International Journal of Cyber Threat Intelligence and Secure Networking (IJCTISN)
eISSN: 3087-4297
Volume03 Issue11 | Page No: - 01-13

Received: 28 Sep 2026, Revised: 24 Oct 2026, Accepted: 07 Nov 2026, Published: 19 Nov 2026

DOI: https://doi.org/10.55640/ijctisn-v03i11-01

Post-Quantum Cryptography Migration for Agentic Identity Systems: A Practitioner Architecture for Regulated Multi-Cloud Enterprises

Parijat Sharma
Independent Researcher, USA

ABSTRACT: AI agents and other non-human identities depend on public-key cryptography for authentication, token signing, certificates, key exchange, software provenance, and secure service-to-service communication. Future quantum computers could undermine the RSA and elliptic-curve mechanisms these depend on, while data with a long confidentiality life is already exposed to harvest-now-decrypt-later collection. This paper proposes a crypto-agile migration architecture that takes the agent identity lifecycle, rather than the application or the network protocol, as the unit of migration. It inventories cryptographic dependencies and links each to agent ownership, privilege, data sensitivity, autonomy, recovery requirement, and revocation path; classifies and prioritizes them by confidentiality lifetime and exposure; establishes crypto-agile control points that externalize algorithm choice from application code; validates post-quantum capability against certificate, token, and message size limits before any cohort moves; runs hybrid or dual-trust operation through a bounded compatibility window; migrates by risk-ordered cohort with defined stop conditions; and retires classical-only trust only once rollback, audit, and recovery criteria are satisfied. The architecture is designed and has not been validated as a completed enterprise migration. Algorithm parameter sizes used in the analysis are taken from the published standards; migration percentages, downtime, and performance impact are not claimed and are identified as pilot outputs.

KEYWORDS: Post-quantum cryptography; agentic AI; non-human identity; workload identity; crypto agility; hybrid cryptography; ML-KEM; ML-DSA; identity governance; multi-cloud security; credential lifecycle; certificate migration; harvest now decrypt later; regulated enterprise; cryptographic inventory.

Introduction

The migration to post-quantum cryptography is usually presented as a protocol problem. Inventory the algorithms, upgrade the libraries, replace the certificates, and move the traffic. For most of an enterprise estate that framing is roughly correct, if considerably harder than it sounds.

For agent and non-human identity systems it is the wrong unit of work, and this paper is an argument that the unit should be the identity lifecycle instead.

The reason is that a machine identity is not principally a protocol endpoint. It is a bundle of trust relationships with a lifecycle attached: an issuance path, an attestation mechanism, a credential store, a set of relying parties that accept its assertions, a revocation path, and a recovery procedure. Cryptography is embedded at every one of those points, and they do not migrate independently. Upgrading a workload's TLS stack while its federation partner still validates the old signature accomplishes nothing except a failed handshake. Upgrading the certificate authority while ten thousand pinned trust stores still carry the old anchor produces an outage rather than a migration. The dependencies run through the identity, so the identity is the correct unit.

Three properties of agent identities make this harder than the human-facing equivalent.

They are numerous and they churn. A container's

identity may exist for seconds. A migration plan expressed as a list of endpoints to upgrade has no purchase on a population that is continuously recreated, and any inventory built by enumeration is stale before it is complete.

They are created outside central identity processes. Pipelines, infrastructure-as-code, and integration configuration all create identities, so a central migration programme does not have a complete list of what it must migrate and cannot get one by asking.

Their trust relationships cross boundaries the migration programme does not control. Cross-cloud federation, third-party APIs, certificate authorities, key management services, and hardware-backed key stores are all participants, and the achievable cryptographic profile for any relationship is set by its least capable participant.

The urgency is asymmetric in a way that shapes the sequencing. Signature forgery becomes a problem only when a cryptographically relevant quantum computer exists. Confidentiality loss does not wait, because an adversary can collect encrypted traffic, archives, tokens, and protected datasets now and decrypt them when capability arrives. Anything with a long confidentiality life is therefore already exposed, which means key establishment must be migrated ahead of signatures on risk grounds even though signature migration is the more visible piece of work.

NIST published the first three standards in August 2024 [1]-[3] and set out a transition timeline in which 112-bit-security classical algorithms are deprecated after 2030 and disallowed after 2035 [4]. That is the planning envelope. It is long enough to do this properly and short enough that starting late is a real risk for estates with long-lived data and slow-moving third parties.

This paper contributes four things.

First, it makes the agent identity lifecycle the migration unit, linking each cryptographic dependency to owner, privilege, data sensitivity, autonomy, recovery requirement, and revocation path rather than to a hostname.

Second, it specifies dual-trust transition patterns for identity systems specifically, covering what runs old and new simultaneously and how acceptance is progressively restricted, together with the split-trust failure that a partial migration produces.

Third, it treats size growth as a first-class compatibility risk rather than as a performance footnote, on the analysis in Section 5.2 that the binding constraints for identity systems are structural limits on tokens, headers, and certificate chains rather than CPU cost.

