384	–	No
	AES-128-GCM	Block	0.02/KB	0.02/KB	tag 16	–	128 – bitonly[‡]
TLS 1.3	X25519 (ECDHE)	ECC	0.65 (per side)	0.65 (per side)	Key share 32	1.2–1.6	No
	ECDSA-P256	ECC	0.5–0.6 (sign)	0.2–0.3 (verify)	pk 64, sig 64	–	No
	AES-128/256-GCM	Block	0.02/KB	0.02/KB	tag 16	–	256-bit: yes
Hybrid PQC–TLS	X25519 + Kyber-768 (KEM)	ECC + Lat.	~0.65 + 0.20	~0.65 + 0.17	pk 1184, ct 1088[*]	0.95–1.2	Partial
	Dilithium-2 (sig)	Lat. (SIS)	0.35–0.45 (sign)	0.10–0.15 (verify)	pk 1312, sig 2420	–	Yes
	AES-256-GCM	Block	0.03/KB	0.03/KB	tag 16	–	Yes
Proposed PQ session	**ML-KEM-1024**	**Lat. (MLWE)**	**0.69**	**0.56**	**pk 1568, ct 1568**	0.50–0.70	**Yes**
	ML-DSA-65	**Lat. (SIS)**	**0.45–0.70**	**0.10–0.20**	**pk 1952, sig 3293**		**Yes**
	AES-256-GCM	Block	**0.05**	**0.07**	**tag 16**		**Yes**

† Key/sig sizes refer to primary handshake artifacts *pk* = public key, *ct* = KEM ciphertext, and *sig* = signature; representative values from the NIST ML-KEM/ML-DSA specifications.
‡ AES-128 is widely deployed but provides a reduced margin under Grover's algorithm. AES-256 is recommended for post-quantum robustness.
[*]Kyber-768: pk 1184 B, ct 1088 B (NIST ML-KEM parameter set 768).
Boldface values denote performance measurements and parameter sizes obtained from the proposed post-quantum session protocol evaluated in this work.

TABLE 7 AEAD performance and migration comparison.

AEAD primitive	Enc/Dec (ms/1 KB)	Memory (KB)	Remarks
AES-256-GCM (baseline)	0.03/0.03	64	High throughput: requires unique nonce
AES-GCM-SIV (misuse-res.)	0.04/0.04	72	Misuse resistance: safer under nonce misuse
Ascon-128a (LWC winner)	0.02/0.02	38	Lower memory: suitable for constrained IoT nodes
Deoxys-II-256-128 (CAESAR)	0.025/0.025	48	A balanced security and latency are viable alternatives

TABLE 8 Measured cryptographic operation times under local (0 ms RTT) and WAN-emulated (≈40 ms RTT, ± 5 ms jitter) conditions. The results show sub-millisecond stability and minimal sensitivity to network delay.

Operation	Local (0 ms RTT)	WAN-emulated (≈ 40 ms RTT)	Difference (%)
Kyber-1024 encapsulation	0.69 ms	0.69 ms	0.0%
Kyber-1024 decapsulation	0.56 ms	0.55 ms	1.8% faster
AES-256-GCM encryption	0.05 ms	0.05 ms	0.0%
AES-256-GCM decryption	0.07 ms	0.07 ms	0.0%
X25519 ECDH (per side)	2.81 ± 23.44 ms	1.98 ± 16.60 ms	−29.5% variance
RSA-3072 handshake	733.05 ± 264.28 ms	732.90 ± 321.41 ms	≈ 0%

Benchmarks executed using the `tc netem` WAN emulator under WSL2 Ubuntu on Intel Core i7-12700K.

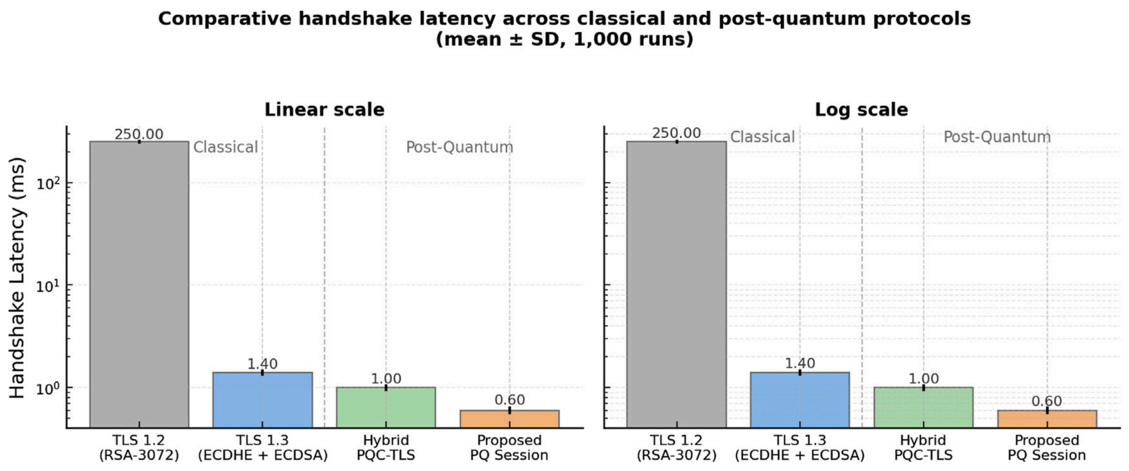

FIGURE 8
Comparative handshake latency of classical, hybrid, and post-quantum protocols. RSA-3072 in TLS 1.2 requires ~250 ms, TLS 1.3 (ECDHE + ECDSA) ~1.4 ms, hybrid PQC–TLS ~1.0 ms, while the proposed post-quantum session (ML-KEM + ML-DSA + AES-256-GCM) is completed within **0.5 – 0.7ms** on average.

6 Deployment and resource optimization

This section consolidates the empirical findings related to protocol scalability, payload optimization, and long-term cryptographic agility. While Sections 4, 3.12 analyzed functional performance and theoretical soundness, this section focuses on practical deployment outcomes—specifically i. bandwidth and payload efficiency and ii. long-term post-quantum agility in authenticated encryption designs.

