COALITION FOR SECURE AI

Addendum: Post-Quantum Cryptography Considerations for MCP

Workstream 4: Secure Design Patterns for Agentic Systems

Version 05f6d750f276be68fac11059059bd08076d49324

15 July 2026

Contents

Addendum: Post-Quantum Cryptography Considerations for Model Context Protocol (MCP)

Purpose

Post-quantum cryptography (PQC) has moved from research to standards: NIST has ratified ML-KEM (FIPS 203), ML-DSA (FIPS 204), and SLH-DSA (FIPS 205), and the US NSA's CNSA 2.0 sets a phased migration schedule for national-security systems. This addendum gives MCP deployment teams a short summary of *what is changing*, *where to prepare*, and *how to reduce migration cost*.

This is not an implementation guide, an algorithm-selection guide, or a substitute for consulting NIST/CNSA specs, IETF drafts, or your platform's TLS documentation. MCP tool and server authors should not be writing cryptographic code; the recommendations below deliberately push all cryptographic implementation into well-supported components (TLS libraries, proxies, managed platforms) that ship, audit, and update PQC on the reader's behalf.

What is changing, and when

There are two distinct migration problems, and they move on different timelines:

- **Transport confidentiality — "Harvest-Now-Decrypt-Later" (HNDL).** Classical key-exchange (RSA, ECDH) is vulnerable to Shor's algorithm. Traffic captured today can be decrypted post-Q-day. Any data that traverses an MCP boundary with a sensitivity horizon longer than the quantum-computing timeline (biometrics, identity roots, classified material, long-term contractual data) is exposed *today* — this is the pressing case.
- **Signature integrity — forgery risk.** A quantum adversary can only forge signatures *after* Q-day. But the signature migration is structurally slower because it touches JOSE/JWT, OAuth/OIDC, code-signing, certificate chains, and audit-trail formats — all still stabilizing PQ variants. Practitioners should track proposals and prepare to migrate to new standards once they emerge.

Reference timelines readers should track:

Milestone	Source	Date
ML-KEM / ML-DSA / SLH-DSA ratified	NIST FIPS 203 / 204 / 205	2024
CNSA 2.0 required for new NSS procurements	NSA CNSA 2.0	Jan 2027
TLS 1.3 (or successor) required across federal systems	EO 14144	Jan 2030
PQC key encapsulation	EO 14412	Jan 2031
PQC digital signatures	EO 14412	Jan 2032
CNSA 2.0 exclusive use — federal systems	NSA CNSA 2.0	2033
NSM-10 full migration deadline	NSM-10	2035
Industry Q-day working estimate	Google, Cloudflare	~2029

Recommendations for MCP deployments

The controlling principle: **do not build your own PQC — inherit it from well-supported, actively audited components.** The practices below reduce migration cost without turning MCP tool authors into cryptographers.

1. Rely on well-supported TLS libraries and platforms

PQ-capable key exchange is arriving natively in mainstream TLS stacks (OpenSSL, BoringSSL, WolfSSL, rustls) and in managed TLS at the CDN, load-balancer, service-mesh, and cloud-provider layer. The industry-deployed hybrid — combining a classical curve with ML-KEM — is being rolled out across major CSPs and browsers through 2025–2026.

Prefer these paths over library-specific PQ bindings written into MCP server code. **Do not invent MCP-specific cryptographic framings.**

2. Externalize crypto at the network edge (proxy / sidecar / gateway)

Rather than adding PQ-TLS support to every MCP tool, server, or agent, terminate TLS at a **proxy, gateway, or sidecar** and let the MCP process consume plain HTTP inside the trust boundary. This is the same pattern operators already use for mTLS, WAFs, and observability, and it lets platform teams roll out PQ transport changes independently of application code.