Fourth, it specifies the evidence package a regulated enterprise must produce, so that migration is auditable as a controlled change rather than assertable as a completed one.

1.1 Note on Scope and Evidence

This paper is a practitioner architecture. It is not a report of a completed migration and does not present modelled figures as measurements.

The architecture was developed and refined during 2026, drawing on the author's work in multi-cloud security architecture, identity governance, non-human identity, agentic AI security, cryptographic control design, and regulated-enterprise risk management.

1.1 What Exists

Practical architecture work: identity lifecycle controls, cloud key-management patterns, certificate and secret governance, workload identity design, agent authorization concepts, logging requirements, and recovery planning.

1.2 What Is Not Claimed

The author has not led, and does not claim, a completed production migration of AI-agent identity systems to post-quantum cryptography across an enterprise multi-cloud estate. There is no defensible figure for the share of an agent identity estate migrated, the number of agents converted, measured downtime, or end-to-end performance impact. These are not estimated. Section 8 states what a pilot would produce.

1.3 On the Numbers That Do Appear

Section 5.2 and Table 2 give key, ciphertext, and signature sizes for the standardized algorithms. Those are published parameters of FIPS 203, 204, and 205 [1]-[3] and are exact, not measured by the author and not

modelled. Where the paper reasons about the operational consequence of those sizes, the reasoning is labeled as analysis rather than as observation.

1.4 Standards Currency

Claims about standardization status are stated as of August 2026 and are attributed to dated documents. Where a mechanism is still an Internet-Draft rather than a published standard, the paper says so rather than implying settled status.

Systems are described by function. No employer, product, vendor, or internal control state is disclosed.

2. Background and related work

2.1 The standards position

The published position as of August 2026 is as follows. Key encapsulation is standardized as ML-KEM in FIPS 203 [1], and digital signatures as ML-DSA in FIPS 204 [2]. Alongside them sits FIPS 205, which specifies SLH-DSA, a stateless hash-based scheme offered as the conservative alternative because it rests on different mathematical assumptions from the lattice constructions [3]. All three were finalized in August 2024. NIST IR 8547 sets out the transition from classical to post-quantum standards and the deprecation schedule [4]. The NCCoE practice guide addresses migration mechanics, and its volume on cryptographic discovery is directly relevant to the inventory problem this paper treats as the first stage [5].

The theoretical basis is older and is not in dispute. Shor's algorithm gives a quantum polynomial-time method for integer factorization and discrete logarithms [6], which is the assumption underlying RSA and elliptic-curve cryptography. Grover's algorithm gives a quadratic speedup for unstructured search [7], which affects symmetric primitives far less severely and is the reason symmetric cryptography and hashing require parameter increases rather than replacement.

One point of currency matters for accuracy. Hybrid key exchange combining X25519 with ML-KEM-768 is widely deployed in browsers and at major content delivery networks, and forms a substantial share of TLS 1.3 handshakes as of early 2026. The IETF specification for hybrid ECDHE-MLKEM key agreement in TLS 1.3 remains an Internet-Draft rather than a published RFC as of this writing, and post-quantum signatures inside the TLS handshake are markedly less deployed than post-quantum key exchange. The paper states this asymmetry because it directly determines what an enterprise can actually do now: key establishment can be hybridized today in many paths, while certificate-chain migration cannot proceed at the same pace.

2.2 Crypto agility

Crypto agility is the property that lets algorithms be replaced without rewriting the systems that use them, and it is what determines whether this migration is a project or a permanent capability. Rameshan and Messmer contribute both an assessment framework for application-level cryptographic agility [8] and an intent-based API design in which callers declare what they need rather than which algorithm to use [9]. The second is the pattern this architecture adopts at its control points, because externalizing algorithm choice from application code is what makes Stage 4 of Section 4 possible at all.

Balaji et al. report on operationalizing post-quantum TLS in financial infrastructure, including automated configuration profiling and hybrid deployment [10], which is among the closest available work to a regulated-enterprise migration account. Jaikissoon addresses hybrid deployment from the policy side [11].

The gap in this body of work is consistent: it is application-centric and protocol-centric. It treats the cryptographic dependency as a property of a service. None of it links the dependency to the identity that the cryptography exists to establish, and therefore none of it can prioritize by agent privilege, autonomy, or revocation requirement, or reason about what happens to an agent's delegation chain when one hop migrates and the next does not.

2.3 Identity as the missing frame

The identity literature supplies the frame the migration literature lacks. NIST SP 800-207A establishes identity-based access control for cloud-native applications across multi-cloud environments [12]. Avirneni develops the identity control plane as a unifying layer [13]. Tallam addresses authorization propagation and delegation across multi-agent systems [14], which is precisely the structure that a partial cryptographic migration fractures. The OWASP non-human identity risks [15] catalogue the estate this

migration must cover, and long-lived secrets and improper offboarding are as much migration obstacles as they are security risks: a credential nobody owns is a credential nobody can migrate.

This paper joins the two. The migration mechanics come from [5], [8]-[11]. The unit of work, the prioritization basis, and the governance frame come from [12]-[15].