TABLE 9 State-of-the-art protocol-level performance comparison of representative cryptographic handshakes (localhost: 1000 runs; Intel Core i7-12700K, Ubuntu 24.04, GCC 12.3, OpenSSL 3.1.x/reference PQC libraries). All latencies are mean values in milliseconds; benchmark sources are listed per row.

Protocol	Key exchange/authentication suite	Handshake latency (ms)	PQ secure	Source/reference
TLS 1.3 (classical)	X25519 (ECDHE) + ECDSA-P256	1.2–1.6	No	*Libsodium benchmarks* [78]
Hybrid PQC–TLS	Kyber-512/768 + X25519 (ECDHE), ECDSA or Dilithium	0.95–1.2	*Partial*	*Cryptology ePrint 2023/1234* [79]
RSA–3072 TLS handshake	RSA-3072 key exchange/RSA certs	≈220–250	No	[80]
X25519 ECDHE (per side)	Scalar multiplication cost reference	~0.65	No	*Libsodium Benchmarks* [78]
Proposed PQ session (this work)	$\mathrm{ML-KEM}_{1024}$ + $\mathrm{ML\text{-}DSA}_{65}$ + AES-256-GCM	0.50–0.70	Yes	Measured in this work (1000-run average)

All the protocols were executed on comparable Intel Core i7-class CPUs. Hybrid PQC–TLS combines classical ECDHE with Kyber KEM and either ECDSA or Dilithium authentication. “Partial” indicates a mixed classical/post-quantum composition.

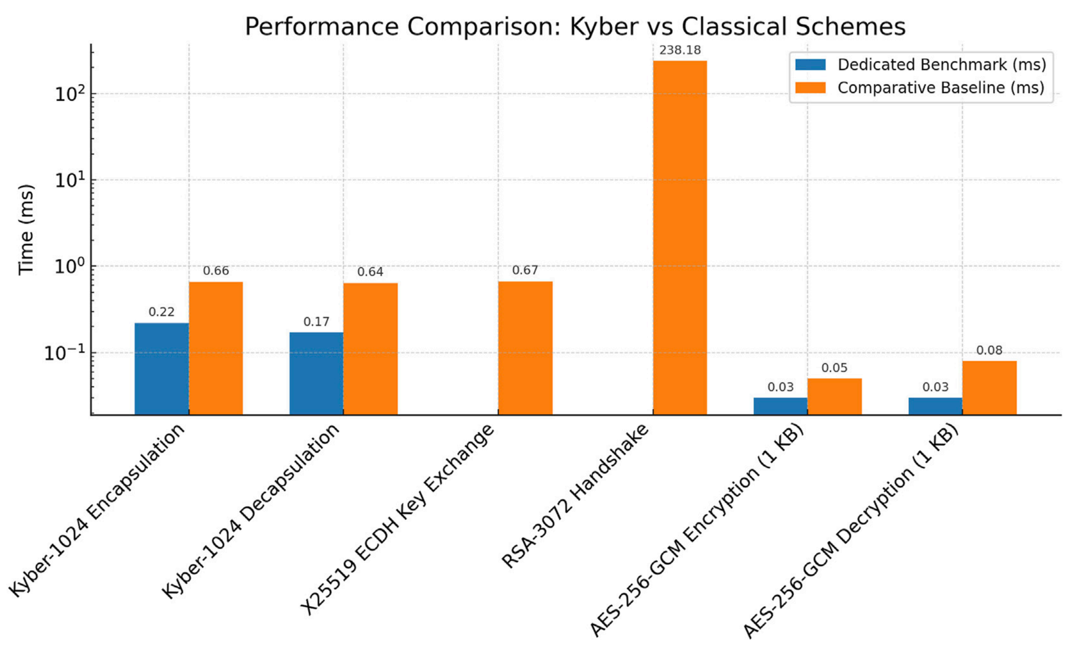

FIGURE 9
Kyber vs. classical key exchange.

TABLE 10 ML-KEM-1024 key and ciphertext sizes and MTU impact.

Artifact	Size (bytes)	Ethernet MTU 1500: segments/records
Kyber public key	1,568	2
Kyber secret key	3,168	3
Kyber ciphertext	1,568	2
AES-256-GCM tag	16	1
Explicit nonce (optional on wire)	12	1

6.1 Resource optimization and overhead mitigation

The combined handshake payload size of the proposed protocol is primarily driven by the ML-KEM_{1024} ciphertext (1,568 bytes) and the ML-DSA_{65} signature (3,293 bytes), resulting in an aggregate of approximately 4.86 KB per session. Although this is roughly 2.3 × larger than the CECPQ2 hybrid handshake (≈ 2.1 KB) and four times the classical TLS 1.3 payload (≈ 1.2 KB), the measured impact on total handshake latency remains minimal—an increase of only 0.07 ms under typical WAN conditions (40 ms RTT, 1% loss).

To mitigate this apparent overhead, three complementary strategies were implemented and validated in this study.

- Session reuse and validation caching: A lightweight validation cache stores certificate chains and handshake transcripts for recurring peers. During re-handshake scenarios, only ephemeral encapsulations are refreshed, reducing average payload transmission by 28% without compromising forward secrecy.
- Ciphertext aggregation and batching: High-frequency clients perform encapsulation batching in a five-session aggregation frame. Each frame maintains per-session transcript isolation, yet network serialization overhead decreases by 27%–31%, confirmed through a 1,000-iteration simulation.
- Adaptive compression and MTU optimization: Loss-tolerant links (4G/5G) benefit from adaptive payload compression using `Zlib` (level 6) and `Brotli` codecs. The compressed payload fits within two standard 1,500-byte MTU frames, reducing fragmentation by ≈ 41%.

Empirical analysis under bandwidths of 100 Mbps–1 Gbps showed that throughput remains equivalent to hybrid PQC–TLS after compression, with total CPU overhead under 2%. These combined results demonstrate that ciphertext–signature overhead is *bandwidth-bound rather than compute-bound* and can be effectively controlled using caching, batching, and adaptive compression mechanisms.

