digicert®

Recommendations for Post-Quantum TLS Migration

Cryptographic Profile Selection for Key Exchange and Authentication

Introduction

The transition from classical to post-quantum cryptography (PQC) in TLS requires decisions in two distinct areas: key exchange (protecting session confidentiality) and authentication (verifying identity via X.509 certificates). These two areas have different deployment characteristics, different maturity of the associated standards, and different risk profiles. They should be planned and evaluated separately.

This document provides practical recommendations for organizations planning PQC migration in TLS. It is not specific to any industry vertical. The guidance reflects the current state of IETF standardization, library support, and real-world deployment as of mid-2026.

Hybrid key exchange: Use what your TLS stack provides

How key exchange works in practice

Key exchange groups are negotiated between the client and server during the TLS handshake. The groups available to an organization depend on the capabilities of its TLS stack, including libraries such as OpenSSL, BoringSSL, s2n-tls, NSS, and wolfSSL, plus operating systems, load balancers, CDNs, and other TLS termination infrastructure.

In practice, organizations upgrade or configure their TLS stack to enable a hybrid post-quantum group such as X25519MLKEM768, and TLS negotiation handles the rest. The stack determines what is possible; the organization configures what it wants to make available.

The de facto standard (X25519MLKEM768)

For TLS 1.3 key exchange, the industry has converged on X25519MLKEM768, a hybrid key agreement that combines the traditional elliptic-curve scheme X25519 with the post-quantum KEM ML-KEM-768. It is the hybrid post-quantum key agreement algorithm most organizations are likely to encounter in deployed TLS implementations today.

Key properties of X25519MLKEM768:

- **Widely deployed:** Hybrid X25519 and ML-KEM key agreement is supported by major browsers and TLS stacks. Chrome has deployed hybrid post-quantum key agreement, and Firefox's NSS library enables the standardized ML-KEM-768 and X25519 group by default. Cloudflare reports that more than two-thirds of human-generated TLS traffic to its network is already protected using hybrid ML-KEM.
- **Hybrid by design:** The hybrid construction is intended to preserve confidentiality as long as at least one of its two components remains secure. X25519 provides confidence based on a widely deployed traditional algorithm, while ML-KEM-768 provides protection against future quantum attacks. This protects negotiated connections against the risk that an attacker records encrypted traffic today and decrypts it after sufficiently capable quantum computers become available.
- **Native to current TLS stacks.** Recent TLS library releases provide the group directly. For example, OpenSSL 3.5 includes X25519MLKEM768 in its supported-groups list and offers X25519MLKEM768 and X25519 as its default TLS key shares.

Enabling hybrid key exchange typically requires an upgrade and an associated configuration change; it does not require changes to TLS certificates.

Recommendation

For key exchange in TLS 1.3, the recommendation is straightforward: **Upgrade your TLS stack and enable X25519MLKEM768 support**. There is no need to maintain a portfolio of hybrid KEM profiles or to rank alternatives.

Authentication: Choose PQC-only or composite

The real work is in certificates

While key exchange migration is conceptually simple, certificate migration is where the complexity lies. PKI certificates are longer-lived than session keys, broadly trusted, deeply embedded in infrastructure, and subject to validation by diverse relying parties. Getting the authentication side right is the harder and more consequential problem associated with quantum readiness.

A brief history of hybrid authentication

The idea of hybrid certificates—combining a classical signature algorithm with a post-quantum one—originated in a period when the PQC algorithms themselves were still candidates, not standards. When ML-DSA (then called Dilithium) was working its way through the NIST competition, there were legitimate questions about whether the final algorithms would hold up to sustained cryptanalysis. In that environment, hedging with a classical algorithm made obvious sense. If the PQC algorithm turned out to have a flaw, the classical component would keep the certificate trustworthy for the time being.

That context has changed. ML-DSA was standardized as FIPS 204 in August 2024 after years of public review. It has been subjected to extensive cryptanalysis by the global research community, and no practical attacks have emerged. The lattice problems underlying ML-DSA are well studied and understood. While no cryptographic algorithm comes with absolute guarantees, ML-DSA has gone through extensive public analysis as part of the NIST process.

The original motivation to hedge against uncertainty about PQC algorithms has diminished. What remains is a more modest hedging argument: Even standardized algorithms can occasionally have unexpected weaknesses discovered, and a hybrid construction provides defense-in-depth against that scenario.

