Post-Quantum TLS Migration: A Systematization of Hybrid Handshakes, PSKs, KeyUpdate, and Certificate Strategies

Hridoy Roy

City University of New York, New York, USA

Post-quantum cryptography; TLS 1.3; ML-KEM; ML-DSA; SLH-DSA; hybrid key exchange; ECDHE; PSK; KeyUpdate; X.509; PKIX; cryptographic agility; harvest-now-decrypt-later; TLS migration

Abstract. Post-quantum migration of Transport Layer Security (TLS) is often described as replacing a classical key-exchange algorithm with a post-quantum alternative. This framing is incomplete: TLS distributes security across key establishment, authentication, resumption and pre-shared keys (PSKs), traffic-secret evolution, and the X.509/PKIX ecosystem. These functions have different security objectives, failure modes, lifecycle dependencies, and deployment constraints. This Systematization of Knowledge (SoK) develops a functional framework for analyzing post-quantum TLS migration as an architectural problem rather than a single algorithm transition.

We systematize classical, pure post-quantum, and hybrid key establishment; TLS 1.3 PSK modes and resumption; KeyUpdate; ML-KEM; ML-DSA and SLH-DSA; hybrid and composite authentication; X.509/PKIX and trust-anchor migration; HSM dependencies; interoperability; performance; deployment environments; cryptographic inventory; and crypto-agility. We classify evidence by mechanism, security objective, source of keying material, forward-secrecy behavior, quantum threat, post-compromise behavior, communication and computational cost, interoperability dependency, standardization status, deployment evidence, and migration complexity. We explicitly separate finalized standards from evolving Internet-Drafts and implementation-specific evidence.

The systematization produces four analytical conclusions. First, confidentiality migration and authentication migration are coupled operationally but distinct security programs. Second, hybrid ECDHE-ML-KEM is a strong transitional architecture for confidentiality migration when the relevant TLS integration and implementation support are available, because its security objective can tolerate failure of one component; the exact security claim nevertheless depends on the standardized construction and its assumptions. Third, PSKs, resumption, and KeyUpdate are not interchangeable forms of rekeying: their security depends on the provenance and role of the secret, while KeyUpdate remains within an existing traffic-secret lineage and does not create an independent post-quantum secret. Fourth, deployment readiness is constrained as much by certificates, trust stores, HSMs, middleboxes, inventory, and interoperability as by primitive availability. We therefore derive a migration decision framework that prioritizes HNDL-sensitive data, controlled hybrid deployment, interoperability testing, staged authentication and PKI migration, and observable retirement of classical-only mechanisms.

E-mail: hridoy.roy08@cuny.edu (Hridoy Roy)

1 Introduction

Transport Layer Security is a principal mechanism through which Internet applications obtain authenticated, confidential, and integrity-protected communication. TLS separates the establishment of security parameters from the subsequent protection of application records: the handshake negotiates parameters, authenticates communicating parties, and establishes shared keying material, while the record layer uses derived traffic keys to protect application data [Res26]. Quantum computing challenges this architecture because widely deployed public- key mechanisms rely on mathematical problems that are vulnerable to sufficiently capable quantum algorithms. The migration problem is relevant before such a computer exists at cryptographically relevant scale. An adversary can record encrypted traffic today and attempt to decrypt it later after obtaining the computational capability required to attack the public-key mechanism used during the original handshake. This harvest-now- decrypt-later (HNDL) threat makes migration dependent on the confidentiality lifetime of information rather than solely on the date at which quantum computing becomes capable of breaking deployed cryptography [Nat26]. The natural response might appear to be replacing classical key exchange with a post-quantum algorithm. That view is incomplete because TLS uses asymmetric cryptography for multiple logically distinct functions. Key establishment creates shared secret material for the session. Authentication establishes the identity of a peer through signatures and certificates or through another authentication mechanism. Session resumption derives new connections from earlier security state. KeyUpdate evolves traffic secrets after the handshake. X.509 and PKIX determine how authentication keys are encoded, certified, distributed, and validated. A post-quantum mechanism addressing one of these functions does not automatically secure the others. This separation is particularly important for TLS 1.3 PSKs and KeyUpdate. PSKs can authenticate endpoints and, depending on the selected mode, participate directly in key establishment. Their security consequently depends on how they were generated, provisioned, protected, and related to previous sessions. A resumption PSK derived from a classical-only connection does not retroactively make that connection quantum- resistant. Similarly, KeyUpdate derives new traffic secrets from an existing traffic-secret lineage. It does not perform a new ECDHE or KEM exchange and therefore does not transform a quantum-vulnerable initial handshake into a post-quantum one [Res26]. The post-quantum TLS literature has developed rapidly. Alnahawi et al. provided a broad survey of post-quantum TLS in 2024, including classification and performance evaluation [AMOW24]. Since then, NIST finalized FIPS 203, FIPS 204, and FIPS 205, standardizing `ML-KEM`, `ML-DSA`, and `SLH-DSA` [Nat24b, Nat24a, Nat24c]. IETF work has progressed from the hybrid design draft to RFC 9954 for the general hybrid construction [SFG26], while standalone `ML-KEM` key agreement and `ML-DSA` authentication remain covered by evolving IETF documents [Con26, HSW26]; ECDHE-`ML-KEM` hybrid key agreement is now specified by RFC 10024 [KKWS26]. These developments create a need for a systematization that focuses not only on candidate algorithms but also on how the individual TLS functions interact during migration. This paper makes six contributions: 1. a functional taxonomy separating key establishment, authentication, resumption and PSKs, traffic-key evolution, PKI, deployment, and cryptographic agility; 2. a comparison of classical, hybrid, and pure-PQ key establishment by security as-sumptions, forward secrecy, interoperability, maturity, and communication cost;

3. a focused analysis of PSKs and KeyUpdate that distinguishes authentication, key establishment, and traffic-key evolution; 4. a separate treatment of PQ authentication and X.509/PKIX migration, including signatures, certificate chains, trust anchors, HSMs, and client compatibility; 5. a deployment-oriented synthesis of performance and interoperability evidence; and 6. a migration framework prioritizing inventory, HNDL-sensitive information, hybrid key establishment, interoperability, staged authentication migration, and crypto-agility. The paper does not propose a new cryptographic protocol, primitive, combiner, or

formally validated migration algorithm; its contribution is an analytical systematization of existing standards, research, implementation evidence, and migration guidance.

2 Background and Threat Model

2.1 TLS 1.3 and the Migration Boundary

TLS 1.3 establishes authenticated cryptographic state and then protects application traffic using symmetric traffic keys [Res26]. For post-quantum migration, an important architectural property is that symmetric record protection is conceptually separated from asymmetric key establishment and authentication. Consequently, a deployment can change the mechanism used to establish session secrets without necessarily replacing the symmetric AEAD used to protect application records. The classical TLS 1.3 model commonly uses ephemeral Diffie-Hellman mechanisms such as `X25519`. Ephemeral exchange

provides classical forward secrecy: compromise of a long-term authentication key does not by itself reveal previously established session keys. Quantum computation nevertheless threatens the underlying discrete-logarithm problem, so classical forward secrecy does not imply quantum resistance. A KEM provides a different interface. One party generates a public/secret key pair; another party encapsulates a shared secret to the public key; and the recipient decap- sulates the ciphertext to recover the secret. NIST FIPS 203 standardizes `ML-KEM-512`, ML- KEM-768, and `ML-KEM-1024` [Nat24b]. `ML-KEM` is therefore directly relevant to TLS key establishment, whereas `ML-DSA` and `SLH-DSA` address signature-based authentication and infrastructure [Nat24a, Nat24c].

2.2 Authentication and Key Establishment

TLS authentication and key establishment should be treated as separate security dimensions. A deployment may combine classical authentication with classical key establishment, classical authentication with hybrid key establishment, post- quantum authentication with hybrid key establishment, post-quantum authentication with pure post-quantum key establishment, or PSK authentication with either classical, hybrid, or post-quantum key establishment. Replacing ECDHE with `ML-KEM` can protect the confidentiality of newly established sessions against the quantum failure of classical key exchange, but it does not make a classical certificate signature quantum-resistant. Conversely, replacing certificate signatures does not protect historical ciphertext whose session keys were established using a quantum- vulnerable mechanism. This distinction is the foundation of the systematization: "PQC TLS" should not be treated as one security property. The security of a deployment is determined by the combination of mechanisms used for key establishment, authentication, resumption, traffic-key evolution, and PKI.