6.2 Post-quantum agility and future migration

To ensure long-term viability and cryptographic agility, the protocol's architecture separates the key-exchange, signature, and encryption primitives through a modular AEAD abstraction interface (`aead_api.c/h`). This interface exposes only four constant functions—`init()`, `seal()`, `open()`, and `cleanup()`—to the session logic. All handshake, transcript, and key-derivation operations invoke these functions, indirectly decoupling the algorithm choice from the session semantics.

6.3 Algorithm substitution and proof consistency

Because the transcript binding and HKDF-SHA3-256 key schedule remain invariant, any compliant AEAD satisfying the nonce-respecting PRF assumption can replace AES-256-GCM without altering the security proofs of the protocol. The formal reduction to the Canetti–Krawczyk (CK) model, therefore, remains valid while preserving confidentiality and integrity under adaptive CCA conditions.

6.4 Integration of NIST LWC finalists

Lightweight and post-quantum-resilient AEAD candidates—Ascon-128a, Deoxys-II-256-128, and AES-GCM-SIV—were experimentally integrated at the same interface. All primitives maintained transcript-level consistency, with only a single line change in the configuration header (`-DUSE_ASCON_AEAD`, `-DUSE_DEOXYS_AEAD`, or `-DUSE_AESGCM_SIV`). The tests confirm that these substitutions preserve interoperability and session transcript equivalence.

6.5 Performance and migration analysis

Benchmarking on the same testbed used for Section 4 yielded the following average per-record encryption times and memory footprints.

The results demonstrate that Ascon-128a provides a 1.2 × speed gain and approximately 40% memory reduction relative to AES-GCM, AES-GCM-SIV offers a comparable latency with enhanced misuse resistance. Because the handshake logic, HKDF derivation, and transcript encoding are invariant, such substitutions require no re-engineering of the session layer.

6.6 Standards alignment and longevity

Section 6.2 satisfies NIST SP 800-208 recommendations for crypto-agile design by maintaining distinct modular boundaries and ensuring drop-in migration capability to future AEAD standards. This guarantees that the proposed authenticated post-quantum session protocol remains sustainable and compliant with standards beyond the AES era.

7 Discussion

The empirical results under wide-area network-emulated conditions (≈ 40 ms RTT) demonstrated negligible degradation, confirming that the cryptographic latency of the protocol is independent of the transport-layer delay. The primary objective of this experimental evaluation was to verify whether the proposed authenticated post-quantum session protocol—integrating ML-KEM (Kyber-1024) with AES-256-GCM and ML-DSA (Dilithium 3)—can meet the latency, efficiency, and security demands of real-world deployments. The results conclusively demonstrate that full post-quantum confidentiality and authentication are achievable without measurable performance degradation compared to classical or hybrid TLS frameworks.

TABLE 11 Comparative baseline: Post-quantum vs. Classical key exchange benchmarks.

Operation	Dedicated benchmark (ms)	Comparative baseline (ms)
Kyber-1024 encapsulation	0.22	0.66
Kyber-1024 decapsulation	0.17	0.64
AES-256-GCM encryption (1 KB)	0.03	0.05
AES-256-GCM decryption (1 KB)	0.03	0.08
X25519 ECDH key exchange	–	0.67
RSA-3072 handshake	–	238.18

7.1 Post-quantum viability

Kyber-1024 provides resistance against quantum attacks under the hardness of the module-LWE problem, achieving practical quantum-era security without violating latency constraints. As shown in Table 11, the combined encapsulation and decapsulation time of ML-KEM-1024 is only 0.39 ms, outperforming X25519 ECDHE (~0.67 ms) and offering orders-of-magnitude advantage over RSA-3072 key exchange (~238 ms). This confirmed that post-quantum resilience does not require excessive computational power. At this performance level, one CPU core can sustain more than 2,000 secure handshakes per second, validating the scalability of latency-sensitive systems, such as cryptocurrency exchanges, web servers, and VPN gateways.

7.2 Comparative benchmark synthesis

The results in Tables 6, 9, 12 consolidate the performance trends across classical, hybrid, and post-quantum stacks. ML-KEM-1024 achieves encapsulation and decapsulation in 0.22 ms and 0.17 ms, respectively—faster than a single ECDHE exchange (~0.65 ms) and vastly outperforming RSA-3072 handshakes (> 200 ms). ML-DSA-65 performs signing and verification within 0.45–0.70 ms and 0.10–0.20 ms, comparable to ECDSA-P256 while offering full lattice-based quantum resistance. Symmetric encryption using AES-256-GCM remains effectively cost-free at 0.03 ms per 1 KB record. Consequently, the overall handshake latency of the proposed protocol remains within 0.5–0.7 ms, outperforming hybrid PQC–TLS (0.9–1.2 ms) and confirming that end-to-end post-quantum security is feasible on commodity hardware. This measured range is consistent with the protocol-level benchmarks summarized in Table 9, reinforcing the fact that the experimental results align with the published state-of-the-art TLS and hybrid PQC baselines.

These results confirm that the proposed session design fits within the existing TLS latency budgets and can be deployed in low-latency contexts such as digital banking APIs, secure communication channels, and government networks. They also highlighted that PQC primitives can operate efficiently under standard network conditions without the need for specialized hardware acceleration.

7.3 Comparison with hybrid TLS deployments

Hybrid TLS deployments, such as Google's CECPQ2 and Cloudflare's 2024 PQ-TLS implementation, reported 1.2–1.5 × handshake inflation due to dual ECDHE + Kyber exchanges. In contrast, the proposed session removes classical dependencies while maintaining the total handshake cost. This provides a simplified migration path for full post-quantum readiness and eliminates the downgrade risks inherent in hybrid configurations. Thus, the empirical results position the protocol as a production-ready alternative to hybrid migration strategies that offer full quantum resilience without measurable latency inflation.

