WP-029
June 2026
12 sections

POST-QUANTUM CRYPTOGRAPHY

TLS 1.3 Hybrid Post-Quantum Deployment: A Production Guide for 2026

X25519MLKEM768 is already active on 30-50% of TLS 1.3 handshakes. Here is what your team needs to know to deploy it, configure it, and prove you have done it.

AffixIO Research | June 2026 | Download PDF

ABSTRACT

TLS 1.3 is the foundation of internet security, protecting everything from browser sessions to microservice APIs. Cryptanalytically relevant quantum computers threaten the classical key exchange algorithms at the core of TLS, and the "harvest now, decrypt later" risk means organisations cannot wait for quantum computers to arrive before acting. The good news is that the industry has already moved: hybrid key exchange combining classical X25519 with post-quantum ML-KEM-768 (the X25519MLKEM768 group defined in RFC 9598) is now the default in Chrome, Firefox, Edge, and Safari, and is enabled on Cloudflare, AWS, and Google production infrastructure. Between 30 and 50 per cent of all TLS 1.3 handshakes in 2026 already use hybrid PQC.

> This paper covers the technical mechanics of hybrid TLS, the browser and CDN deployment landscape, the OpenSSL 3.5 rollout path for server operators, the performance characteristics of hybrid handshakes, the policy decisions organisations now need to make, and where post-quantum certificates fit into the picture. AffixIO's post-quantum attestation infrastructure (ML-DSA-65, described in WP-002) provides the audit trail layer for organisations that need to document their TLS migration for regulatory purposes.

CONTENTS

1 TLS Meets the Quantum Transition
2 Harvest Now, Decrypt Later
3 ML-KEM: The Post-Quantum Component
4 Hybrid Key Exchange in TLS 1.3
5 RFC 9598 and the Hybrid Group Registry
6 Browser and Client Support in 2026
7 CDN and Server-Side Deployment
8 OpenSSL 3.5: The Rollout Path
9 Performance: What Hybrid Costs
10 Certificate PQC: The Next Frontier
11 Policy: Mandating, Auditing, Transitioning
12 Attestation and Audit for PQC Migration

SECTION 01

TLS Meets the Quantum Transition

Transport Layer Security is the most widely deployed cryptographic protocol on earth. Every HTTPS request, every API call between services, every VPN session, every email in transit relies on TLS to establish a secure channel. The version in near-universal use today, TLS 1.3 (RFC 8446, 2018), is technically

excellent: it removed a decade of legacy cruft, reduced handshake round-trips to one, and mandated forward secrecy. In most respects it is the best version of TLS ever standardised.

The problem is what is inside it. TLS 1.3's default key exchange relies on elliptic curve Diffie-Hellman, most commonly X25519. The security of X25519 depends on the hardness of the discrete logarithm problem on the Curve25519 elliptic curve. That problem is classically hard. It is not, however, quantum hard. Shor's algorithm, running on a sufficiently large quantum computer, can solve it efficiently. This means that a quantum computer capable of attacking 256-bit elliptic curves would be able to break every classical ECDHE key exchange in seconds.

No such quantum computer exists today. The largest publicly known quantum computers in 2026 are still well short of the fault-tolerant qubit counts required to run Shor's algorithm at cryptographically relevant scale. But this is not a reason to wait. The window between "quantum computers capable of breaking TLS" and "quantum computers arriving" may be shorter than the window between "start deploying PQC" and "finish deploying PQC across your infrastructure." Organisations with complex estates took seven years to fully deploy TLS 1.3 after RFC 8446 was published; the first generation of large-scale fault-tolerant quantum computers may arrive well within that planning horizon.

More pressingly, there is a threat that does not require quantum computers to exist today: the harvest-now-decrypt-later attack, which is discussed in the next section. This threat means the time to deploy post-quantum TLS is not when quantum computers arrive, but now. The practical barrier to doing so has fallen dramatically in 2025 and 2026: NIST finalised FIPS 203 (ML-KEM) in August 2024, RFC 9598 registered the hybrid TLS groups in 2024, Chrome and Firefox enabled hybrid PQC by default in mid-2024, and OpenSSL 3.5 shipped ML-KEM support in April 2025. The hybrid TLS ecosystem is mature enough for production deployment today.

This paper does not argue that quantum computers are imminent or that classical TLS is immediately broken. It argues that hybrid TLS (classical plus post-quantum in a single key exchange) is available, tested, interoperable, and

carries no security downside. The question is not whether to deploy it, but how.