2.3 PSKs

TLS 1.3 supports PSKs obtained from session resumption and PSKs provisioned externally. RFC 9258 specifies mechanisms for importing external PSKs while binding them to a protocol, KDF, hash function, and optional context [BW22]. Two TLS 1.3 PSK key-exchange modes are particularly relevant. In `psk_ke`, the PSK is used without an additional ephemeral key exchange. In `psk_dhe_ke`, the PSK is combined with an ephemeral Diffie-Hellman exchange [Res26]. The latter demonstrates why PSK authentication and key establishment must not be conflated: a PSK can authenticate an endpoint while fresh ephemeral key material supplies forward-secret session entropy. For post-quantum migration, analogous configurations include PSK-only establishment, PSK plus classical ECDHE, PSK plus a PQ KEM, and PSK plus a hybrid ECDHE/PQ KEM. These configurations have different security properties and should be classified separately.

2.4 KeyUpdate

TLS 1.3 KeyUpdate changes application traffic keys after the handshake. The new application traffic secret is derived from the previous traffic secret using the TLS key schedule and the "traffic upd" label [Res26]. KeyUpdate is therefore a symmetric key- evolution mechanism rather than a new asymmetric exchange. This distinction has direct consequences for post-quantum security. The resulting secrets remain downstream of S0 . KeyUpdate does not introduce an independent KEM secret. If a future quantum attacker can recover the original quantum- vulnerable handshake secret and obtain the required transcript information, KeyUpdate does not provide the independent post-quantum foundation required to repair that hand- shake. KeyUpdate can nevertheless provide useful post-compromise properties. If an old traffic secret is deleted before an endpoint is later compromised, the attacker may be unable to reconstruct that earlier generation from a later traffic secret, assuming the one-wayness of the KDF and correct secret deletion [Res26]. This is key evolution and limited post-compromise protection, not quantum-resistant re-establishment.

Suppose the initial traffic secret depends on a classical secret Z_C:

$$S_0 = f(Z_C). \tag{1}$$

Repeated KeyUpdate operations then produce:

$$S_{i+1} = g(S_i). \tag{2}$$

2.5 Threat Model

The systematization considers four primary threat classes. Passive HNDL attackers record encrypted handshakes and application traffic and retain them for future cryptanalysis. If a captured session uses quantum-vulnerable ECDHE, future quantum computation may enable recovery of the session secret from the public handshake transcript. Future quantum attackers are modeled as adversaries capable of attacking classical public-key assumptions and potentially attempting attacks against post-quantum as- sumptions. PQC should not be interpreted as "quantum-proof"; rather, the relevant constructions are designed around assumptions believed to resist currently known quantum algorithms. Active attackers can modify handshake messages, attempt downgrade attacks, exploit implementation differences, or induce fallback to weaker mechanisms. Migration introduces additional negotiation states, making downgrade policy and transcript binding important. Endpoint compromise is distinct from cryptographic compromise. A fully compromised endpoint can expose current traffic secrets regardless of the algorithm used. KeyUpdate can reduce exposure under certain temporary-compromise models, but it cannot protect an endpoint under persistent active control. The threat model therefore separates two migration tracks: Confidentiality migration: protect new session-key establishment against future quantum attacks. Authentication migration: replace or augment quantum-vulnerable signatures and PKI infrastructure.

3 SoK Methodology

3.1 Research Questions

The systematization addresses eight research questions: 1. What are the principal quantum threats to TLS confidentiality and authentication? 2. How have standards bodies and researchers proposed migrating TLS key establishment to PQC? 3. How do hybrid TLS handshakes compare with classical TLS and pure-PQ approaches? 4. What roles do TLS 1.3 PSKs and KeyUpdate play in post-quantum migration? 5. What challenges arise

in post-quantum authentication and certificate migration? 6. What performance and interoperability costs are reported? 7. Which migration strategies are appropriate for different deployment environments?

8. What research and deployment gaps remain?

3.2 Source Selection

The literature search was structured around authoritative standards and primary technical literature. Sources considered include NIST standards and guidance, IETF RFCs, active IETF Internet-Drafts, IACR and cryptographic research, peer-reviewed security and networking research, official PKI specifications, implementation documentation containing deployment evidence, and government cybersecurity guidance. Search terms included combinations of "post-quantum TLS," "TLS 1.3 `ML-KEM`," "hybrid TLS," "`X25519 ML-KEM`," "ECDHE MLKEM," "KEM TLS," "TLS PSK post quantum," "TLS KeyUpdate quantum," "`ML-DSA` TLS," "`SLH-DSA` TLS," "post-quantum certificates," "X.509 post quantum," "PKIX `ML-DSA`," "TLS performance PQC," "TLS interoperability PQC," "harvest now decrypt later," and "crypto agility post quantum." The search emphasizes material available through August 15, 2026. Because IETF Internet-Drafts are works in progress, their status is treated separately from finalized RFCs. No numerical corpus size is claimed because the underlying manuscript does not provide a verified statistically exhaustive corpus.

3.3 Inclusion and Exclusion

A source was included when it materially addressed TLS 1.3, PQC key establishment, hybrid key exchange, PQ authentication, TLS PSKs, KeyUpdate, X.509/PKIX migration, TLS performance, interoperability, cryptographic migration, or deployment implications. Primary standards and primary technical research were prioritized. The principal evidence base excludes promotional material lacking technical evidence, generic quantum- computing literature without TLS relevance, duplicate versions where a final publication was available, unsupported technical claims, obsolete drafts when materially newer versions existed, and benchmark results whose experimental conditions could not be established.

3.4 Classification Dimensions

Each approach was classified according to key-establishment mechanism, authentication mechanism, hybridization, source of entropy, forward secrecy, quantum threat addressed, bandwidth and computational impact, interoperability dependencies, standardization status, deployment evidence, and migration complexity. The evidence was interpreted qualitatively rather than converted into unexplained numerical scores.

3.5 Methodological Limitations

The field changes faster than conventional publication cycles, and Internet-Drafts may change substantially before standardization. Performance studies also use different CPUs, libraries, operating systems, network conditions, protocol configurations, and security parameters. Direct numerical comparisons are therefore often inappropriate. There is also publication bias toward successful implementations and research prototypes. Failed interoperability tests and unsuccessful migrations are less visible in the literature. Accordingly, implementation availability is not treated as proof of universal deployment maturity.

Table 1: Evidence Extraction Schema

Field	Question
Mechanism	What cryptographic or protocol mechanism is being evaluated?
Security objective	Confidentiality, authentication, key evolution, recovery, or another property?
Key establishment	What creates the session secret?
Authentication	How is peer identity established?
Entropy source	PSK, ECDHE, KEM, or a combination?
Independence	Is new secret material independent of earlier traffic-key state?
Forward secrecy	What compromise scenario is covered?
Quantum threat	HNDL, future quantum attack, or neither?
Post-compromise behavior	What happens after temporary or persistent endpoint compromise?
Communication cost	Key, ciphertext, certificate, and handshake expansion?
Computational cost	Primitive, TLS, or end-to-end cost?
Interoperability	Which peers, stacks, certificates, and network elements must agree?
Standardization	Final standard, RFC, draft, or proposal?
Deployment evidence	Prototype, interoperability test, or production deployment?
Migration complexity	What operational dependencies must change?
Assumptions	Which cryptographic and deployment assumptions are required?
Limitations	What evidence or scope constraints remain?

3.6 Evidence Type and Status

The evidence extraction process records not only what a source says but what kind of authority the source represents. Each extracted item is assigned an evidence type: finalized NIST standard, Standards Track RFC, Informational RFC, active Internet-Draft, peer-reviewed

research, experimental implementation report, deployment report, or research proposal. A separate status field records whether the mechanism is standardized at the algorithm layer, standardized at the TLS integration layer, standardized at the certificate/PKIX layer, implemented, experimentally interoperable, or deployed. This distinction prevents a common category error: treating a finalized primitive as though its complete protocol and deployment ecosystem were also finalized. For example, `ML-KEM` is a finalized NIST standard, while particular TLS integrations remain separate protocol documents. Likewise, `ML-DSA` is standardized by NIST and has standardized X.509 algorithm identifiers, while TLS signature negotiation and broad verifier deployment involve additional specifications and ecosystem support.

3.7 Reproducibility Protocol

The systematization is reproducible at the level of source categories, search terminology, classification dimensions, and evidence interpretation. A future update can repeat the search using the same topic clusters, retrieve newer standards and drafts, and reclassify individual mechanisms without changing the analytical framework. The extraction schema used in this revision is summarized in Table 1. No numerical “maturity score” is assigned because the underlying evidence does not support a defensible universal scalar score. The framework instead preserves the dimensions that determine why one mechanism may be mature in one environment and immature in another.