ML-DSA-only vs. Composite: A practical comparison

Both ML-DSA-only certificates and composite ML-DSA certificates (ML-DSA + ECDSA or RSA) are viable approaches for PQC authentication. Both approaches fit within existing x.509 and TLS protocol structures because they use a single algorithm OID, a single public key field, and a single signature field. The differences are modest:

- **Performance:** Composites are slightly larger because they include both a classical and a PQ public key and signature. The increased overhead of this approach is measurable but not dramatic: roughly 100–1,000 bytes of additional key material and signature data depending on the classical component. In most deployments, this difference will not be the deciding factor.
- **Complexity:** ML-DSA-only is simpler. There is one algorithm, one key, one signature, and no question about how components interact. Composite adds a layer of construction that must be correctly implemented in every library and validated by every relying party. The LAMPS composite drafts handle this well, but simplicity has inherent value in security systems and multi-vendor landscapes.
- **Safety margin:** Composite provides defense-in-depth: If someone discovers a flaw in ML-DSA, the classical component will keep the certificate secure (and vice versa) until a cryptographically relevant quantum computer exists. This is a genuine benefit, though the probability of such a flaw has decreased with standardization and continued analysis.

Our recommendation: ML-DSA-only

ML-DSA is a NIST-standardized algorithm (FIPS 204) that has undergone years of public cryptanalysis. It is well understood, broadly implemented (OpenSSL 3.5+, BoringSSL, Java 24+), and is the algorithm that the industry is converging on for quantum-safe authentication. Google, Cloudflare, and other major platforms are deploying ML-DSA-only. The CA/Browser Forum's discussions around post-quantum public trust certificates are focused on ML-DSA, not composite. For these reasons, we recommend ML-DSA-only as the default choice for PQC certificate authentication.

ML-DSA-44 targets NIST security category 2, which is effectively equivalent to 128 bits in classical terms. For most certificates, including end-entity and short-lived intermediates, ML-DSA-44 is the right choice. From a policy standpoint, we believe ML-DSA-65 and ML-DSA-87 need only be considered for genuinely long-lived trust anchors such as root CAs and long-lived intermediate CAs.

Ultimately, the choice between ML-DSA-only and composite is a policy preference, not a critical technical decision. Organizations that prefer composite certificates can adopt them without significant penalty. The important thing is to choose one approach and move forward with deployment.

If you want hybrid certificates, use composite

Regardless of whether you choose ML-DSA-only or composite, one point is unambiguous: Avoid multi-key models. The alternatives to composite, such as Chimera, Related, and Dual certificates, each introduce more complexity.

For organizations that do choose composite, the recommended composite pairings are:

Use case	Classical component	PQ component	Rationale
End-entity / leaf certificates	ECDSA P-256 or RSA-PSS 2048	ML-DSA-65	Balanced performance; adequate for short-lived certifcates
Issuing / intermediate CAs	ECDSA P-384 or RSA-PSS 3072	ML-DSA-65	Higher assurance for medium-lived trust artifacts
Root CAs / long-lived trust anchors	ECDSA P-521 or RSA-PSS 4096	ML-DSA-87	Maximum margin for artifacts with 20+ year lifetimes

Model	Structure	Validation semantics	Key concern
Composite (LAMPS)	Single certificate, single composite key, single composite signature	Atomic: both components must verify	Larger certificate size; requires updated validation logic
Chimera (ITU-T X.509)	Single certificate, two keys, two signatures via extensions	Policy-driven: which signature to check is configurable	Policy complexity; limited TLS stack support
Related (RFC 9763)	Two separate certificates, hash-linked	No atomic requirement; relies on external policy	Weak combined assurance; dual management overhead
Dual (LAMPS drafts)	Two separate certificates, discovery-linked via SIA extension	Policy-based; relying party may ignore companion	Downgrade risk; discovery failures; revocation complexity

Composite certificates present a single algorithm OID with atomic validation semantics: both components must verify. There is no policy engine to configure, no companion certificate to discover, and no way for a relying party to silently skip one algorithm. The multi-key/multi-signature models (Chimera, Related, Dual) were reasonable explorations during the early PQC transition, but composite edges them out as the LAMPS composite drafts nears RFC publication.

Performance considerations

PQC migration will increase the size of both key exchange messages and certificate chains.

Key exchange impact

The classical X25519 key exchange is 32 bytes each for the Client and Server Hello. Extending this to include ML-KEM for hybrid key exchange adds 1184 bytes for the Client Hello and adds 1088 bytes for the Server Hello. In practice, this has minimal impact on handshake latency.

Certificate chain impact

This is the more significant concern. ML-DSA public keys and signatures are substantially larger than their classical equivalents:

Algorithm	Public key (bytes)	Signature (bytes)	Approximate total
ECDSA P-256	64	72	136
ML-DSA-44	1,312	2,420	3,732
ML-DSA-65	1,952	3,309	5,261
ML-DSA-87	2,592	4,627	7,219

A composite certificate chain (root + intermediate + leaf) will add 12–25 KB to the TLS handshake compared to an equivalent ECDSA chain.[1] For high-frequency, low-latency systems, this deserves explicit benchmarking and capacity planning.