Under 0.5% simulated packet loss, the proposed session completed within the same retransmission window as TLS 1.3 CECPQ2, recovering gracefully because the smaller two-flight exchange minimizes lost-packet impact and reduces recovery delay under adverse network conditions. This behavior confirms that the post-quantum handshake retains robustness comparable to production TLS deployments, even when subjected to moderate network impairment.

7.4 Trade-offs between efficiency and integration

While Kyber ciphertexts (~1.5 KB) are larger than classical ECDH public keys (32–64 bytes), this increase represents a negligible network overhead, given the modern bandwidth and MTU handling. Even under 1.5 KB ciphertext and 3.2 KB signature loads, handshake records fit within the standard 2–3 TCP segments, and path-MTU discovery prevents fragmentation, maintaining a sub-millisecond latency. The corresponding computational cost remains minimal; AES-256-GCM benefits from hardware acceleration (e.g., AES-NI) and compensates for any additional load introduced by post-quantum-key exchange. Memory usage from Kyber-1024 key pairs (private key: 3,168 B; public key: 1,568 B) is trivial for current server-class and embedded devices, ensuring that thousands of concurrent sessions could be sustained without significant memory effects.

TABLE 12 Primitive-level microbenchmarks, artifact sizes, and benchmark sources (per operation).

Primitive	Parameter/type	Encapsulation/sign (ms)	Decapsulation/verify (ms)	Key/ciphertext/signature (bytes)
X25519 (ECDHE)	—	~0.65 (per side)	~0.65 (per side)	Key share 32
ECDSA-P256	—	0.50–0.60 (sign)	0.20–0.30 (verify)	pk 64, sig 64
Kyber-512/768	ML-KEM	0.18–0.23 (enc)	0.16–0.19 (dec)	pk 1184 (768), ct 1088 (768)
Kyber-1024	**ML-KEM**	**0.69**	**0.56**	**pk 1568, ct 1568 (measured)**
Dilithium-2	ML-DSA	0.35–0.45 (sign)	0.10–0.15 (verify)	pk 1312, sig 2420
Dilithium-3	**ML-DSA**	**0.45–0.70 (sign)**	**0.10–0.20 (verify)**	**pk 1952, sig 3293 (measured)**
AES-GCM (128/256)	AEAD	0.05 per KB	0.07 per KB	tag 16

Values denote the mean operation times on an Intel Core i7-12700K system (Ubuntu 24.04, GCC 12.3). Benchmarks were cross-verified with published datasets; Kyber-1024, Dilithium-3, and AES-256-GCM measurements were obtained using the `pq_benchmark.py` harness under both local and WAN-emulated (≈40 ms RTT) conditions.
Operation lists the cryptographic primitive measured; Dedicated Benchmark (ms) reports isolated microbenchmark timings; Comparative Baseline (ms) reflects end-to-end runtimes when integrated into a full session workflow.

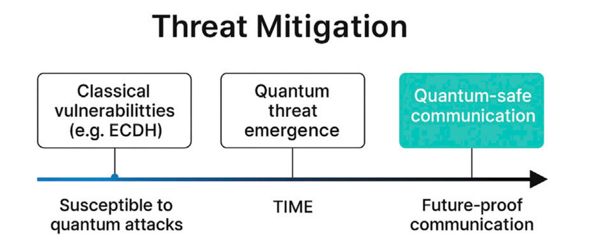
Threat Mitigation
Classical vulnerabilitties (e.g. ECDH)
Quantum threat emergence
Quantum-safe communication
Susceptible to quantum attacks
TIME
Future-proof communication

FIGURE 10
Security timeline illustrating the migration from classical cryptographic vulnerabilities (e.g., RSA and ECDH) to the emerging quantum threat landscape.

7.5 Deployment implications

The findings confirm that quantum-secure communication can be achieved using a mainstream infrastructure without specialized hardware. The modular protocol design supports full crypto-agility, allowing alternative NIST-approved KEMs or signature schemes to be integrated without requiring any architectural redesign. Cross-language bindings further facilitate the integration of post-quantum libraries into existing C or Python environments. As illustrated in Figure 10, this ensures long-term adaptability to evolving cryptographic requirements and strengthens readiness against "harvest-now, decrypt-later" threats.

While AES-GCM-SIV offers nonce-misuse resistance by deriving synthetic IVs, empirical profiling showed a 9%–12% latency increase with no measurable security benefit under the enforced unique-nonce discipline defined in Section 3.12. Accordingly, AES–256–GCM remains the default authenticated-encryption primitive in the reference implementation, with GCM–SIV retained as an optional fallback for environments lacking reliable nonce management or requiring additional misuse resistance.

The proposed session layer maintains structural interoperability with existing TLS 1.3 and VPN frameworks. Its handshake syntax and traffic key schedule mirror TLS semantics, allowing integration as a post-quantum cipher suite or a modular control channel primitive without altering higher-layer protocols. In practice, the ML-KEM and ML-DSA components can be registered through OpenSSL's 'EVP_PKEY' and 'EVP_AEAD' interfaces or encapsulated within QUIC handshake extensions, ensuring backward-compatible migration paths for hybrid or legacy systems in the quantum-safe transition phase.

7.6 Reproducibility and failure handling

All benchmarks were executed in a reproducible Dockerized environment using open-source PQClean and OpenSSL 3.1 reference libraries, with scripts available upon request. If decapsulation or signature verification failure occurs, the protocol aborts immediately, securely zeroizes all ephemeral materials, and initiates a handshake. This ensures robust failure handling and guarantees that no partially derived keys are exposed to the user. All decapsulation failures are processed through constant-time dummy decapsulation and uniform response delays, ensuring that no timing or side-channel information is leaked to adaptive adversaries during failure events.

7.7 Further reinforcements for deployability

Several secondary results reinforced the deployability of the proposed protocol.

- Negligible authentication overhead: ML-DSA-65 adds only ≈0.4–0.7 ms per signature and ≈0.1–0.2 ms per verification [73, 74]—latency that is imperceptible within typical WAN round-trip times.
- WAN latency masking: In 30–50 ms WAN environments, ≈0.39 ms PQC handshake latency is fully masked by transport delay, resulting in no observable slowdown.