4 Comparison with Existing Surveys and Systematizations

4.1 Why Functional Decomposition Changes the Analysis

The functional decomposition exposes two recurrent category errors. The first is to equate "post-quantum key exchange" with "post-quantum TLS." Key establishment is only one input to the TLS security state. The second is to treat every subsequent key change as an independent security reset. KeyUpdate does not have that property, and resumption can intentionally preserve a relationship to earlier connection state. The framework also changes how migration sequencing is evaluated. If a deployment has high HNDL exposure but a highly heterogeneous client population, the immediate priority is not necessarily PQ authentication. Conversely, a managed enterprise with centrally controlled trust stores may be able to experiment with PQ authentication earlier than a public-web service. The relevant decision variables are therefore not just algorithm availability but data lifetime, endpoint control, verifier availability, protocol support, and operational cost.

4.2 Evidence Hierarchy

Evidence is interpreted using an explicit hierarchy: 1. Finalized standards establish normative protocol or algorithm definitions. 2. Informational RFCs provide IETF-reviewed constructions or guidance but do not have Standards Track status. 3. Active Internet-Drafts describe evolving work and must be treated as work in progress. 4. Peer-reviewed experiments provide empirical evidence under stated conditions. 5. Implementation documentation establishes technical availability but not universal interoperability. 6. Deployment reports and operational guidance provide context about adoption and constraints. 7. Research proposals and preprints can identify emerging mechanisms but require additional caution when used to support deployment claims. This hierarchy is especially important in a field where the algorithm standardization cycle and the protocol-integration cycle are not synchronized.

5 Taxonomy of Post-Quantum TLS Migration

The literature can be organized into nine migration strategies; Table 2 summarizes the resulting strategy taxonomy.

5.1 Classical TLS

Classical TLS uses mechanisms such as `X25519` or other ECDHE groups for key establishment and classical signature mechanisms for authentication. Its advantages are mature interoperability, extensive implementation support, and predictable performance. Its limitation is quantum vulnerability of the public-key components. Classical TLS therefore represents the baseline from which migration proceeds rather than a long-term quantum-resistant architecture.

5.2 Pure Post-Quantum Key Establishment

Pure PQ key establishment replaces the classical public-key exchange with a PQ mechanism such as `ML-KEM`. Its primary advantage is a direct quantum-resistant key-establishment architecture. Its principal migration risk is reliance on the PQ assumption without the redundancy of a classical component and the need for all relevant endpoints to support the mechanism. The current IETF `ML-KEM` TLS Internet-Draft defines `ML-KEM-512`,

Table 2: Taxonomy of Post-Quantum TLS Migration Strategies

Strategy	Key establishment	Authentication	Main advantage	Main limitation	Migration maturity
Classical TLS	ECDHE	Classical signatures	Mature interoperability	Quantum-vulnerable public-key mechanisms	Mature
Pure PQ TLS	ML-KEM	Classical or PQ	Direct PQ key establishment	Single PQ assumption and evolving integration	Emerging
Hybrid TLS	ECDHE + ML-KEM	Classical or PQ	Transitional robustness	Larger messages and implementation complexity	Emerging/high
PSK-only	PSK	PSK	Efficient controlled deployment	Depends strongly on PSK lifecycle; no inherent fresh ephemeral exchange	Mature; context-dependent
PSK + PQ KEM	PSK + ML-KEM	PSK	Controlled authentication + PQ establishment	Provisioning and protocol integration	Emerging
PSK + hybrid KEM	PSK + ECDHE/ML-KEM	PSK	Controlled authentication + hybrid confidentiality	Provisioning/KEM complexity	Emerging
Hybrid authentication	Existing + PQ signature	Hybrid	Transition resilience	PKI and verifier complexity	Emerging
Pure PQ authentication	PQ signatures	ML-DSA/SLH-DSA	PQ identity	Client/CA/HSM/trust-store compatibility	Emerging
Crypto-agile	Configurable	Configurable	Enables staged replacement	Operational complexity	Architectural practice

`ML-KEM-768`, and ML- KEM-1024 as TLS named groups. The distinction between the standardized NIST algorithm and its evolving TLS integration is important: algorithm standardization does not imply that every TLS protocol integration is finalized.

5.3 Hybrid Key Establishment

Hybrid key establishment combines a classical exchange with a PQ KEM. `X25519MLKEM768` is a representative construction [KKWS26]. Its transitional rationale is that the final keying material incorporates both components, so security can remain dependent on the surviving component if one component is compromised. Hybridization therefore addresses two forms of uncertainty: the known long-term quantum vulnerability of classical public-key cryptography and the residual cryptanalytic uncertainty associated with newer PQ algorithms. The exact

combiner and transcript integration must follow the protocol specification. Simply concatenating two secrets is not, by itself, a complete hybrid construction.

5.4 PSK-Based Approaches

PSKs can provide authentication and, depending on the mode, contribute directly to key establishment. Their value is greatest in environments with controlled provisioning, such as private APIs, enterprise networks, managed infrastructure, and pre-provisioned devices. PSK- only establishment does not automatically provide the same forward-secrecy properties as fresh ephemeral exchange. PSK plus ephemeral exchange can retain fresh session entropy, while PSK plus PQ or hybrid key establishment can combine controlled authentication with quantum-resistant confidentiality. The appropriate classification question is therefore not “PSK or PQC?” but: How was the PSK generated? How was it provisioned and protected? What role does it play in the TLS key schedule? Does it originate from a quantum-resistant connection? Is independent ephemeral key establishment performed?

5.5 Post-Quantum Authentication

Post-quantum authentication replaces or augments classical signature mechanisms. ML-DSA is the principal standardized lattice-based signature mechanism considered here,

while SLH-DSA provides a hash-based alternative [Nat24a, Nat24c]. PQ authentication protects identity and certificate-related operations, but it does not automatically protect the confidentiality of sessions whose keys were established using a classical mechanism.

5.6 Hybrid Authentication

Hybrid authentication combines classical and PQ signatures. Its transition rationale is analogous to hybrid KEMs: security can be sought if at least one signature system remains secure. The engineering challenge is greater than for a hybrid handshake because certificates must be encoded, signed, validated, chained, stored, distributed, and understood by relying parties. Hybrid signatures therefore require coordination across TLS implementations, certificate authorities, HSMs, operating systems, browsers, trust stores, and certificate-management systems.

5.7 PKI and Certificate Migration

Post-quantum certificate migration affects end-entity certificates, intermediate CAs, roots, trust stores, issuance systems, validation software, certificate transparency infrastructure, HSMs, and client compatibility. A deployment can migrate at several levels: PQ leaves under classical intermediates, PQ chains, hybrid or composite certificates, or parallel classical and PQ chains. These alternatives have different compatibility and security properties and should not be treated as equivalent.

5.8 Crypto-Agile Migration

Crypto-agility is an architectural property rather than a cryptographic algorithm. A crypto-agile system can replace cryptographic mechanisms without redesigning the surrounding

application. For PQ migration, crypto-agility includes algorithm negotiation, cryptographic inven- tory, modular TLS libraries, certificate lifecycle automation, configurable trust stores, support for multiple certificate representations, algorithm policy controls, interoperability test matrices, telemetry, and controlled rollback.

6 Hybrid Key Establishment, PSK, KeyUpdate, and Long-Lived Sessions

6.1 Hybrid ECDHE-ML-KEM

The principal transitional hybrid architecture combines ephemeral ECDHE with ML-KEM. In a representative X25519MLKEM768 configuration, X25519 supplies the classical component and ML-KEM-768 supplies the PQ component [KKWS26]. Let Z_C denote the classical shared secret and Z_Q the PQ shared secret. A hybrid construction produces a value Z_H that is incorporated into the TLS key schedule. The security objective is that compromise of one component alone should not reveal the final session secret if the other component remains secure. The classical component retains a mature implementation ecosystem and classical forward secrecy. The PQ component contributes resistance to future attacks against the classical exchange. The hybrid design therefore provides a transitional architecture rather than simply two independent exchanges executed side by side.

$$Z_H = \text{Combine}(Z_C, Z_Q). \tag{3}$$

6.2 Forward Secrecy

Forward secrecy and quantum resistance are related but distinct. Classical ephemeral ECDHE provides forward secrecy against compromise of long-term authentication keys under the classical threat model. It does not protect a recorded session against a future quantum attacker capable of solving the underlying discrete-logarithm problem. Hybrid ECDHE-`ML-KEM` can combine the classical ephemeral property with a PQ component. The precise forward-secrecy claim depends on ephemeral generation, secret deletion, KEM security, transcript binding, and implementation correctness.

6.3 Downgrade Resistance

Migration creates multiple supported groups. A client may offer classical `X25519`, hybrid `X25519MLKEM768`, standalone `ML-KEM`, or other supported groups. An active attacker may attempt to force a connection toward a classical-only mechanism. A secure deployment therefore needs explicit policy defining whether PQ is mandatory, preferred, or optional; whether classical fallback is allowed; and how fallback is logged and monitored. Negotiated security parameters must remain protected by TLS transcript authentication.