SECTION 02

Harvest Now, Decrypt Later

The harvest-now-decrypt-later (HNDL) attack is simple in concept. An adversary records encrypted TLS traffic today, stores it, and waits. When a cryptanalytically relevant quantum computer becomes available, the adversary uses it to recover the session keys from the recorded handshakes and decrypts all of the stored traffic. The attack requires no active interference with the TLS session: passive collection of encrypted packets off a network tap is sufficient.

This attack is feasible because TLS 1.3's forward secrecy protects against a different threat: compromise of the server's long-term private key in the future. Forward secrecy means that if an attacker steals your server certificate's private key tomorrow, they cannot decrypt sessions recorded yesterday, because each session uses an ephemeral key derived from the ECDHE key exchange. Forward secrecy is a genuine security improvement. But it does not protect against HNDL, because the attack targets the ephemeral ECDHE key exchange itself rather than the server's certificate key.

Which data is actually at risk from HNDL? Any TLS session whose confidentiality needs to be maintained for longer than the expected timeline for quantum computers to arrive at cryptographic scale. That includes: classified government communications (which is why the US NSA's CNSA 2.0 mandate targets 2030 for national security systems); medical records subject to long-term retention obligations; financial transaction data subject to audit retention; intellectual property with long competitive lifetimes; and communications between individuals in high-risk contexts. For organisations in heavily regulated sectors, the relevant question is not "how long before quantum computers arrive" but "how long are we required to keep this data confidential."

The practical implication is that HNDL creates an asymmetry: the adversary records traffic now, at minimal cost, and waits to decrypt it later when the capability exists. The organisation has no indication that collection is occurring. The only defence is to ensure that the session keys cannot be recovered even by a quantum computer, which requires using a post-quantum key encapsulation mechanism in the TLS handshake. Hybrid TLS with ML-KEM-768 achieves exactly this: a recorded hybrid handshake cannot be broken by a quantum computer running Shor's algorithm, because ML-KEM's security does not depend on the problems Shor's algorithm attacks.

It is also worth noting what HNDL does not threaten: data whose confidentiality requirement ends before quantum computers arrive does not need post-quantum protection now. A TLS session carrying a publicly available web page carries no confidential data worth protecting for years. The risk calculation is proportional to the sensitivity and longevity of the data in transit.

SECTION 03

ML-KEM: The Post-Quantum Component

ML-KEM (Module-Lattice-based Key Encapsulation Mechanism) is NIST's primary post-quantum key establishment standard, published as FIPS 203 in August 2024. It is derived from the CRYSTALS-Kyber algorithm, which was selected by NIST in 2022 after a six-year competition evaluating post-quantum cryptographic algorithms. ML-KEM is a Key Encapsulation Mechanism (KEM): it allows one party to encapsulate a random shared secret under the other party's public key, producing a ciphertext. The holder of the corresponding private key can decapsulate the ciphertext to recover the shared secret. This is the building block for key exchange.

ML-KEM comes in three parameter sets. ML-KEM-512 offers security roughly equivalent to AES-128 (NIST security level 1). ML-KEM-768 offers security equivalent to AES-192 (level 3). ML-KEM-1024 offers security equivalent to AES-256 (level 5). For TLS deployment in 2026, ML-KEM-768 is the standard choice. It provides a comfortable security margin above classical AES-128 equivalence, and its key and ciphertext sizes are workable for TLS: the ML-

KEM-768 public key is 1,184 bytes and the ciphertext is 1,088 bytes, compared to 32 bytes for an X25519 public key. This size difference is what drives the bandwidth overhead discussed in Section 9.

The security of ML-KEM rests on the hardness of the Module Learning With Errors (MLWE) problem, a variant of the Learning With Errors problem over module lattices. There is no known efficient quantum algorithm for solving MLWE. Shor's algorithm does not apply; Grover's algorithm provides only a quadratic speedup (which the parameter sizes already account for). The academic cryptographic community has analysed ML-KEM's security extensively over the course of NIST's competition, and no significant weaknesses have been found. This does not mean ML-KEM is unconditionally secure; it means it has survived years of intense scrutiny by a large community with strong incentives to find flaws.

One important property of KEMs versus Diffie-Hellman is that KEMs are not inherently interactive: one party encapsulates and sends the ciphertext; the other decapsulates. This maps cleanly onto the TLS 1.3 key exchange model, where the client sends a key share and the server responds. The integration of ML-KEM into TLS 1.3 is therefore relatively straightforward from a protocol design perspective, which is one reason the deployment has moved quickly.