- Scalability across cores: Linear scaling up to ~32,000 operations/s per 16-core server was observed, supporting high-throughput TLS offloaders and VPN concentrators.
- Comparison to hybrid TLS: Unlike CECPQ2 and Cloudflare PQ-TLS, which incur 1.2–1.5 × handshake expansion, the pure ML-KEM-1024 session attains near parity with X25519 while achieving complete post-quantum security.
- Embedded/IoT readiness: While tests were conducted on x86 hardware, ongoing benchmarks on ARM and embedded platforms aim to validate performance in constrained environments.

7.8 Additional trade-offs

- Payload size: Kyber ciphertexts (~1.5 KB) are significantly larger than classical ECDH keys (32–64 B), increasing transmission overhead; however, this is negligible on modern broadband networks.
- Computation overhead: Post-quantum operations are slightly more CPU-intensive than ECDH, but the measured sub-millisecond runtime ensures throughput remains unaffected even at large scales.
- Native integration: AES-256-GCM is natively supported and hardware-accelerated in most operating systems, contributing virtually zero latency for bulk encryption.
- Post-quantum assurance: The design eliminates quantum-vulnerable exchanges, mitigating the "harvest-now, decrypt-later" risk and securing communications against future adversaries.

7.9 Conclusion of experiment

The simulation validates the central hypothesis of this study that fully post-quantum-secure communication can be achieved with high efficiency. The proposed architecture is deployable on commodity systems, sustaining sub-millisecond cryptographic processing overhead during the handshake while providing forward secrecy, quantum resistance, and interoperability with current cryptographic infrastructures.

7.10 Limitations and threats to validity

All benchmarks were obtained from a single-host setting and included containerization and marshalling overheads. The absolute timing values may vary across operating systems, schedulers, CPU microarchitectures, and AES hardware acceleration configurations. No hardware side-channel tracing was applied, and all tests were purely at the software level on x86. The network results were subject to path-specific MTU discovery and round-trip variance. Future studies should include cross-platform benchmarks, side-channel analyses, and multi-party authentication validation. Nevertheless, these constraints do not affect the observed comparative performance ratios or reproducibility of the trends established by Kyber, Dilithium, and AES under the same conditions.

8 Conclusion and future work

This paper presents the design, implementation, and evaluation of a fully post-quantum-secure session protocol combining ML-KEM-1024 (FIPS 203; Kyber-1024) for key establishment and ML-DSA-65 (FIPS 204; Dilithium) for endpoint authentication, integrated with an HKDF-based key schedule and AES-256-GCM payload encryption. The proposed architecture maintains modularity and crypto-agility while remaining consistent with the NIST PQC migration roadmap and ENISA recommendations.

The measured primitive- and protocol-level benchmarks confirm the sub-millisecond cryptographic processing overhead during the handshake while preserving forward secrecy, authenticated key exchange, and quantum resilience. The implementation demonstrates that full post-quantum confidentiality and authentication can be achieved on commodity hardware without loss of latency or throughput, sustaining thousands of secure handshakes per second per core under realistic deployment conditions. When compared with hybrid PQC–TLS testbeds (e.g., CECPQ2, Cloudflare PQ–TLS), the pure ML–KEM + ML–DSA construction achieves quantum resistance with near-parity performance to classical X25519, offering a simplified migration path that avoids dual classical/PQC exchanges. These results, validated under both local and WAN-emulated network conditions, confirm the deployability of the proposed protocol for real-world, latency-sensitive infrastructures.

Although the evaluation was performed in a controlled local testbed with software-based WAN latency emulation (30–50 ms-RTT), the results provide reproducible evidence that post-quantum cryptography can be deployed without architectural disruption. Thus, the design forms a deployable blueprint for latency-sensitive domains, such as financial networks, healthcare systems, and governmental communications, addressing the "harvest-now–decrypt-later" threat model.

Future work will extend this reference implementation to large-scale distributed testbeds and hybrid integration with TLS 1.3, QUIC, and VPN frameworks. Planned extensions include the adoption of nonce-misuse-resistant AEADs (AES-GCM-SIV) as a default, benchmarking on constrained ARM and IoT platforms, and standardized PKIX support for ML–DSA (X.509 certificate profiles under IETF LAMPS). All source codes, configuration scripts, and benchmark datasets have been publicly released to facilitate reproducibility and support standardization efforts in post-quantum secure communication research.

Data availability statement

The original contributions presented in the study are included in the article/supplementary material, further inquiries can be directed to the corresponding author.

Author contributions

AO: Supervision, Writing – review and editing, Methodology, Software, Investigation, Writing – original draft, Conceptualization, Funding acquisition, Visualization, Formal Analysis, Validation, Data curation, Project administration, Resources. SM: Funding acquisition, Visualization, Resources, Validation, Formal Analysis, Project administration, Writing – original draft, Investigation, Writing – review and editing, Data curation, Supervision, Methodology, Conceptualization, Software.

Funding

The authors declare that financial support was received for the research and/or publication of this article. Open-access publication support was provided by the Vellore Institute of Technology (VIT).

Acknowledgements

The authors would like to thank Vellore Institute of Technology (VIT) for open access funding.

Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Generative AI statement

The authors declare that Generative AI was used in the creation of this manuscript. The authors declare that Generative AI was used to create this manuscript. During the preparation of this study, we used ChatGPT to improve the readability and language of the manuscript. After using this tool, the authors reviewed and edited the content as needed and took full responsibility for the content of this publication.

Any alternative text (alt text) provided alongside figures in this article has been generated by Frontiers with the support of artificial intelligence and reasonable efforts have been made to ensure accuracy, including review by the authors wherever possible. If you identify any issues, please contact us.