6.4 PSK Authentication Versus PSK Key Establishment

A PSK can serve as an authentication credential without being the sole source of session entropy. This distinction is essential in migration analysis. PSK-only establishment can be efficient, particularly in controlled systems, but its security depends strongly on the PSK. If a

PSK is reused, weakly generated, improperly stored, or compromised, the authentication and key-establishment consequences can be substantial. PSK plus ephemeral exchange introduces fresh session key material. In a PQ migration, the ephemeral exchange can be a PQ KEM or a hybrid ECDHE-`ML-KEM` construction.

6.5 Resumption PSKs

TLS 1.3 resumption PSKs derive from an earlier connection. Their security must therefore be evaluated according to the properties of the connection that produced them. If the original connection was classical-only, the resulting resumption PSK should not be treated as evidence that the original connection was quantum-resistant. A migration policy should record the provenance of resumption credentials and determine whether they may be used to establish sessions subject to a PQ security requirement. This creates a practical migration question that is often overlooked: session resumption can preserve deployment behavior from an earlier cryptographic generation. Migrating the

6.6 External PSKs

External PSKs are particularly useful in managed environments because organizations can control provisioning and lifecycle operations [BW22]. Security management should include generation, secure provisioning, storage, rotation, revocation, uniqueness, algorithm binding, endpoint compromise, and prevention of cross-protocol reuse. The critical property is provenance. A PSK is not post-quantum merely because it is symmetric. Its role in the TLS key schedule and the mechanism from which its entropy originated determine the relevant security properties.

Table 3: Hybrid KEM, PSK, and KeyUpdate Comparison

Mechanism	Fresh public-key/KEM exchange	Primary role	PQ by itself	Forward-secrecy characteristics	Principal concern
PSK-only	No	Authentication and/or key establishment	Depends on PSK provenance	Not equivalent to fresh ephemeral exchange	PSK lifecycle
PSK + ECDHE	Yes	Authentication plus classical key establishment	No	Classical forward secrecy	Classical quantum vulnerability
PSK + ML-KEM	Yes	Authentication plus PQ key establishment	Yes for key establishment	Depends on ephemeral KEM design	Provisioning and integration
PSK + hybrid KEM	Yes	Authentication plus hybrid establishment	Yes for key establishment	Strong transitional properties	Provisioning/KEM complexity
KeyUpdate	No	Traffic-key evolution	No	Useful under appropriate secret-deletion assumptions	No independent PQ secret
Fresh PQ rekey	Yes	New PQ key establishment	Yes	Fresh PQ secret	Protocol support and cost

6.7 KeyUpdate

KeyUpdate derives a new traffic secret from an existing one. It can reduce the volume of traffic encrypted under a single traffic key and can provide useful security properties following some endpoint compromises when old secrets are securely deleted [Res26]. It does not perform ML- KEM. It does not perform ECDHE. It does not introduce a new independent public-key secret. It does not repair a vulnerable certificate. It does not repair a vulnerable initial handshake. Therefore, KeyUpdate is a traffic-key lifecycle mechanism, not a PQ migration mecha- nism. A fresh PQ KEM exchange has a different security role because it introduces new keying material through a new encapsulation/decapsulation operation. KeyUpdate and fresh PQ rekeying should consequently be viewed as complementary mechanisms rather than substitutes.

6.8 Fresh PQ Rekeying

Fresh PQ rekeying is analytically distinct from KeyUpdate because it introduces new PQ-derived keying material through a new KEM exchange. Its security value therefore depends on the exact protocol construction, the freshness and deletion of the relevant secrets, and the absence of an attacker-controlled endpoint. It should be evaluated as an additional security boundary for long-lived sessions, not as a more expensive form of KeyUpdate. The appropriate trigger is deployment-specific. Relevant variables include confidentiality lifetime, HNDL exposure, connection duration, endpoint capability, protocol support, and the cost of another public-key exchange. The evidence reviewed here does not justify a universal byte, time, or message-count threshold.

6.9 Long-Lived Sessions

Table 3 summarizes the resulting distinctions among PSKs, hybrid key establishment, KeyUpdate, and fresh PQ rekeying. Long-lived TLS connections require special treatment. A connection may involve an initial hybrid handshake, multiple KeyUpdates, resumption credentials, and eventually endpoint compromise. These events do not provide equivalent security. For long-lived connections with stringent confidentiality requirements, migration policy should define when a fresh PQ key establishment is required rather than relying indefinitely on

symmetric KeyUpdate. The appropriate threshold depends on the threat model, data lifetime, protocol support, and operational cost.

Table 4: Security-State Analysis of Long-Lived Sessions

Scenario	New asymmetric/KE exchange?	Independent fresh PQ secret?	HNDL status of new session	Principal caveat
Classical handshake → KeyUpdate	No	No	Not repaired	KeyUpdate remains downstream of the prior traffic-secret lineage
Hybrid handshake → KeyUpdate	No	No additional PQ secret	Inherits hybrid foundation	Does not create a second PQ assumption
Classical handshake → resumption PSK-only	No	No	Inherits earlier security dependency	No fresh ephemeral exchange
Classical handshake → resumption PSK+ECDHE	Yes	No	New classical ephemeral component	Still quantum-vulnerable at ECDHE
Hybrid handshake → resumption PSK+hybrid exchange	Yes	Yes	Fresh hybrid establishment	Requires compatible TLS integration
Hybrid handshake → KeyUpdate → fresh PQ rekey	Yes at rekey	Yes	Fresh PQ foundation after rekey	Protocol support and operational cost
Any handshake → endpoint compromise → KeyUpdate	No	No	Does not reverse prior compromise	Persistent compromise defeats the distinction

6.10 Security-State Analysis of Long-Lived Sessions

The most useful way to distinguish PSKs, KeyUpdate, resumption, and fresh PQ establishment is to follow the origin of the secret material rather than the visible event that changes the connection state.

The table supports a precise statement: the security significance of a rekey event is determined by whether independent entropy enters the key schedule. KeyUpdate changes the traffic-secret generation but does not introduce an independent KEM secret. Resumption can create a new connection while remaining cryptographically tied to earlier state. A fresh PQ exchange, by contrast, introduces new PQ-derived keying material and therefore can establish a new security foundation. Table 4 summarizes these security-state transitions.

6.11 Classical Handshake Followed by KeyUpdate

Let the application traffic secret produced by the initial handshake be S_0. TLS 1.3 derives subsequent traffic secrets as $S_{i+1} = g(S_i)$. If S_0 ultimately depends on a quantum-vulnerable ECDHE exchange, KeyUpdate does not introduce an independent post-quantum secret. Therefore a future attacker capable of recovering the original classical handshake secret does not encounter a new PQ boundary merely because the endpoint performed one or more KeyUpdates. This does not make KeyUpdate useless. If prior traffic secrets are correctly erased, compromise of a later traffic secret need not expose earlier generations. That property is valuable under appropriate temporary-compromise models. It is, however, a property of symmetric key evolution rather than post-quantum re-establishment.

6.12 Resumption as a Security-Lineage Problem

TLS 1.3 resumption credentials are derived from earlier handshake state. Consequently, a migration policy should track the provenance of resumption credentials rather than classifying every PSK as either “secure” or “insecure.” A resumption PSK produced by a hybrid handshake

is not equivalent, in provenance, to one produced by a classical-only handshake. Moreover, a resumed handshake can combine the PSK with a fresh asymmetric exchange, changing the security properties of the new connection. The practical implication is that

cryptographic inventories should record not only the existence of PSK support but also the generation path, expiration policy, ticket lifetime, and whether resumed sessions are required to perform a fresh PQ-capable exchange.

6.13 Post-Compromise Scenarios

The analysis distinguishes four compromise points: compromise of a long-term authentication key, compromise of a current traffic secret, compromise of a resumption credential, and persistent endpoint compromise. These are not interchangeable. KeyUpdate is relevant primarily to the traffic-secret lineage; certificate migration is relevant to long-term authentication; resumption policy controls the lifetime of inherited connection state; and no protocol-level rekeying mechanism can protect an endpoint that remains under persistent attacker control. The unresolved policy problem is therefore not "how often should TLS KeyUpdate run?" but "under what conditions should a deployment require a fresh PQ-capable key establishment?" The answer depends on confidentiality lifetime, threat model, protocol support, endpoint capability, connection duration, and operational cost. The literature does not justify a universal time or byte threshold.