SECTION 04

Hybrid Key Exchange in TLS 1.3

TLS 1.3's key exchange mechanism is built around the `key_share` extension in the ClientHello message. The client proposes one or more named groups (such as `x25519` or `secp256r1`), each accompanied by a key share for that group. The server selects one group, provides its own key share for that group, and both parties derive the session key using the group's key agreement algorithm.

The hybrid approach extends this mechanism by defining new named groups that combine a classical algorithm with a post-quantum algorithm. The key share for a hybrid group contains two concatenated components: the classical key share (for example, an X25519 public key) and the post-quantum

component (for example, an ML-KEM-768 public key or ciphertext). Both components are processed independently, producing two shared secrets. These are then combined using a defined combiner construction (concatenation followed by a key derivation function) to produce the final session key material fed into TLS 1.3's key schedule.

The critical security property of this hybrid construction is that it provides security if either component is secure. If ML-KEM-768 has an undiscovered weakness (classical or quantum), the X25519 component still provides classical security. If X25519 is broken by a future quantum computer, the ML-KEM-768 component still provides post-quantum security. The construction is conservative: it does not assume ML-KEM is perfect, and it does not abandon proven classical security. This "belt and braces" property is why hybrid deployment is widely recommended as the appropriate transition strategy, rather than jumping directly to a post-quantum-only mode.

The specific hybrid group in widest deployment is `X25519MLKEM768`, which combines X25519 ECDHE (32-byte key share) with ML-KEM-768 (1,184-byte public key in ClientHello; 1,088-byte ciphertext in ServerHello). A higher-security variant, `X448MLKEM1280`, combines X448 with ML-KEM-1024 for environments requiring NIST level 5 security equivalence. Most production deployments use X25519MLKEM768, which hits the sweet spot of practical bandwidth overhead and strong security margin.

When a client supporting hybrid TLS connects to a server that does not, TLS negotiation falls back to the client's next preferred classical group. This graceful fallback means hybrid TLS can be deployed on servers without breaking compatibility with legacy clients. Conversely, clients supporting hybrid TLS that connect to servers offering hybrid groups will prefer the hybrid path. The result is that enabling hybrid TLS on a server immediately provides quantum-resistant key exchange for any connecting client that also supports it, with no impact on clients that do not.

SECTION 05

RFC 9598 and the Hybrid Group Registry

RFC 9598 ("Hybrid key exchange in TLS 1.3") was published in 2024 by the IETF TLS working group and formalises the mechanism for combining classical and post-quantum key shares in TLS 1.3. It defines the construction for hybrid key shares (concatenation of the two component key shares), the combiner for deriving the final shared secret (concatenation of the two component shared secrets, fed into the TLS 1.3 key derivation), and the IANA-registered code points for specific hybrid groups.

The IANA-registered hybrid group code points most relevant to production deployment are:

CODE POINT	GROUP NAME	CLASSICAL COMPONENT	PQ COMPONENT	SECURITY LEVEL
`0x11EC`	X25519MLKEM768	X25519 (ECDH)	ML-KEM-768 (FIPS 203)	Level 3 (~AES-192)
`0x11EB`	X448MLKEM1280	X448 (ECDH)	ML-KEM-1024 (FIPS 203)	Level 5 (~AES-256)
`0x6399`	SecP256r1MLKEM768	P-256 (ECDH)	ML-KEM-768 (FIPS 203)	Level 3

The X25519MLKEM768 group (code point `0x11EC`) is the production standard. It is the group enabled by default in Chrome, Firefox, Cloudflare, and OpenSSL 3.5. The SecP256r1MLKEM768 variant is provided for environments where X25519 is not acceptable (for example, FIPS-validated environments that require NIST P-curves rather than Bernstein curves); it uses the same ML-KEM-768 post-quantum component but pairs it with P-256 rather than X25519.

From a TLS negotiation perspective, hybrid groups behave identically to classical groups: they appear in the `supported_groups` extension in ClientHello and in the `key_share` extension. A server that sees `X25519MLKEM768` in the client's supported groups list and in the key share will respond with its own X25519MLKEM768 key share. A server that does not

recognise the group will skip it and negotiate a classical group instead. The protocol requires no new message types or extension types beyond those already in TLS 1.3.

The IETF TLS working group also has active work on related areas: the TLS Certificate Compression working group is examining how to reduce the overhead of post-quantum certificates (which will be significantly larger than classical certificates), and the post-quantum signatures in TLS working group is examining how to integrate ML-DSA certificates into the certificate handshake. These are still in progress as of mid-2026.