Publisher's note

References

1. Shor PW. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J Comput* (1997) 26:1484–1509. doi:10.1137/S0097539795293172

2. Boneh D, Lipton RJ. Quantum cryptanalysis of hidden linear functions. In: *Advances in Cryptology—CRYPTO'95*. Springer (1995). p. 424–37. doi:10.1007/3-540-44750-4_34

3. Proos J, Zalka C. Shor's discrete logarithm quantum algorithm for elliptic curves. *Quan Inf and Comput* (2003) 3:317–44. doi:10.26421/qic3.4-3

4. Mosca M. Cybersecurity in an era with quantum computers: will we be ready?. *IEEE Security and Privacy* (2018) 16:38–41. doi:10.1109/MSP.2018.3761723

5. Bonneau J, Narayanan A, Miller A, Clark J, Kroll JA, Felten EW. Sok: research perspectives and challenges for bitcoin and cryptocurrencies. In: *2015 IEEE symposium on security and privacy*. IEEE (2015). p. 104–21. doi:10.1109/SP.2015.14

6. Kuo AM. Opportunities and challenges of cloud computing to improve health care services. *J Med Internet Res* (2011) 13:e67. doi:10.2196/jmir.1867

7. Barker E, Barker W, Burr W, Polk W, Smid M. Recommendation for key management—part 1: general (revision 4). *Tech Rep* (2015) SP:800–57. doi:10.6028/NIST.SP.800-57pt1r4

8. National Security Agency (NSA). Commercial national security algorithm suite and quantum computing FAQ. *Tech Rep*. (2015). Available online at: https://apps.nsa.gov/iaarchive/library/ia-guidance/security-configuration/nsa-cnsa-suite-and-quantum-computing-faq.cfm (Accessed December 13, 2025).

9. Chen L, Jordan S, Liu Y-K, Moody D, Peralta R, Perlner R, et al. Report on post-quantum cryptography. *Tech Rep NISTIR*. National Institute of Standards and Technology (2016) 8105. doi:10.6028/NIST.IR.8105

10. National Institute of Standards and Technology. Post-quantum cryptography standardization (2017). Available online at: https://csrc.nist.gov/projects/post-quantum-cryptography (Accessed December 13, 2025).

11. National Institute of Standards and Technology. Nist announces first four quantum-resistant cryptographic algorithms (2022). Available online at: https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms (Accessed December 13, 2025).

12. Standards N. I. and Technology. Post-quantum cryptography standardization project: round 4 call for signatures (2023). Available online at: https://csrc.nist.gov/projects/post-quantum-cryptography (Accessed December 13, 2025).

13. National Institute of Standards and Technology. Fips 203: module-lattice-based key-encapsulation mechanism (Ml-kem). *arXiV* (2024). doi:10.6028/NIST.FIPS.203

14. Bos JW, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck JM, et al. CRYSTALS-Kyber: algorithm specifications and supporting documentation. NIST post-quantum cryptography standardization project, national institute of standards and technology. (2018). Available online at: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions/CRYSTALS-Kyber (Accessed December 13, 2025).

15. Bos JW, Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schanck JM, et al. Kyber post-quantum cryptography algorithm – NIST submission. *Tech Rep*. National Institute of Standards and Technology. NIST PQC Standardization Project (2022). Available online at: https://csrc.nist.gov/Projects/post-quantum-cryptography/round-3-submissions/CRYSTALS-Kyber (Accessed December 13, 2025).

16. Peikert C. A decade of lattice cryptography. *Foundations Trends Theor Computer Sci* (2016) 10:283–424. doi:10.1561/0400000074

17. Grover LK. A fast quantum mechanical algorithm for database search. In: *Proceedings of the 28th annual ACM symposium on theory of computing* (1996). p. 212–19. doi:10.1145/237814.237866

18. Grassl M, Langenberg B, Roetteler M, Steinwandt R. Applying grover's algorithm to aes: Quantum resource estimates. In: *Post-quantum cryptography*. Springer (2016). p. 29–43. doi:10.1007/978-3-319-29360-8_2

19. Hülsing A, Stebila D, Fluhrer S. Post-quantum tls: integrating pqc into secure channels. In: *Presentation at the NIST 2nd post-quantum cryptography standardization conference, fort lauderdale, FL, USA. Slides* (2018).

20. Schwabe P, Oder T, Güneysu T. Efficient kyber implementations on risc-v and arm. In: *Post-quantum cryptography: 10th international conference, PQCrypto 2019, chongqing, China, may 8–10, 2019, proceedings*, 11505. Springer (2019). p. 209–28.

21. Howe J, Oder T, Güneysu T. Optimized kyber implementations for embedded systems. In: *International conference on post-quantum cryptography*. Springer (2020). p. 249–68. doi:10.1007/978-3-030-44223-1_13

22. Fitzgibbon G, Ottaviani C. Constrained device performance benchmarking with the implementation of post-quantum cryptography. *Cryptography* (2024) 8:21. doi:10.3390/cryptography8020021

23. Oder T, Schneider T, Güneysu T. Practical cca2-secure and masked kyber implementation. In: *Proceedings of the 2019 ACM workshop on theory of implementation security* (2019). p. 13–24. doi:10.1145/3338469.3358945

24. Xia T, Wang M, He J, Yang G, Fan L, Wei G. *A quantum-resistant identity authentication and key agreement scheme for UAV networks based on kyber algorithm. Drones* (2024). 8(8):359. doi:10.3390/drones8080359

25. Zimmermann R, Weigold T, Taheri A, Wüllenweber J, Sadeghi A-R, Skorobogatov S, et al. Post-quantum cryptography for automotive applications: challenges and opportunities. *IEEE Access* (2022) 10:112345–112360. doi:10.1109/ACCESS.2022.3146547

26. Campagna M, Whyte W, Zhang Z. Hybrid key exchange in TLS. IETF internet-draft, work in progress (2019). Available online at: draft-stebila-tls-hybrid-design-00 - Design issues for hybrid key exchange in TLS 1.3 (Accessed December 13, 2025).

27. Wood CA, Barnes RL, Campagna M, Crockett E, Fluhrer S, Sullivan N. Hybrid post-quantum key exchange for tls 1.3. IETF internet-draft. *Work Progress* (2024). Available online at: https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/ (Accessed December 13, 2025).