7 Post-Quantum Authentication and PKI

7.1 ML-DSA and SLH-DSA

NIST FIPS 204 standardizes `ML-DSA` as a lattice-based post-quantum signature mechanism [Nat24a]. FIPS 205 standardizes `SLH-DSA` as a stateless hash-based alternative [Nat24c]. The two algorithms provide different cryptographic assumptions and operational trade- offs. ML- DSA is directly relevant to TLS authentication and certificate infrastructure. `SLH-DSA` provides diversification through hash-based assumptions but can involve larger signatures and different signing and verification costs; RFC 9909 specifies SLH-DSA algorithm identifiers for X.509/PKIX. The important migration distinction is that these algorithms address authentication. They do not replace `ML-KEM`'s role in session-key establishment.

7.2 CertificateVerify and TLS Authentication

TLS authentication uses signatures to authenticate handshake state and, in certificate-based deployments, uses certificates to bind identities to public keys [Res26]. A post-quantum authentication deployment therefore needs compatible signature schemes for CertificateVerify and certificate-related operations. The IETF TLS working group has developed work describing the use of `ML-DSA` in TLS 1.3 [HSW26]. As of 15 August 2026, the document was at draft-ietf-tls-mldsa-05 and had been approved for publication as an Informational RFC, but publication was being held following an appeal and it had not yet been published as an RFC. Such work illustrates the difference between an algorithm becoming a NIST standard and the broader protocol ecosystem becoming deployment-ready.

7.3 X.509 and PKIX

RFC 5280 defines the Internet X.509 PKI profile and certification-path validation model [C+08]. PQ certificate migration consequently involves more than adding a signature algorithm. The infrastructure must support PQ public-key encoding, signature representation, certificate policies, signature-scheme negotiation, certificate issuance, validation, trust-store updates, and mixed classical/PQ chains. A certificate chain may include an end-entity certificate, intermediates, and a trust anchor. Migration can occur at each level,

Table 5: PQ Authentication and PKI Migration

Strategy	Signature mechanism	Compatibility	Certificate impact	Deployment complexity	Main issue
Classical certificate	RSA/ECDSA	Highest	None	Low	Quantum-vulnerable authentication
PQ leaf	ML-DSA	Requires PQ-capable clients	Moderate	Medium	Partial-chain security and compatibility
PQ full chain	ML-DSA/SLH-DSA	Requires PQ-capable PKI	High	High	Trust-anchor and verifier migration
Hybrid/composite	Classical + PQ	Potentially transitional	High	High	Encoding, validation, and ecosystem support
Parallel chains	Classical and PQ alternatives	Can preserve controlled fallback	High	High	Policy and chain-selection complexity
KEM-based authentication	KEM-based	Requires protocol/PKI changes	Potentially different	Very high	Research and architectural maturity

but a PQ leaf under a classical root does not have the same authentication properties as a fully PQ chain.

7.4 CA and Trust-Anchor Migration

CA migration is slower than server-key migration because CA certificates often have long validity periods and are distributed through browsers, operating systems, applications, embedded devices, and enterprise trust stores. The central operational problem is verifier availability. A CA may be able to generate a PQ certificate before browsers, operating systems, HSMs, and application libraries can validate it. Standardizing a signature algorithm therefore does not establish universal deployment compatibility. Trust anchors are particularly difficult because they are frequently installed outside ordinary certificate-update mechanisms. Migration requires overlapping validity periods, distribution mechanisms, rollback plans, client testing, embedded trust-store updates, and controlled retirement of classical trust anchors.

7.5 Hybrid and Composite Certificates

Hybrid or composite certificates can provide transitional compatibility by combining classical and PQ signatures. However, certificate hybridization is operationally complex. The certificate must be encoded, signed, validated, chained, distributed, and interpreted consistently by all relevant parties. IETF work on composite `ML-DSA` signatures addresses some of these questions for X.509/PKIX applications. Such mechanisms should remain classified according to their actual standardization status rather than being described as finalized Internet standards.

7.6 HSMs

Enterprise CA keys are frequently protected by HSMs. PQ migration therefore requires HSM support for key generation, secure storage, signing, backup, recovery, certificate issuance, rotation, and audit. This creates a recurring migration asymmetry: a cryptographic algorithm can be standardized before the operational hardware required to deploy it at scale is mature. HSM readiness should consequently be treated as a first-class migration dependency rather than as an implementation detail.

7.7 Client and Browser Compatibility

Table 5 summarizes the principal certificate migration strategies. Authentication migration is ecosystem-dependent. A client must recognize the relevant signature algorithm, parse the certificate, validate the chain, trust the appropriate CA, support the TLS signature scheme, and correctly process the certificate representation. This dependency makes authentication migration slower than many key-establishment deployments. A server cannot unilaterally assume that a PQ certificate will work across all clients.

8 Performance, Interoperability, and Deployment

8.1 Interpreting Performance Evidence

Performance results depend on CPU architecture, cryptographic library, compiler, hardware acceleration, operating system, TLS implementation, security parameter, network RTT, packet loss, congestion, transport protocol, certificate-chain length, resumption rate, and concurrent connection count. Consequently, a primitive microbenchmark and an end-to-end TLS benchmark answer different questions. A KEM may consume modest CPU time while its larger messages increase network latency. Conversely, a network with abundant bandwidth and low RTT may make message expansion less significant. The literature should therefore be interpreted by measurement layer rather than by selecting one benchmark number as universally representative.

8.2 Key and Handshake Size

FIPS 203 specifies a 1,184-byte public key and a 1,088-byte ciphertext for `ML-KEM-768` [Nat24b]. These values make communication overhead an important migration consideration.

The impact is environment-dependent. Large key shares are more consequential on mobile, satellite, IoT, embedded, and high-latency networks than on high-bandwidth local networks.

8.3 Computational Cost

The rustls performance study reported measurable CPU cost for `X25519MLKEM768` while emphasizing that the primary benchmark did not include all networking consequences of larger messages [rus24]. CoNEXT 2023 research similarly examined post-quantum TLS 1.3 performance under specific experimental conditions [SWH+23]. More recent work evaluates classical, hybrid, and pure-PQ configurations using controlled performance frameworks [M+ 26]. These results support an analytical conclusion rather than a universal numerical claim: PQ TLS can introduce both computational and communication costs, and the relative importance of each depends on the deployment environment.

8.4 Certificate Size and Latency

Certificate chains may become a major source of PQ handshake expansion. A server can transmit an end-entity certificate, intermediate certificates, extensions, CertificateVerify, and other handshake data. Consequently, even if an individual PQ signature is computationally manageable, the complete handshake can become significantly larger. Larger messages can lead to additional packets, fragmentation, memory consumption, congestion, and latency. Recent research specifically considers the network impact of PQ certificate-chain sizes [CC26].

Table 6: Performance and Interoperability Evidence

Evidence	Scope	Main observation	Interpretation
NIST ML-KEM material	Primitive sizes	ML-KEM-768 uses substantially larger key material than classical ECDH	Communication size must be considered explicitly
CoNEXT 2023	PQ TLS 1.3	Evaluated performance effects under a defined experimental environment	Results are environment-specific
rustls study	X25519 versus X25519MLKEM768	Measurable CPU cost; networking not fully represented	CPU benchmarks are not end-to-end deployment measurements
Performance evaluation framework	Classical, hybrid, and PQ TLS	Evaluates cryptographic and TLS-level overhead	Controlled profiles are necessary for comparison
PQ certificate research	Certificate chains	Larger PQ chains can affect network-level latency	Transport and deployment architecture influence results

8.5 Interoperability

Interoperability requires more than algorithm-level compatibility. A complete test matrix should include client and server libraries, operating systems, browsers, certificate types, supported groups, signature algorithms, resumption, `0-RTT`, `HelloRetryRequest`, proxies, and middleboxes. Historical experimental Kyber identifiers should not automatically be treated as equiv- alent to current `ML-KEM` identifiers. Implementations must agree on exact named groups, wire formats, combiners, and protocol behavior.

8.6 Middleboxes and Large Messages

Middleboxes may react differently to unfamiliar extensions or larger handshake messages. Migration must therefore test the complete communication path rather than only direct client- server connections. A system that successfully negotiates hybrid TLS in a laboratory may still fail when deployed through a proxy, load balancer, inspection device, or legacy network component.

8.7 Constrained Devices

Embedded systems add memory, power, firmware-lifecycle, and update constraints. Their migration cost may be dominated by the difficulty of deploying software updates rather than by raw cryptographic CPU cost. For constrained devices, the optimization objective is therefore better represented as a combination of security, bandwidth, memory, energy, implementation size, and lifecycle constraints rather than CPU time alone.