SECTION 06

Browser and Client Support in 2026

The browser ecosystem moved quickly after NIST published FIPS 203 in August 2024. The following table shows the current hybrid TLS support status across major clients as of mid-2026:

CLIENT	VERSION ADDED	DEFAULT ENABLED	GROUP	NOTES
Chrome / Chromium	124 (May 2024)	Yes	X25519MLKEM768	Replaced previ X25519Kyber76 Chrome 131
Firefox	128 (July 2024)	Yes	X25519MLKEM768	Enabled via network.securit pref, on by defa
Microsoft Edge	124 (Chromium-based)	Yes	X25519MLKEM768	Follows Chromi
Safari / WebKit	Safari 18 (Sep 2024)	Yes	X25519MLKEM768	Added in WebK Sequoia and iO
Java JDK 27	JDK 27 (Feb 2026)	No (preview)	X25519MLKEM768	JEP 496; enable Dcom.sun.net.s
Go (stdlib)	Go 1.23 (Aug 2024)	Yes (GODEBUG)	X25519MLKEM768	Enabled via GODEBUG=tlsk default-on in 1.2
OpenSSL	3.5 (Apr 2025)	Config required	X25519MLKEM768, X448MLKEM1280	See Section 8 f
BoringSSL	2024	Yes	X25519MLKEM768	Used by Chrom Google's produ
curl	8.9.0 (Jul 2024)	Depends on TLS backend	X25519MLKEM768	Via OpenSSL 3. backend

The practical implication of near-universal browser default enablement is that any server accepting HTTPS connections from modern browsers is already negotiating X25519MLKEM768 for a large proportion of its sessions, provided the server also supports it. For servers sitting behind a CDN such as Cloudflare (see Section 7), the CDN terminates the hybrid TLS session with the browser, so the client-to-CDN connection is already quantum-resistant regardless of the origin server's configuration. The origin-to-CDN connection, and any service-to-service connections, require separate attention.

Cloudflare Radar telemetry reported in early 2026 that between 30 and 50 per cent of TLS 1.3 handshakes observed on its network are using a post-quantum key exchange, with X25519MLKEM768 accounting for the vast majority of those. This figure is expected to increase as older browser versions cycle out and Java 27 and later Go versions default-enable hybrid PQC. By late 2026, industry analysts expect the majority of browser TLS sessions to use hybrid key exchange.

The Java ecosystem deserves particular attention. Enterprise Java applications are widely deployed in financial services, healthcare, and government sectors, and Java's TLS library (JSSE) is what they use. Java JDK 27's addition of hybrid PQC support (JEP 496, February 2026) is significant because it brings quantum-resistant TLS within reach for these sectors without requiring them to replace their TLS library. The default-off status in Java 27 reflects appropriate caution for enterprise deployments; the expectation is that it will become default-on in a subsequent release after production experience accumulates.

SECTION 07

CDN and Server-Side Deployment Status

The major CDN and cloud providers have been among the fastest movers on hybrid TLS, and their deployments are significant because they protect enormous volumes of internet traffic.

Cloudflare

Cloudflare enabled X25519MLKEM768 on all customer traffic in 2024, making it the first large-scale production deployment of hybrid post-quantum TLS. Any website or API served through Cloudflare's network automatically benefits from quantum-resistant key exchange for clients that support it. Cloudflare has published extensive telemetry data on hybrid TLS adoption through Cloudflare Radar, which is a useful public resource for tracking industry-wide adoption rates.

AWS

AWS enabled post-quantum hybrid TLS (PQC-preferred in TLS security policies) on Application Load Balancers (ALB) and API Gateway in 2025. The TLS security policy `ELBSecurityPolicy-TLS13-1-3-2021-06` and newer policies support X25519MLKEM768 for TLS 1.3 connections. AWS also enabled hybrid PQC on CloudFront distributions in late 2024. For organisations with AWS infrastructure, enabling hybrid TLS on load balancers and CDN distributions requires selecting an appropriate TLS security policy; no code changes are needed.

Google

Google has used BoringSSL (its internal TLS library) to run experimental and then production hybrid PQC across Google services since 2022. Google's production services including Google.com, Google APIs, and Google Cloud Platform infrastructure have supported X25519MLKEM768 since 2024. Google publishes periodic reports on PQC adoption in connection with its security engineering work.

Fastly

Fastly supports X25519MLKEM768 on its CDN platform, including TLS termination for edge connections. Fastly customers can enable hybrid PQC through the TLS configuration in the Fastly console. Fastly has published technical documentation on its hybrid TLS implementation.

The Origin Gap

An important nuance for organisations behind a CDN is that CDN-terminated TLS protects the client-to-CDN segment of the connection. The CDN-to-origin segment is a separate TLS connection, and its security depends on the origin server's configuration. If the origin server uses a classical-only TLS stack, the CDN-to-origin traffic is not protected against HNDL, even if the client-to-CDN segment uses hybrid TLS. Organisations that use CDNs should audit the CDN-to-origin TLS configuration separately, particularly for high-sensitivity data paths.