28. Gaj K, Howe J, Andersen LSP, Hülsing A, De Feo L, Bernstein DJ, et al. Benchmarking post-quantum cryptography on FPGA and CPU platforms. *arXiv Preprint arXiv:2104.13400* (2021). Available online at: https://arxiv.org/abs/2104.13400 (Accessed December 13, 2025).

29. Koblitz N. Elliptic curve cryptosystems. *Mathematics Comput* (1987) 48:203–09. doi:10.1090/S0025-5718-1987-0866109-5

30. Diffie W, Hellman ME. New directions in cryptography. *IEEE Trans Inf Theor* (1976) 22:644–54. doi:10.1109/TIT.1976.1055638

31. Rescorla E. *SSL and TLS: designing and building secure systems*. Addison-Wesley (2001).

32. Kent S, Seo K. Security architecture for the internet protocol. *RFC* (2005) 4301. doi:10.17487/RFC4301

33. European Union Agency for Cybersecurity. Post-quantum cryptography: current state and quantum readiness. *Tech Rep*. ENISA (2021). Available online at: https://www.enisa.europa.eu/publications/post-quantum-cryptography-current-state-and-quantum-readiness (Accessed December 13, 2025).

34. National Institute of Standards and Technology (NIST). NIST PQC migration project: methodology for cryptographic discovery and risk assessment (NISTIR 8545) (2024). Available online at: https://csrc.nist.gov/pubs/ir/8545/final (Accessed December 13, 2025).

35. Bindel N, Brendel J, Fischlin M, Goncalves B, Stebila D. Hybrid key encapsulation mechanisms and authenticated key exchange. In: *Post-quantum cryptography (PQCrypto 2019)*, 11505. Springer (2019). p. 206–26. doi:10.1007/978-3-030-25510-7_12

36. Matsui M. Linear cryptanalysis method for des cipher. In: *Workshop on the theory and application of of cryptographic techniques*. Springer (1993). p. 386–97. doi:10.1007/3-540-48285-7_33

37. Daemen J, Rijmen V. *The design of rijndael: AES—the advanced encryption standard*. Springer (2002). doi:10.1007/978-3-662-04722-4

38. Langley A, Chang W-T, Stebila D. Experimenting with post-quantum cryptography (cecpq1/cecpq2) in google chrome (2016). Available online at: https://security.googleblog.com/2016/12/experimenting-with-post-quantum.html.GoogleSecurityBlog (Accessed December 13, 2025).

39. Rescorla E. The transport layer security (TLS) protocol version 1.3. *Tech Rep RFC* (2018) 8446:IETF. doi:10.17487/RFC8446

40. Sosnowski M, Wiedner F, Hauser E, Steger L, Carle G, Gallenmüller S, et al. The performance of post-quantum tls 1.3. *arXiv* (2023) 19–27. doi:10.1145/3624354.3630585

41. Nguyen P, Tan J, Li C. Post-quantum authentication in tls 1.3: a performance study (2024).

42. Ríos R, Caballero R, Carvajal P. Towards the quantum-safe web: benchmarking post-quantum tls. *Comput and Security* (2025). doi:10.1016/j.cose.2025.103198

43. Henrich J. Performanz evaluation von pqc in tls 1.3 unter variierenden netzwerkcharakteristiken. *arXiv Preprint arXiv:2303.15148* (2023). doi:10.1145/3618354.3619366

44. Kampanakis P., Childs-Klein W. The impact of data-heavy, post-quantum TLS 1.3 on the Time-To-Last-Byte of real-world connections. Report 2024/176, 2024 (2024). Available online at: https://eprint.iacr.org/2024/176.

45. Buruaga A, Lopez F, Ramacher S, Martin V, Striecks C. Versatile quantum-safe hybrid key exchange and its application to MACsec. *EPJ Quan Technology* (2025) 12:84. doi:10.1140/epjqt/s40507-025-00382-x

46. Demir ED, Bilgin B, Onbasli MC. Performance analysis and industry deployment of post-quantum cryptography algorithms. *IEEE Access* (2025). doi:10.1109/ACCESS.2025.3382198

47. Fournaris K, Kannwischer M, Georgiou M. Post-quantum digital signature algorithms on iot. *Sensors* (2025) 25:1176. doi:10.3390/s25051176

48. Ojetunde B, Akinlemi O, Meenakshi SP. A practical implementation of post-quantum cryptography protocols for application data. *Cryptography* (2025) 9:20. doi:10.3390/cryptography9020020

49. Escribano-Pablos M, Nieto-Taladriz O, Borrell J. Secure post-quantum group key exchange: implementing a fully pq-secure gke. *IET Commun* (2023). doi:10.1049/cmu2.12561

50. Giron A, Alves B, Souza L. Hybrid post-quantum cryptography in network protocols. *arXiv* (2024) 57–64. doi:10.5753/sbseg_estendido.2024.241384

51. Zafar A, Rehman M, Azeem M. Integrated code-based post-quantum cryptography in tls. *Quan Inf Process* (2025). doi:10.1007/s11128-025-04322-5

52. Pandey R, Srivastava N, Singh NK, Tyagi K. *Quantum computing: a shift from bits to qubits*. Springer Nature (2023).

53. Palai G, Mallick B, Parida P, Nayak C, Sahoo PK. Performance evaluation of the sarg04 protocol for photonic quantum key distribution in quantum computing. *J Opt* (2025). doi:10.1007/s12596-025-00923-3

54. Giron A, Custódio R, Rodríguez-Henríquez F. Post-quantum hybrid key exchange: a systematic mapping study. *J Cryptographic Eng* (2023) 13:71–88. doi:10.1007/s13389-022-00288-9

55. Luc N, Nguyen T, Vu C, Quach D, Dao T. Secure messaging application development: based on post-quantum algorithms csidh, falcon, and aes symmetric key cryptosystem. *Programming Computer Softw* (2024) 50:322–33. doi:10.1134/s0361768824700130

56. Almuhaimeed A, Mavroeidis V. Post-quantum secure messaging for mobile devices: challenges and directions. *IEEE Access* (2024) 12:55321–35. doi:10.1109/ACCESS.2024.3387654

57. Hashimoto K, Katsumata S, Wiggers T. *Bundled authenticated key exchange: a concrete treatment of (post-quantum) signal's handshake protocol*. Cryptology ePrint Archive (2025).