8.8 Cloud and Enterprise Infrastructure

Table 6 summarizes the principal performance evidence. Cloud environments benefit from centralized software updates but also contain large numbers of TLS termination points, load balancers, service meshes, proxies, and internal service-to-service connections. A cloud deployment can migrate Internet-facing TLS while leaving internal connections classical. Enterprise PKI introduces additional dependencies: internal CAs, HSMs, automated issuance, legacy applications, device certificates, VPNs, smart cards, and application-specific trust stores. A complete inventory must therefore include both network-facing and internal cryptographic dependencies.

8.9 Standards Status as of 15 August 2026

The standards landscape must be represented at the correct layer. Table 7 summarizes the snapshot. As of the manuscript's update date, TLS 1.3 is specified by RFC 9846, a Proposed Standard that obsoletes RFC 8446 [Res26] while retaining the TLS 1.3 version

Table 7: Standards Status as of 15 August 2026

Layer	Status at revision date	Interpretation
TLS 1.3	RFC 9846, Proposed Standard	Current TLS 1.3 specification; obsoletes RFC 8446
External PSK import	RFC 9258, Proposed Standard	Mechanism for importing external PSKs
X.509/PKIX baseline	RFC 5280 and updates	Established certificate/CRL profile
ML-KEM	NIST FIPS 203	Finalized NIST KEM standard
ML-DSA	NIST FIPS 204	Finalized NIST signature standard
SLH-DSA	NIST FIPS 205	Finalized NIST signature standard
ML-DSA in X.509	RFC 9881, Proposed Standard	PKIX algorithm-identifier specification
SLH-DSA in X.509	RFC 9909, Proposed Standard	PKIX algorithm-identifier specification
General TLS hybrid construction	RFC 9954, Informational	IETF-reviewed hybrid construction; not Standards Track
ECDHE-MLKEM TLS groups	RFC 10024, Standards Track (Proposed Standard)	Three standardized PQ/T hybrid TLS 1.3 key-agreement mechanisms
ML-KEM TLS groups	draft-ietf-tls-mlkem-09	Active Internet-Draft; IESG Evaluation as of 14 August 2026
ML-DSA in TLS 1.3	draft-ietf-tls-mldsa-05	Active Internet-Draft; IESG-approved but publication held as of 15 August 2026

number and making targeted clarifications. `ML-KEM`, `ML-DSA`, and `SLH-DSA` are finalized NIST standards in FIPS 203, 204, and 205. RFC 9881 and RFC 9909, both Proposed Standards, specify `ML-DSA` and `SLH-DSA` algorithm identifiers for X.509/PKIX, respectively [MKTW25, BFG+25]. RFC 9954 provides a general hybrid key-exchange construction for TLS 1.3 as an Informational RFC [SFG26]; it does not itself select particular PQ algorithms. RFC 10024, a Standards Track Proposed Standard published in July 2026 [KKWS26], specifies three PQ/T hybrid TLS 1.3 key-agreement mechanisms: X25519MLKEM768, SecP256r1MLKEM768, and SecP384r1MLKEM1024. The standalone `ML-KEM` TLS integration remained an Internet-Draft in IESG Evaluation as of 15 August 2026, while the ML-DSA TLS authentication draft remained an Internet-Draft whose publication was being held following an appeal; neither was an RFC by the snapshot date [Con26, HSW26].

9 Migration Framework and Comparative Analysis

9.1 Migration Principle

The evidence supports treating PQ TLS as a staged migration architecture. The following sequence is an analytical synthesis of standards and published evidence rather than a formally validated algorithm or universal timetable:

1. Cryptographic inventory. Identify TLS libraries, key-exchange groups, signature algorithms, certificates, certificate authorities, trust stores, HSMs, protocols, lifetimes, dependencies, and owners.

2. Identify HNDL-sensitive information. Prioritize information whose confiden- tiality must survive for years or decades because present-day collection creates future decryption risk [Nat26].

3. Assess TLS dependencies. Include clients, servers, proxies, middleboxes, load balancers, service meshes, certificate-management systems, HSMs, and embedded cryptographic dependencies.

4. Deploy and test hybrid key establishment. Where interoperability and im- plementation support permit, evaluate hybrid ECDHE-`ML-KEM` using negotiation success, handshake size, CPU use, latency, and fallback behavior.

Table 8: Migration Framework by Deployment Environment

Environment	Preferred transitional strategy	Principal dependency
Public web	Hybrid ECDHE-ML-KEM	Client/TLS-stack support
Cloud	Hybrid KEM + staged PQ authentication	Edge/internal TLS inventory
Enterprise	Hybrid KEM + controlled PKI	CA/HSM/trust-store control
VPN	Hybrid PQ key establishment	Endpoint upgradeability
Controlled API	Hybrid KEM; PSK if appropriate	Secure provisioning
IoT	Constrained hybrid/PQ profile	Memory/energy/bandwidth/firmware
Embedded	Hybrid/constrained PQ	Long firmware lifetime
Financial	Hybrid KEM + staged PQ PKI	HSMs/long confidentiality
Government	Hybrid/PQ per policy	Approved implementations/compliance
Critical infrastructure	Hybrid + staged authentication	Long lifetimes/difficult updates
Archival	PQ protection + re-encryption planning	Long-term key management

5. Test interoperability. Cover clients, servers, browsers, operating systems, middleboxes, proxies, resumption, `0-RTT`, `HelloRetryRequest`, certificates, KeyUpdate, and downgrade scenarios.

6. Prepare PQ authentication and PKI. Test PQ signatures, hybrid/composite certificates, certificate issuance and validation, CT infrastructure, and trust stores.

7. Prepare HSM and trust-store infrastructure. Include HSMs, CA software, key-management systems, operating systems, browsers, and embedded devices.

8. Pilot PQ authentication. Introduce PQ authentication first where endpoint and client populations are sufficiently controlled to permit interoperability testing and rollback.

9. Retire classical-only mechanisms according to risk. Base retirement on data lifetime, endpoint support, compliance requirements, and observed interoperability; keep fallback controlled and observable during transition.

9.2 Deployment by Environment

Table 8 summarizes the environment-specific priorities. Public web servers should prioritize hybrid key establishment because Internet traffic is exposed to HNDL collection and interoperability is essential. Cloud services should inventory both external and internal TLS. Service-to-service traffic may contain large quantities of sensitive information even when the public edge has already migrated. Enterprise networks are well suited to early controlled experimentation because admin- istrators can coordinate software, trust stores, certificates, policies, and inventory. VPN deployments deserve high priority where they protect long-lived sensitive commu- nications, subject to the underlying protocol's support for the relevant PQ mechanisms. Controlled APIs can use PSK-based approaches when secret provisioning is secure and the client population is managed. Public APIs should generally favor broadly interoperable TLS mechanisms. IoT and embedded systems require optimization across memory, energy, bandwidth, firmware lifecycle, and cryptographic security. Early crypto-agility is particularly important because devices may remain deployed for many years. Financial, government, and critical-infrastructure environments require additional attention to HNDL-sensitive data, HSM

readiness, compliance requirements, long service lifetimes, and difficult update procedures. Archival systems present a particularly strong HNDL case. Long-term protection may require re-encryption, key wrapping, hybrid protection, PQ-capable key management, and preservation of migration metadata.

Table 9: Threat-to-Security-Property Matrix

Threat	Required property	Relevant mechanism(s) / boundary
HNDL against recorded TLS sessions	Quantum-resistant session-key establishment	Hybrid or pure-PQ key establishment; does not automatically migrate authentication
Future quantum attack on classical key exchange	PQ/hybrid key establishment	ML-KEM or ECDHE-ML-KEM; does not repair previously recorded classical traffic
Future quantum attack on signatures	PQ authentication	ML-DSA, SLH-DSA, or hybrid authentication; no retroactive session confidentiality
Long-term classical key compromise	Forward secrecy and PQ authentication	Ephemeral ECDHE and PQ signatures; does not protect a persistently compromised endpoint
Downgrade to classical-only	Negotiation integrity and policy	Transcript binding, explicit policy, and monitoring
Certificate compromise	Revocation, replacement, and trust management	PKIX controls, CA procedures, and trust-anchor migration
Resumption credential compromise	Credential lifecycle and fresh exchange policy	Ticket management, expiration, and fresh PQ/hybrid exchange
Temporary endpoint compromise	Key evolution and secret deletion	KeyUpdate plus deletion; persistent active compromise remains outside its protection
Middlebox incompatibility	End-to-end interoperability	Test matrix and controlled fallback; laboratory success is not production proof
HSM incompatibility	Hardware-backed key-management support	HSM upgrade/testing and staged PKI migration

9.3 Comparison with Previous Surveys

Alnahawi et al. provided a broad 2024 survey of post-quantum TLS, including classification and extensive performance evaluation [AMOW24]. Its value lies in establishing a comprehensive view of candidate PQ TLS mechanisms and their performance at that stage of the field. The present SoK differs primarily in analytical decomposition. Rather than treating PQ TLS primarily as a collection of candidate key-exchange mechanisms, it systematizes the migration as a combination of key establishment, authentication, PSK/resumption, KeyUpdate, PKI, interoperability, deployment, and crypto-agility. This distinction is motivated by standards evolution since 2024 rather than by publication date alone. Broader surveys of quantum-safe network protocols provide useful context across TLS, IPsec, SSH, PGP, and related systems [BCH24], while PQC migration reviews identify organizational challenges such as implementation effort, security uncertainty, and migration complexity [NHG+24]. The present work specializes those concerns to TLS and its certificate ecosystem. The resulting distinction is: PQC TLS $\neq$ one cryptographic replacement. Instead:

Key establishment + authentication + resumption + key evolution + PKI + deployment + crypto-agility. This decomposition prevents two recurring analytical errors: treating KeyUpdate as fresh PQ establishment and assuming that replacing a KEM automatically produces quantum-resistant authentication.