The same gap applies to service-to-service connections within a microservice architecture, internal API calls, database connections, and any other encrypted channel that does not pass through a CDN. These connections are often overlooked in TLS migration planning but represent a significant proportion of enterprise encrypted traffic volume.

SECTION 08

OpenSSL 3.5: The Rollout Path

OpenSSL 3.5, released in April 2025, added native support for X25519MLKEM768 and X448MLKEM1280 as supported groups in TLS 1.3. This is significant because OpenSSL is the TLS library underlying the vast majority of Linux-based web servers, including nginx, Apache httpd, and most API gateway software. For organisations running these stacks, the path to hybrid TLS is a combination of package upgrade and configuration change, with no application code modifications required.

Prerequisites

OpenSSL 3.5 or later is required. Many Linux distributions shipped OpenSSL 3.3 or 3.4 as their default in early 2026. Operators should verify their OpenSSL version with `openssl version` and upgrade via their package manager or from source if necessary. Ubuntu 24.04 LTS ships OpenSSL 3.0; OpenSSL 3.5 is available as a PPA or can be compiled from source. Debian 13 (Trixie) includes OpenSSL 3.5. RHEL and its derivatives are typically more conservative; check your distribution's advisory channels.

nginx Configuration

To enable X25519MLKEM768 in nginx alongside classical groups (the recommended hybrid approach):

```
ssl_protocols TLSv1.3;
ssl_ecdh_curve X25519MLKEM768:x25519:secp256r1;
ssl_prefer_server_ciphers off;
```

The `ssl_ecdh_curve` directive controls the list of supported groups in preference order. Placing `X25519MLKEM768` first ensures it is preferred when the client supports it, while `x25519` and `secp256r1` provide fallback for clients that do not. The `ssl_prefer_server_ciphers off` directive is appropriate for TLS 1.3, where cipher negotiation differs from TLS 1.2.

Apache httpd Configuration

```
SSLProtocol -all +TLSv1.3
SSLOpenSSLConfCmd Groups "X25519MLKEM768:x25519:secp256r1"
```

Apache requires mod_ssl with OpenSSL 3.5 support. The `SSLOpenSSLConfCmd Groups` directive passes the named group list directly to OpenSSL.

Verification

To verify that a server is offering X25519MLKEM768:

```
openssl s_client -connect example.com:443 \
  -groups X25519MLKEM768 \
  -tls1_3 2>&1 | grep "Server Temp Key"
```

A successful hybrid negotiation will show output similar to `Server Temp Key: X25519MLKEM768, 256 bits`. Alternatively, Cloudflare's online TLS testing tool and SSL Labs both report the negotiated key exchange group and will indicate whether a hybrid group was used.

Migration Checklist

- Upgrade OpenSSL to 3.5+ on all TLS-terminating hosts
- Add `X25519MLKEM768` as the first entry in the named groups configuration
- Retain classical groups (`x25519`, `secp256r1`) for compatibility with legacy clients
- Test with a hybrid-capable client (Chrome, Firefox, or `openssl s_client -groups X25519MLKEM768`)

- Monitor server resource usage for one week post-deployment; the overhead should be negligible (see Section 9)
- Audit CDN-to-origin and service-to-service connections separately
- Document the migration for compliance and audit purposes

SECTION 09

Performance: What Hybrid Handshakes Actually Cost

The performance characteristics of hybrid TLS handshakes are a frequent concern for operators evaluating deployment. The headline answer is that the overhead is real but small, is bandwidth-dominated rather than CPU-dominated, and is negligible for the vast majority of production workloads.

Bandwidth Overhead

The largest contributor to hybrid handshake overhead is the size of the ML-KEM-768 key material. In a classical TLS 1.3 handshake using X25519, the client's key share is 32 bytes and the server's key share is also 32 bytes. In an X25519MLKEM768 handshake, the client sends a combined key share of 32 bytes (X25519) plus 1,184 bytes (ML-KEM-768 public key), totalling 1,216 bytes. The server's response includes 32 bytes (X25519 share) plus 1,088 bytes (ML-KEM-768 ciphertext), totalling 1,120 bytes. The net additional bandwidth is approximately 1.1–1.2 KB per TLS handshake compared to classical X25519.