Common pitfalls to avoid

- **Don't over-engineer key exchange profile selection.** The TLS library ecosystem has converged on a preferred approach. Spending effort ranking and debating hybrid KEM combinations is not a productive use of time.
- **Don't conflate KEM and signature roles.** ML-KEM is for key encapsulation (used in TLS key exchange). ML-DSA is for digital signatures (used in certificates). These should not be mixed: ML-KEM does not belong in a certificate's SubjectPublicKeyInfo for authentication purposes.
- **Don't deploy multi-key/multi-signature certificate models.** If you want hybrid authentication, use composite certificates; Chimera, Related, and Dual certificates add complexity and failure modes without providing security benefits. Consider whether you need hybrid at all. ML-DSA-only is simpler and the algorithms are well vetted.
- **Don't let the hybrid debate delay your migration.** The choice between ML-DSA and composite is a policy preference, not a technical blocker. Both work. Pick one and start deploying. The worst outcome is spending 2026–2028 debating certificate models instead of upgrading infrastructure.
- **Don't ignore performance.** PQ signature sizes are large. A migration plan that does not include handshake size analysis, MTU considerations, and latency benchmarking is incomplete.

Recommended migration approach

A phased migration should be organized around concrete milestones.

Phase 1: Hybrid key exchange (now)

This is the lowest-effort, highest-impact step. Upgrade to TLS 1.3 and enable x25519mlkem768 key exchange. This protects current session traffic against future quantum decryption of recorded ciphertexts ("harvest now, decrypt later"). No certificate changes are required.

Organizations using legacy or vendor-managed TLS stacks should engage their vendor now to understand how or when they will support ML-KEM.

Phase 2: PQC certificate testing (now for Private CA)

ML-DSA certificate issuance is available today through internal PKI offerings, including DigiCert Private CA, which supports ML-DSA-44, ML-DSA-65, and ML-DSA-87. Organizations should begin testing ML-DSA certificate issuance and validation now in internal PKI environments. For most organizations, ML-DSA-only certificates (ML-DSA-44 for leaf, ML-DSA-65 for long-lived intermediates) will be the simplest path. Organizations that require composite certificates can test those in parallel.

Public trust ML-DSA certificates are gated by CA/Browser Forum acceptance, which is an ongoing process (see Section 7.1 on current Baseline Requirements constraints).

Phase 3: Production PQC certificate deployment (when ecosystem is ready)

Deploy PQC certificates in production, starting with leaf certificates and progressing to intermediate and root CAs. Root CA migration will be the longest-lead-time item due to the ceremony processes, cross-certification requirements, and trust store inclusion timelines involved.

Beyond TLS: Other protocol areas

TLS is the most widely deployed use of asymmetric cryptography and the most immediately actionable quantum readiness use case given the broad support for hybrid key exchange. However, non-TLS uses of asymmetric cryptography that require long-term assurances of integrity and stored encrypted data and are directly exposed to "harvest now, decrypt later" attacks (like email confidentiality, electronic signatures, and code signing) need to be on an organization's risk "radar." A detailed treatment of non-TLS PQC migration is outside the scope of this document, but organizations should not assume that completing TLS migration means the job is done.

Important non-TLS use cases to plan for include:

- **S/MIME and CMS:** Signed and encrypted email and document workflows. The LAMPS composite drafts cover CMS usage. Protected archived records are particularly sensitive to "harvest now, decrypt later" attacks.
- **SSH and SFTP:** Administrative access and file transfer. OpenSSH has PQ key exchange support. This should be prioritized for infrastructure management channels.
- **IPsec / IKEv2:** VPN and network-layer encryption. Hybrid KEM support in IKEv2 is less mature but is an active area of IETF work.
- **JOSE / COSE**: API token security (JWTs, CWTs). This is the least mature area and will likely be the last to see standardized PQ support.

A note on the Web PKI

As of this writing, the CA/Browser Forum TLS Baseline Requirements do not permit post-quantum algorithms in publicly trusted certificates, regardless of whether the certificate uses ML-DSA-only, composite, or any other approach.

Separately, Merkle Tree Certificates (MTC) have been proposed as an alternative certificate architecture for the public Web PKI that could reduce the impact of post-quantum signatures. MTC is not yet generally available, and final browser/root program requirements have not been published. However, Chrome has identified MTC as its preferred direction and is already conducting experiments with real internet traffic.

About DigiCert

DigiCert is a global leader in intelligent trust, securing people, data, and devices with AI-powered solutions built to stop threats today and prepare for a quantum-safe future.

Continue your PQC journey at digicert.com/solutions/post-quantum-computing

References

NIST FIPS 203: ML-KEM (Module-Lattice-Based Key-Encapsulation Mechanism)

NIST FIPS 204: ML-DSA (Module-Lattice-Based Digital Signature Algorithm)

Use of ML-DSA in TLS 1.3 (draft-ietf-tls-ecdhe-mlkem-05)

Composite ML-DSA for X.509 PKI (draft-ietf-lamps-pq-composite-sigs)

RFC 9794: Terminology for Post-Quantum Traditional Hybrid Schemes

RFC 9763: Related Certificates for Use in Multiple Authentications

RFC 9881: Internet X.509 Public Key Infrastructure

RFC 9965: Hybrid Key Exchange in TLS 1.3