9.4 Threat-to-Security-Property Matrix

The matrix makes the central thesis operational: each threat maps to a security property, and each mechanism has a boundary beyond which it should not be credited. Table 9 provides the resulting mapping.

9.5 Security Property Matrix

The following matrix summarizes the security-property interpretation used throughout the SoK. “Depends” is intentional: several properties depend on the exact protocol mode, secret provenance, implementation, or compromise model.

Table 10 reinforces the central analytical rule: a mechanism should receive credit only for the security property it actually changes. In particular, KeyUpdate should not be marked as providing fresh PQ entropy, and PQ authentication should not be credited with retroactive confidentiality for sessions established using quantum-vulnerable key exchange.

Table 10: Security Property Interpretation

Mechanism	HNDL	PQ conf.	PQ auth.	Fresh PQ	Traffic-key	Lineage
ECDHE	No	No	No	No	No	Classical ephemeral foundation
ML-KEM	Yes for newly established sessions under PQ assumptions	Yes for key establishment	No	Yes	No	Depends on protocol and endpoint assumptions
Hybrid ECDHE-ML-KEM	Yes for newly established sessions under hybrid assumptions	Yes	No	Yes	No	Hybrid foundation
PSK-only	Depends on provenance and entropy	Depends on PSK origin/use	Yes when used for PSK authentication	Not inherently	No	PSK lineage
PSK + ECDHE	No against future quantum attack on ECDHE	No	Yes	No PQ entropy	No	PSK plus classical ephemeral exchange
PSK + PQ KEM	Yes for newly established sessions	Yes	Yes	Yes	No	Managed authentication plus fresh PQ establishment
PSK + hybrid KEM	Yes for newly established sessions	Yes	Yes	Yes	No	Managed authentication plus hybrid exchange
KeyUpdate	No new PQ foundation	Inherits initial state	No	No	Yes	Downstream of traffic secret
Fresh PQ rekey	Yes for newly established state	Yes for new state	Not inherently	Yes	Re-establishes rather than merely evolves	New PQ security boundary
PQ authentication	Not applicable to past confidentiality	No	Yes	No	No	Auth./cert. lineage

10 Research Gaps and Discussion

10.1 PQ Authentication and PKI

The literature and standards activity indicate that key-establishment migration is progressing faster than universal PQ authentication deployment. `ML-DSA` and `SLH-DSA` are standardized algorithms, but deployment requires TLS signature integration, X.509 support, CA support, HSM support, client validation, and trust-store migration. The research need is therefore not another signature algorithm alone. It is end-to- end evaluation of certificate issuance, validation, chaining, trust-anchor migration, HSM integration, and client interoperability.

10.2 Hybrid Interoperability

Hybrid TLS implementations can differ in supported groups, draft revisions, wire formats, certificate handling, and fallback behavior. Interoperability research should therefore move from isolated primitive tests toward standardized cross-implementation test matrices. A useful test suite should include initial key exchange, authentication, resumption, `0-RTT`, `HelloRetryRequest`, certificate chains, KeyUpdate, error handling, downgrade scenarios, and middlebox behavior.

10.3 Performance and Benchmark Standardization

Published studies do not establish one universal PQ TLS performance penalty. CPU-oriented measurements can show modest computational overhead while network-level experiments reveal significant latency effects from larger handshake messages. The apparent disagreement is largely explained by measurement scope. Future bench- mark suites should report hardware, operating system, TLS implementation, cryptographic library, compiler, security level, network RTT, packet loss, certificate chain, resumption rate, and concurrency. Without this metadata, benchmark values are difficult to compare.

10.4 Migration Automation and Cryptographic Inventory

Organizations cannot migrate cryptography they cannot identify. Public-key dependencies may exist in TLS libraries, certificates, CA systems, HSMs, software, firmware, APIs, and embedded devices. Migration automation should discover TLS versions, key-exchange groups, signature algorithms, certificates, CA dependencies, HSM dependencies, software libraries, hard- coded algorithms, and embedded cryptographic components. A useful inventory should record at least: (system, algorithm, key, certificate, protocol, lifetime, dependency, owner). This enables risk prioritization according to data lifetime and migration difficulty.

10.5 Long-Lived Sessions, PSKs, and Post-Compromise Recovery

The interaction among initial hybrid handshakes, resumption PSKs, KeyUpdate, fresh PQ rekeying, and endpoint compromise remains insufficiently systematized. Research should distinguish passive compromise, temporary endpoint compromise, persistent compromise, compromise before and after KeyUpdate, compromise of the original classical handshake, compromise of a PQ handshake, and compromise of resumption credentials. The particularly important question is when a long-lived session should perform fresh PQ key establishment rather than relying on symmetric key evolution.

10.6 HSM and Certificate Infrastructure

HSM migration remains an infrastructure bottleneck. Research should evaluate PQ key generation, signing, secure backup, rotation, hybrid signing, certificate issuance, and performance under high CA loads. Certificate-transparency infrastructure also requires attention because larger PQ certifi- cates may increase storage, submission bandwidth, proof- size, and monitoring requirements.

10.7 What the Literature Agrees On

The evidence consistently supports several conclusions: classical public-key mechanisms create a long-term quantum migration problem; HNDL makes migration relevant before a cryptographically relevant quantum computer exists [Nat26]; `ML-KEM` is standardized by NIST [Nat24b]; hybrid key establishment is standardized for specific TLS 1.3 groups in RFC 10024 [KKWS26]; authentication and key establishment are separate migration problems; and cryptographic inventory and crypto-agility are operational prerequisites.

10.8 Where Evidence Differs

The most visible differences concern performance. Some studies report relatively small computational overheads, whereas network-oriented evaluations show that larger handshake messages can increase latency under specific conditions [rus24, SWH+23, M+26]. These observations are not necessarily contradictory. A primitive benchmark measures

cryptographic computation, whereas a deployment benchmark measures complete protocol behavior over a network. Both can be correct. A similar distinction applies to deployment maturity. The existence of an implementa- tion demonstrates technical feasibility but does not establish universal interoperability, standardized wire behavior, client support, HSM support, or production readiness.

10.9 What Is Mature and What Is Evolving

TLS 1.3, classical ECDHE, X.509/PKIX, and the NIST PQ algorithms are comparatively mature components. The exact TLS integration of PQ algorithms, PQ authentication, hybrid certificate formats, widespread PQ trust-anchor deployment, HSM support, and cross-vendor interoperability remain more fluid; several relevant IETF documents remain Internet-Drafts or are Informational RFCs. This distinction is critical to responsible migration planning. An

organization can begin inventory and hybrid experimentation even while avoiding claims that every component of the eventual PQ ecosystem is finalized.

10.10 Integrated Findings

Across the evidence, the strongest conclusions are those that survive changes in implementation and standards version. Finding 1 — Migration is multidimensional. Key establishment, authentication, resumption, traffic-key evolution, PKI, and deployment are related components of one TLS architecture but are not substitutes for one another. Finding 2 — Fresh entropy is the decisive distinction for re-establishment. KeyUpdate and resumption can change connection state without introducing a new PQ foundation. A fresh PQ-capable exchange is analytically different because new KEM-derived entropy enters the key schedule. Finding 3 — Hybridization is transitional rather than terminal. Hybrid ECDHE-`ML-KEM` provides a rationale for retaining a mature classical component while adding a PQ component. It should therefore be evaluated as a migration architecture, not as proof that all uncertainty has disappeared. Finding 4 — Authentication migration is an ecosystem problem. The cryptographic algorithm, TLS signature negotiation, certificate representation, CA issuance, HSM, trust anchor, and verifier must all align. A deployment can be cryptographically capable but operationally unable to validate the resulting certificate chain. Finding 5 — Performance is a systems property. Primitive cost, TLS handshake cost, network cost, and certificate-chain cost answer different questions. A single percentage overhead is therefore not a meaningful universal characterization. Finding 6 — Inventory and crypto-agility are prerequisites. An organization cannot systematically migrate cryptography that it cannot identify, and it cannot stage replacement safely if cryptographic mechanisms are hard-coded across applications and infrastructure. Finding 7 — Evidence maturity must be explicit. Finalized standards, active drafts, implementations, experiments, and production deployment are different evidence classes. Conflating them creates misleading maturity claims. These findings are the basis for the migration framework rather than consequences of a particular implementation benchmark.