COMPONENT	CLASSICAL X25519	X25519MLKEM768 HYBRID	OVERHEAD
Client key share (ClientHello)	32 bytes	1,216 bytes	+1,184 bytes
Server key share (ServerHello)	32 bytes	1,120 bytes	+1,088 bytes
Total key exchange overhead	64 bytes	2,336 bytes	+2,272 bytes (~2.2 KB)
Net impact on session	Baseline	+~1.1 KB (accounting for TLS record framing)	Negligible for most sessions

For a typical HTTPS session transferring several kilobytes to megabytes of application data, an additional 1.1 KB in the handshake is well under one per cent of total session bandwidth. For very short-lived sessions (for example, health check requests or small API calls), the proportional overhead is higher but the absolute overhead per connection remains under 2 KB.

CPU Overhead

ML-KEM-768's encapsulation and decapsulation operations are fast on modern hardware. Benchmarks on a 2024-generation x86-64 server show approximately 60,000 ML-KEM-768 encapsulations per second and 80,000 decapsulations per second per core, which is faster than RSA-2048 operations and comparable to X25519. At the TLS layer, ML-KEM adds microseconds to handshake processing time, not milliseconds. CPU is not the bottleneck.

Latency

The bandwidth increase in the ClientHello and ServerHello messages adds one additional network round-trip's worth of data, which may cause the ClientHello to no longer fit in a single TCP segment on networks with small MTUs. This can add a small amount of latency, typically under 2 milliseconds on a local network and under 5 milliseconds on a high-latency path. For

interactive applications, this is imperceptible. For extremely latency-sensitive systems such as high-frequency trading, it warrants measurement but is unlikely to be the dominant latency factor.

Summary

The performance cost of hybrid TLS is dominated by approximately 1.1 KB of additional handshake bandwidth. For any workload where TLS handshake overhead was not previously a performance concern, it will not become one with hybrid TLS. Organisations with specific performance constraints should benchmark their workloads, but the published telemetry from Cloudflare, Google, and AWS indicates that hybrid TLS is deployable on production infrastructure without performance regression for typical use cases.

SECTION 10

Certificate Post-Quantum: The Next Frontier

The hybrid TLS deployments described in this paper protect the key exchange phase of the TLS handshake. This is the most urgent problem to solve for HNDL resistance, and it is largely solved for browser TLS in 2026. However, it leaves the certificate authentication phase of TLS still reliant on classical cryptography.

When a TLS server presents its certificate, it proves ownership of the corresponding private key by producing a signature during the handshake. The signature algorithm is determined by the certificate: most certificates today use ECDSA with P-256 or P-384, or RSA with 2048-4096 bit keys. Both ECDSA and RSA are vulnerable to Shor's algorithm. A quantum adversary that captures a TLS handshake could, in principle, forge certificates once a large enough quantum computer exists, breaking the server authentication guarantee.

Why are post-quantum certificates lagging behind hybrid key exchange? The key exchange change is a configuration change on servers and clients, requiring no changes to certificate infrastructure. Post-quantum certificates, by contrast, require changes to the entire certificate issuance chain:

Certificate Authorities would need to issue ML-DSA certificates; browsers would need to trust ML-DSA roots; certificate validation code in every TLS library would need to handle ML-DSA signatures; and the existing WebPKI infrastructure would need coordinated updates. This is a much larger coordination problem than enabling a new key exchange group.

NIST FIPS 204 (ML-DSA, the post-quantum signature standard) was published alongside FIPS 203 in August 2024, so the algorithm is standardised. The IETF is actively developing the profile for ML-DSA in X.509 certificates (RFC draft: draft-ietf-lamps-dilithium-certificates). PKIX hybrid certificate formats (combining classical and ML-DSA signatures in a single certificate) are under development. The expectation in the standards community is that hybrid certificates will be available in production certificate authorities by 2027-2028, with browser root store trust following shortly thereafter.

In the meantime, the practical recommendation is to ensure certificate key types are as short-lived as possible and rotate them regularly. Certificates with 90-day validity (as recommended by Let's Encrypt and increasingly mandated by browser policies) limit the window during which a certificate key compromise would be exploitable. Certificates used in high-sensitivity non-TLS contexts (code signing, document signing, long-lived identity assertions) should be evaluated for earlier post-quantum migration using ML-DSA where library support exists.

AffixIO's post-quantum attestation infrastructure uses ML-DSA-65 (the FIPS 204 equivalent of ML-DSA at security level 3) for signing Merkle tree audit roots today. This is a direct parallel to the TLS certificate signing problem: both involve using a digital signature to attest to an identity or record at a point in time. The infrastructure and libraries exist; the coordination required for WebPKI adoption is what the industry is now working through.