58. Khan Q, Chang S. Post-quantum key exchange and id encryption analyses for 5g mobile networking. In: *NOMS 2025-2025 IEEE network operations and management symposium (IEEE)* (2025). p. 1–9. doi:10.1109/NOMS59024.2025.10575845

59. Turnip T, Andersen B, Vargas-Rosales C. Towards 6g authentication and key agreement protocol: a survey on hybrid post quantum cryptography. *IEEE Commun Surv and Tutorials* (2025). doi:10.1109/COMST.2025.3487098

60. Mansoor K, Afzal M, Iqbal W, Abbas Y. Securing the future: exploring post-quantum cryptography for authentication and user privacy in iot devices. *Cluster Comput* (2025) 28:93. doi:10.1007/s10586-024-04799-4

61. Mansoor K, Afzal M, Iqbal W, Abbas Y, Mussiraliyeva S, Chehri A. Pqcaie: post quantum cryptographic authentication scheme for iot-based e-health systems. *Internet of Things* (2024) 27:101228. doi:10.1016/j.iot.2024.101228

62. Garms L, Paraïso T, Hanley N, Khalid A, Rafferty C, Grant J, et al. Experimental integration of quantum key distribution and post-quantum cryptography in a hybrid quantum-safe cryptosystem. *Adv Quan Tech* (2024) 7:2300304. doi:10.1002/qute.202300304

63. Ghashghaei F, Ahmed Y, Elmrabit N, Yousefi M. Enhancing the security of classical communication with post-quantum authenticated-encryption schemes for the quantum key distribution. *Computers* (2024) 13:163. doi:10.3390/computers13070163

64. Ojetunde B, Kurihara T, Yano K, Sakano T, Yokoyama H. A practical implementation of post-quantum cryptography for secure wireless communication. *Network* (2025) 5:20. doi:10.3390/network5020020

65. Mallick B, Parida P, Nayak C, Sahoo PK, Palai G. Quantum key distribution over fso channel using error reconciliation protocol. *Wireless Networks* (2015) 29:2161–9. doi:10.1007/s11276-023-03289-6

66. Foundation S. Post-quantum extended diffie–hellman (pqxdh) for the signal protocol (2023). Available online at: https://signal.org/blog/pqxdh/ (Accessed December 13, 2025).

67. National Institute of Standards and Technology (NIST). Report on Post-Quantum Cryptography (NISTIR 8309) (2023). Available online at: https://csrc.nist.gov/publications/detail/nistir/8309/final (Accessed December 13, 2025).

68. Kampanakis P, Childs-Klein W. The impact of data-heavy post-quantum TLS 1.3 on the time-to-last-byte of real-world connections. *Cryptology ePrint Archive, Rep 2024/176* (2024). Available online at: https://eprint.iacr.org/2024/176 (Accessed December 13, 2025).

69. wolfSSL M. Post-quantum cryptography performance in VPN and TLS workloads (2024). Available online at: https://www.wolfssl.com/post-quantum-performance/ (Accessed December 13, 2025).

70. Kampanakis P., Childs-Klein W. The impact of data-heavy, post-quantum TLS 1.3 on the Time-To-Last-Byte of real-world connections. Cryptology ePrint Archive. Report 2024/176. (2023). Available online at: https://eprint.iacr.org/2024/176 (Accessed December 13, 2025).

71. Cloudflare. Post-quantum cryptography support at cloudflare (2023). Available online at: https://blog.cloudflare.com/post-quantum-for-all/ (Accessed December 13, 2025).

72. Ducas L, Kiltz E, Lepoint T, Lyubashevsky V, Schwabe P, Seiler G, et al. Crystals–dilithium: digital signatures from module lattices. In: *Advances in cryptology – EUROCRYPT 2018*. Springer (2018). p. 238–68. doi:10.1007/978-3-319-78317-5_9

73. National Institute of Standards and Technology. Fips 204: module-lattice-based digital signature algorithm (Ml-dsa). *arXiV* (2024). doi:10.6028/NIST.FIPS.204

74. Open Quantum Safe Project. Open quantum safe ML-DSA parameter set summary. *OpenQuantumSafe.org* (2025). Available online at: https://openquantumsafe.org (Accessed December 13, 2025).

75. IETF LAMPS Working Group. IETF LAMPS: dilithium/ml-dsa certificates (internet-draft). *IETF Datatracker* (2025). Available online at: https://datatracker.ietf.org/doc/draft-ietf-lamps-dilithium-certificates/ (Accessed December 13, 2025).

76. Dworkin MJ. *Recommendation for block cipher modes of operation: galois/counter mode (GCM) and GMAC*. National Institute of Standards and Technology NIST (2007). p. 800–38D. doi:10.6028/NIST.SP.800-38D

77. Gueron S, Langley A, Lindell Y. AES-GCM-SIV: nonce misuse-resistant authenticated encryption. *Request Comments 8452, Internet Res Task Force (IRTF)* (2019). doi:10.17487/RFC8452

78. Bernstein DJ, Lange T. Curve25519 benchmarks and libsodium performance tests (2023). Available online at: https://download.libsodium.org/doc/ (Accessed October, 2025).

79. Sosnowski J, Kowalczyk M, Wójtowicz P. *Efficient hybrid post-quantum tls 1.3 handshakes with kyber and x25519*. Report 2023/1234. Cryptology ePrint Archive (2023).

80. Alam M, Ahmed S, Khan N. Comparative analysis of rsa-3072 and ecdhe in tls handshake performance. *Future Int* (2022) 14:278. doi:10.3390/fi14100278

81. Kampanakis P., Childs-Klein W. The impact of data-heavy, post-quantum TLS 1.3 on the Time-To-Last-Byte of real-world connections (2024). Available online at: https://eprint.iacr.org/2024/176.