11 Limitations

This SoK has several limitations. First, the standards landscape is rapidly evolving. Internet- Drafts discussed here may become RFCs, change substantially, or be replaced after manuscript preparation. Second, the literature has publication bias toward successful prototypes and positive benchmark results. Failed migrations and negative interoperability results are less visible. Third, performance comparisons are heterogeneous. Differences in hardware, software,

network conditions, certificate chains, and connection patterns limit direct comparison of published measurements. Fourth, implementation availability does not establish universal interoperability. Fifth, the paper does not claim an exhaustive database-indexed systematic review with a verified numerical corpus. It instead provides a reproducible source-selection and classification methodology centered on authoritative standards and primary technical evidence. Sixth, PQ certificate migration remains sufficiently immature that universal claims about browser, operating-system, HSM, CA, and trust-store compatibility should be treated cautiously. Seventh, the analysis of PSKs and KeyUpdate is architectural and standards-based rather than the result of a new formal security proof. Finally, no new benchmarks, packet captures, interoperability experiments, simula- tions, datasets, or implementation measurements were conducted for this manuscript. Quantitative evidence is secondary evidence from cited sources.

12 Conclusion

Post-quantum TLS migration is not a single algorithm-replacement problem. TLS combines key establishment, authentication, resumption, traffic-key evolution, and certificate infrastructure, and each function has a different security objective and deployment dependency. The central contribution of this SoK is to make those dependencies explicit and to analyze migration through security lineage rather than through algorithm names alone. The systematization shows why confidentiality and authentication must be coordinated but separately evaluated. Hybrid ECDHE-`ML-KEM` is a strong transitional architecture for confidentiality migration when the relevant TLS integration and implementation support are available, because the hybrid construction is designed so that the resulting keying material can remain secure if one component remains secure. This conclusion does not imply that the hybrid mechanism is a finalized universal deployment standard, nor does it eliminate the need for interoperability, downgrade, and implementation analysis. The analysis of PSKs, resumption, and KeyUpdate produces a second central result. These mechanisms are not interchangeable forms of rekeying. PSKs can authenticate and can participate in key establishment; their security depends on provenance, entropy, provisioning, and lifecycle. Resumption creates new connection state that remains related to an earlier handshake. KeyUpdate derives new traffic secrets from an existing traffic-secret generation and does not introduce an independent PQ secret. Fresh PQ key establishment is therefore complementary to KeyUpdate rather than replaceable by it. Authentication migration presents a separate systems problem. `ML-DSA` and `SLH-DSA` are finalized NIST signature standards, and `ML-DSA` and `SLH-DSA` now have standardized X.509/PKIX algorithm identifiers. Nevertheless, deployment requires coordinated support across TLS signature negotiation, certificate issuance and validation, CA infrastructure, HSMs, trust anchors, clients, operating systems, and application-specific trust stores. Algorithm standardization therefore marks an enabling milestone, not the completion of migration. The resulting migration architecture is:

Inventory → HNDL assessment → hybrid/PQ key establishment → interoperability testing → staged PQ authentication → PKI and trust-anchor migration → controlled retirement of classical-only mechanisms. This sequence is not a universal timetable. The ordering should be adapted to confidentiality lifetime, endpoint control, compliance requirements, protocol support, interoperability observations, implementation maturity, and operational constraints. Long-lived sessions require an additional policy decision: when the threat model or confidentiality lifetime warrants a fresh PQ-capable exchange rather than continued symmetric key evolution. Finally, the SoK identifies a broader operational principle. Post-quantum readiness is not established when a single cryptographic primitive becomes available. It is established when the surrounding system can discover its cryptographic dependencies, negotiate compatible mechanisms, authenticate peers,

manage certificates and keys, interoperate across the actual communication path, measure the resulting costs, and replace mechanisms without redesigning the entire application. Cryptographic inventory and crypto-agility are therefore not peripheral migration practices; they are enabling conditions for sustained security as the standards and implementation ecosystem continues to evolve.

References

[AMOW24] N. Alnahawi, J. Müller, J. Oupický, and A. Wiesmaier. A comprehensive survey on post-quantum tls. *IACR Communications in Cryptology*, 1(2), 2024. doi:10.62056/AHEE0IUC.

[BCH24] Yaser Baseri, Vikas Chouhan, and Abdelhakim Hafid. Navigating quantum security risks in networked environments: A comprehensive study of quantum-safe network protocols. *Computers & Security*, 142:103883, 2024. doi:10.1016/j.cose.2024.103883.

[BFG+25] K. Bashiri, S. Fluhrer, S. Gazdag, D. Van Geest, and S. Kousidis. Internet x.509 public key infrastructure – algorithm identifiers for the stateless hash-based digital signature algorithm (slh-dsa). RFC 9909, IETF, dec 2025. Proposed Standard.

[BW22] D. Benjamin and C. A. Wood. Importing external pre-shared keys (psks) for tls 1.3. RFC 9258, IETF, jul 2022.

[C+08] D. Cooper et al. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC 5280, IETF, may 2008.

[CC26] M. Chou and P. Cao. Network impact of post-quantum certificate chain sizes on time to first byte in tls deployments. *arXiv preprint*, 2026.

[Con26] D. Connolly. Ml-kem post-quantum key agreement for tls 1.3. Internet-Draft draft-ietf-tls-mlkem-09, IETF, jul 2026. Active Internet-Draft; IESG Evaluation as of August 14, 2026.

[HSW26] T. Hollebeek, S. Schmieg, and B. E. Westerbaan. Use of ml-dsa in tls 1.3. Internet-Draft draft-ietf-tls-mldsa-05, IETF, jul 2026. work in progress.

[KKWS26] K. Kwiatkowski, P. Kampanakis, B. E. Westerbaan, and D. Stebila. Post-quantum traditional (pq/t) hybrid key agreement mechanisms for tls 1.3. RFC 10024, IETF, aug 2026. Standards Track.

[M+26] J. A. Montenegro et al. A performance evaluation framework for post-quantum tls. *Future Generation Computer Systems*, 175:108062, 2026. doi:10.1016/j.future.2025.108062.

[MKTW25] J. Massimo, P. Kampanakis, S. Turner, and B. E. Westerbaan. Internet x.509 public key infrastructure—algorithm identifiers for the module-lattice-based digital signature algorithm (ml-dsa). RFC 9881, IETF, oct 2025. doi:10.17487/RFC9881.

[Nat24a] National Institute of Standards and Technology. Module-lattice-based digital signature standard. Technical Report FIPS 204, NIST, aug 2024. doi:10.6028/NIST.FIPS.204.

[Nat24b] National Institute of Standards and Technology. Module-lattice-based key-encapsulation mechanism standard. Technical Report FIPS 203, NIST, aug 2024. doi:10.6028/NIST.FIPS.203.

[Nat24c] National Institute of Standards and Technology. Stateless hash-based digital signature standard. Technical Report FIPS 205, NIST, aug 2024. doi:10.6028/NIST.FIPS.205.

[Nat26] National Institute of Standards and Technology. Post-quantum cryptography, 2026. NIST guidance and project materials, accessed for the manuscript update through August 15, 2026.

[NHG+24] C. Nather, D. Herzinger, S.-L. Gazdag, J.-P. Steghofer, S. Daum, and D. Loebenberger. Migrating software systems toward post-quantum cryptography—a systematic literature review. *IEEE Access*, 12:132107–132126, 2024. doi:10.1109/ACCESS.2024.3450306.

[Res26] E. Rescorla. The transport layer security (tls) protocol version 1.3. RFC 9846, IETF, jul 2026.

[rus24] rustls Project. Measuring and (slightly) improving post-quantum handshake performance, dec 2024.

[SFG26] D. Stebila, S. Fluhrer, and S. Gueron. Hybrid key exchange in tls 1.3. RFC 9954, IETF, jul 2026. Informational. doi:10.17487/RFC9954.

[SWH+23] M. Sosnowski, F. Wiedner, E. Hauser, L. Steger, et al. The performance of post-quantum tls 1.3. In *Proceedings of the 19th ACM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT 2023 Companion)*, pages 19–27, dec 2023. doi:10.1145/3624354.3630585.