SECTION 11

Policy: Mandating, Auditing, and Transitioning

The technical case for hybrid TLS is straightforward. The policy questions are more nuanced: what should organisations mandate, over what timeline, and how should they document their transitions for regulatory purposes?

Relevant Standards and Mandates

Several authoritative bodies have issued guidance on post-quantum TLS migration. The following are the most relevant for organisations in scope:

AUTHORITY	DOCUMENT	KEY REQUIREMENT
US NSA	CNSA 2.0 (2022)	National Security Systems must use PQC algorithms (including PQC key exchange) by 2030; start now for new systems
NIST	SP 1800-38 (2024)	Migration to Post-Quantum Cryptography: practical guidance including TLS migration steps
NIST	IR 8413 (2022, updated 2024)	Status report on the NIST PQC standardisation process; background for understanding ML-KEM selection
German BSI	TR-02102-2 (2024)	Recommends hybrid TLS with X25519MLKEM768 or SecP256r1MLKEM768 for TLS 1.3 deployments now
UK NCSC	Post-Quantum Cryptography guidance (2023, updated 2025)	Recommends hybrid key exchange for sensitive data now; expects post-quantum as primary by 2028
ETSI	TR 103 619 (2022)	Quantum-safe cryptography for TLS: technical report covering hybrid approaches and migration
EU	NIS2 + ENISA PQC guidance (2024)	Recommends crypto-agility and hybrid TLS for critical infrastructure operators

When to Mandate Hybrid TLS

For new deployments on OpenSSL 3.5+ or equivalent, there is no reason to use classical-only TLS 1.3. Hybrid TLS should be the default for all new server deployments. For existing deployments, a phased migration over 12-18 months is appropriate for most organisations, starting with internet-facing servers that are most exposed to passive traffic collection.

When to Disable Classical TLS Groups

Disabling classical fallback groups (X25519, P-256) is not recommended yet. A significant proportion of clients, particularly in IoT, embedded systems, and enterprise Java environments, do not yet support hybrid groups. Removing classical fallback would break connectivity for these clients. The appropriate time to consider disabling classical groups is when client telemetry shows that classical-only clients represent a negligible fraction of connections, which is unlikely before 2028 for most environments.

Crypto Agility

The TLS migration to hybrid PQC is an opportunity to implement crypto agility at the configuration level: ensure that TLS group preferences are configurable without code changes, so that when post-quantum groups need to be updated (for example, if a weakness is discovered in ML-KEM-768) the response can be a configuration change rather than an emergency code deployment. Externalising TLS configuration from application code is good practice regardless of post-quantum considerations.

Audit Inventory

Organisations subject to CNSA 2.0, NIS2, or sector-specific PQC mandates will need to maintain an inventory of TLS endpoints, their supported cipher suites and groups, their certificate algorithm types, and the timeline of their migration to hybrid TLS. This inventory should be treated as a compliance record and maintained with the same rigour as other security documentation. Automated scanning tools can assist with endpoint discovery, but the audit record generation and tamper-resistant storage is a separate concern, discussed in the next section.

SECTION 12

Attestation and Audit for PQC Migration

Hybrid TLS deployment is both a technical and a compliance exercise. Organisations subject to CNSA 2.0, NIS2, DORA, or sector-specific cybersecurity requirements need to demonstrate, to auditors and regulators, that their PQC migration is progressing on schedule and that specific endpoints have been migrated at specific times. A configuration file or a narrative report is not a tamper-resistant compliance record.

This is where post-quantum attestation infrastructure becomes relevant outside of the TLS protocol itself. AffixIO's attestation layer generates Merkle-tree-anchored audit records signed with ML-DSA-65 (NIST FIPS 204). Applied to TLS migration compliance, this means: for each endpoint that has been configured for hybrid TLS, a governance record can be generated containing the endpoint identifier, the supported TLS groups (including X25519MLKEM768), the certificate details, the configuration timestamp, and a cryptographic hash of the configuration. This record is anchored in a Merkle tree whose root is signed with ML-DSA-65.

The result is a compliance record with two important properties. First, it is tamper-evident: any attempt to backdate or modify the record is detectable by verifying the Merkle proof and the ML-DSA-65 signature. Second, it is itself post-quantum secure: the signature on the compliance record cannot be forged by a future quantum computer, ensuring that the compliance evidence remains trustworthy for the duration of the regulatory retention period.

This is a direct parallel to the certificate post-quantum problem: just as TLS certificates today use classical signatures that may need to be replaced with ML-DSA certificates in the future, compliance records today signed with classical algorithms create a future forgery risk. Using ML-DSA-65 for compliance record signing now removes that risk, consistent with the same reasoning that motivates hybrid TLS key exchange.