Concretely:

- **Server side.** A reverse proxy or sidecar (Envoy, nginx, Caddy, HAProxy — any TLS build with hybrid PQ key-exchange) fronts your MCP server and terminates the hybrid TLS handshake. The MCP server itself does not link a PQC library.
- **Client / agent side.** Agent frameworks and MCP clients calling remote MCP servers can route through a forward proxy or egress gateway with PQ-capable outbound TLS.
- **Managed API gateways and service meshes.** Treat these as first-class candidates as soon as they add PQ support; they are typically the cheapest place to upgrade.

This pattern:

- Isolates cryptographic implementation from MCP protocol code, so upgrades do not require touching every tool.
- Gives you a single control point when defaults, hybrid choices, or standards shift.
- Reduces the risk that MCP tool authors write ad-hoc cryptographic code.

3. Inventory cryptographic surfaces

Enumerate the places cryptography touches your MCP surface so migration planning is bounded:

- **Transport.** MCP server ingress, MCP client egress, internal service-to-service links.
- **Server / workload identity.** Certificate authorities, key formats, revocation channels, mTLS trust roots.
- **Authorization tokens.** OAuth/OIDC, JWTs. Note that PQ-safe JOSE (RFC 9964 onward) is still emerging.
- **Signed receipts and attestations.** Agent-credential receipts, tool-invocation attestations, audit trails.
- **Data at rest.** Records whose sensitivity horizon exceeds Q-day estimates.

You do not need to *migrate* everything today. You need to *know where you would migrate* when standards land.

4. Watch third-party protocol progress; don't front-run it

PQ signature protocols (JOSE, TUF/Sigstore, code-signing formats) are still moving. Track their release notes and adopt when your libraries do. Do not build MCP-specific PQ-signed variants of existing protocols — the threat on digital signatures is not active pre-Q-day, and premature implementations tend to age poorly.

5. Data at rest

For long-lived data traversing MCP, encrypt at rest with **AES-256** (satisfies CNSA 2.0; quantum-resistant with a wide margin under Grover's algorithm). AES-128 remains acceptable where CNSA 2.0 is not a binding requirement and performance matters.

Out of scope

- **Specific algorithm parameter choices** beyond the NIST FIPS 203 / 204 / 205 anchors and CNSA 2.0 constraints. Defer to your platform, TLS library, or regulator.
- **Custom cryptographic protocols or hybrid schemes** not already present in a mainline TLS implementation or a NIST/IETF-standardized protocol.
- **Symmetric primitives** — HMACs, ordinary hash-based constructions, hash-based state attestations. Shor's algorithm does not act on these; Grover halves the exponent on paper, but in practice only offers a modest speedup due to the limitations to parallelize computation, which any 256-bit construction absorbs. Do not place these on the ML-KEM / ML-DSA migration timeline unless there is a specific reason.

When to revisit this addendum

- ML-KEM and ML-DSA become *defaults* (not opt-in) in mainstream TLS libraries and managed platforms.
- JOSE PQ variants stabilize sufficiently for MCP-adjacent JWT / OAuth flows.
- Industry Q-day estimates shift materially — later or sooner.
- New NIST selections (HQC follow-on, alternate signatures) begin landing in shipping software.

References

- NIST FIPS 203 — ML-KEM. https://csrc.nist.gov/pubs/fips/203/final
- NIST FIPS 204 — ML-DSA. https://csrc.nist.gov/pubs/fips/204/final
- NIST FIPS 205 — SLH-DSA. https://csrc.nist.gov/pubs/fips/205/final
- NSA CNSA 2.0. https://media.defense.gov/2025/May/30/2003728741/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS.PD
- UK NCSC — Preparing for Quantum-Safe Cryptography. https://www.ncsc.gov.uk/paper/preparing-for-quantum-safe-cryptography
- Google — Cryptography Migration Timeline (2029 Q-day). https://blog.google/innovation-and-ai/technology/safety-security/cryptography-migration-timeline/
- Cloudflare — Post-Quantum Roadmap. https://blog.cloudflare.com/post-quantum-roadmap/
- IETF RFC 9964 — Composite ML-KEM constructions in JOSE (May 2026).
- Executive Order 14144 — Strengthening and Promoting Innovation in the Nation's Cybersecurity.