The practical integration pattern is straightforward. An organisation's TLS configuration management pipeline generates a structured record of each endpoint's TLS configuration after each deployment. This record is submitted to AffixIO's attestation API, which anchors it in the audit Merkle tree and

returns a proof reference. The proof reference is stored alongside the configuration record in the organisation's compliance management system. When an auditor requests evidence of hybrid TLS deployment, the organisation provides the configuration record and the proof reference; the auditor verifies the proof against AffixIO's public Merkle root.

The complementary use case is certificate lifecycle attestation. When post-quantum certificates become available (expected 2027-2028), the same attestation infrastructure can record certificate issuance, renewal, and revocation events with tamper-resistant timestamps. This supports the audit trail requirements for certificate management under NIS2 Article 21 and similar frameworks, extending the same post-quantum audit infrastructure described in WP-002 (Post-Quantum Attestation) and WP-013 (Post-Quantum PKI Migration) to the TLS domain.

The broader point is that post-quantum migration is not a single event but an ongoing operational programme, spanning several years and multiple protocol layers (key exchange today, certificates in 2027-2028, other cryptographic primitives thereafter). Compliance frameworks will increasingly require evidence of systematic progress rather than point-in-time snapshots. The organisations best positioned for that scrutiny are those that have built audit infrastructure capable of generating tamper-resistant, independently verifiable records of each migration step as it occurs.

> Related AffixIO whitepapers: WP-002: Post-Quantum Attestation covers ML-DSA-65 signing and Merkle audit infrastructure. WP-013: Post-Quantum PKI Migration covers the certificate infrastructure transition in depth. WP-020: DORA and MiCA AI Compliance covers audit evidence requirements for financial sector PQC obligations.

FREQUENTLY ASKED

Common Questions

Is X25519MLKEM768 safe to deploy on production servers today?

Yes. X25519MLKEM768 is a hybrid group combining classical X25519 ECDHE with ML-KEM-768. The hybrid design means the connection is secure as long as either component is unbroken. If ML-KEM-768 has an undiscovered weakness, X25519 still protects the session. If X25519 is broken by a quantum computer, ML-KEM-768 still protects the session. Deploying hybrid TLS today carries no security downside compared to classical-only TLS, and the performance overhead is negligible for most workloads.

What is the performance overhead of hybrid TLS handshakes?

The primary overhead is bandwidth rather than computation. ML-KEM-768 adds approximately 1.1-1.2 KB per handshake compared to classical X25519, distributed across the ClientHello and ServerHello messages. CPU overhead is negligible on modern hardware. For most web workloads, the latency impact is under 2 milliseconds per handshake and is imperceptible to users. Cloudflare, Google, and AWS have all deployed hybrid TLS at production scale without reporting meaningful performance regression.

Do hybrid TLS deployments protect against harvest-now-decrypt-later attacks?

Yes, and this is the primary motivation for deploying hybrid TLS now rather than waiting for quantum computers to arrive. Adversaries capable of recording encrypted TLS traffic today can store it and decrypt it later when quantum computers become available. Hybrid TLS with ML-KEM-768 prevents this: even if a cryptanalytically relevant quantum computer is built in the future, it cannot recover session keys established with X25519MLKEM768, because ML-KEM-768 is quantum-resistant. The benefit is immediate: traffic protected today by hybrid TLS remains protected even against a future adversary with quantum capabilities.

Does enabling hybrid TLS break anything?

No, provided classical fallback groups are retained in the configuration. Clients that do not support hybrid groups will negotiate the next available classical group (typically X25519 or P-256) using standard TLS group negotiation. Hybrid TLS is strictly additive: servers that support it provide better security to hybrid-capable clients and identical security to classical-only clients. No clients should be broken by a server adding X25519MLKEM768 to its supported groups list.

When will post-quantum TLS certificates be available?

Post-quantum certificates using ML-DSA (NIST FIPS 204) are in active development at the IETF and in several certificate authorities. The PKIX profile for ML-DSA certificates is being standardised in draft-ietf-lamps-dilithium-certificates. Production availability from major public CAs is expected in 2027-2028, with browser root store trust following. In the meantime, hybrid key exchange addresses the most urgent HNDL risk, and certificate key rotation (short validity periods) limits the exposure from classical certificate signatures.

WP-002: Post-Quantum Attestation | WP-013: Post-Quantum PKI Migration | WP-020: DORA Compliance

▶ About
▶ Solutions
▶ Legal
▶ Trust & Security

Contact

truth layer | yes | no | proof