3. Where the vulnerable cryptography sits

Public-key cryptography is embedded at seven layers of an agent identity system. Enumerating them precisely matters, because a migration programme that misses a layer produces a split trust state rather than a partial success.

Agent authentication. Client certificates, mutual TLS, signed client assertions, service-account keys, workload identity federation, and API authentication.

Token and message integrity. Signed access tokens, identity tokens, delegated authorization tokens, webhook signatures, event signatures, and agent-to-agent message signing.

Key establishment and transport. TLS handshakes, VPN and private-link encryption, secure channels to tools, secret retrieval, and key wrapping.

Credential storage. Key vaults, hardware-backed key stores, certificate stores, workload secret stores, pipeline variables, and local agent runtimes.

Software and model trust. Code-signing certificates, container and package signatures, model-artifact signatures, update manifests, and provenance attestations.

Cross-cloud trust. Federation metadata, identity-provider signing keys, certificate chains, workload token exchanges, and third-party integrations.

Audit and evidence. Signed logs, immutable records, timestamping, and archival signatures that must remain verifiable across long retention periods.

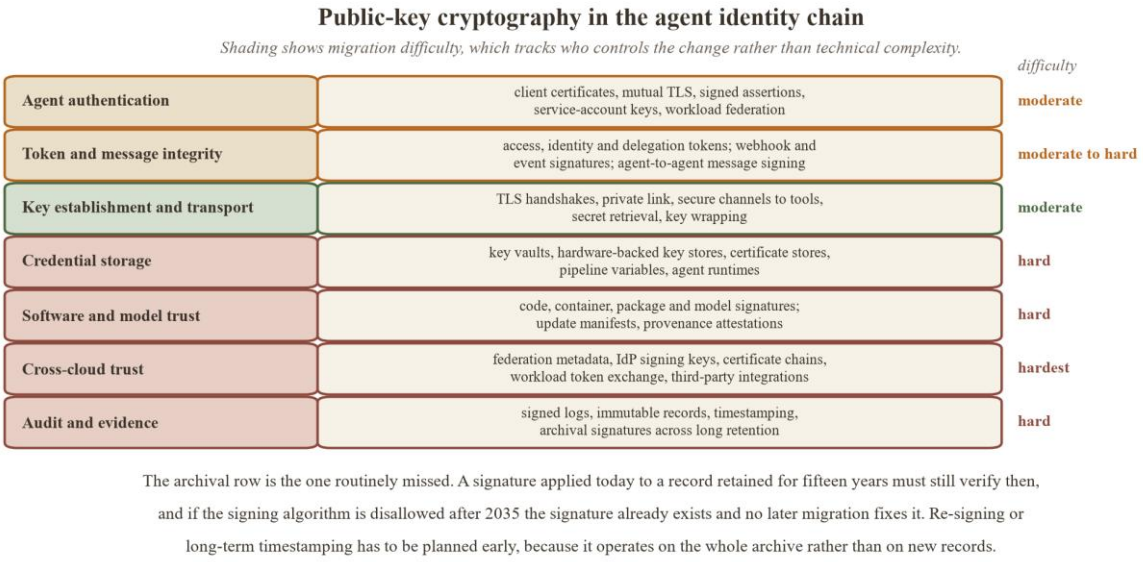

Figure 1. Where public-key cryptography sits in an agent identity chain, from workload attestation through to archived evidence. Shading indicates migration difficulty as assessed in Table 1. The concentration of hard-to-change dependencies at the cross-cloud trust and archival layers is the reason those two layers set the overall programme duration, not the agent runtime layer where most attention goes.

The layers are not equally difficult, and difficulty tracks control rather than complexity. Table 1 sets out the assessment.

Layer	Mechanism	Who controls the change	Difficulty	Why
Agent authentication	mTLS, client certificates, signed assertions, workload federation	Mostly the enterprise	Moderate	Change is largely internal, but federation partners gate the pace
Token and message integrity	Signed access, identity and delegation tokens; event signatures	Enterprise and identity provider	Moderate to hard	Every relying party must accept the new signature before issuance can switch
Key establishment and transport	TLS, private link, key wrapping, secret retrieval	Enterprise, cloud provider, network vendors	Moderate	Hybrid key exchange is the most deployable step available today
Credential storage	Vaults, hardware-backed key stores, certificate stores	Cloud provider and hardware vendor	Hard	Algorithm support and validated modules are set by the vendor, not the enterprise
Software and model trust	Code, container, package and model signing; provenance	Enterprise and supply chain	Hard	Verification happens in places the enterprise does not run
Cross-cloud trust	Federation metadata, IdP signing keys, certificate chains, token exchange	Multiple providers and third parties	Hardest	Bounded by the least capable participant; no unilateral progress possible
Audit and evidence	Signed logs, timestamping, archival signatures	Enterprise, with regulatory constraint	Hard	Signatures must remain verifiable for the full retention period, which may exceed the classical algorithms' remaining life

Table 1. Vulnerable cryptography by layer, with migration difficulty and its cause. Difficulty is determined by who controls the change rather than by technical complexity, which is why the cross-cloud layer is hardest despite being conceptually simple.

The archival row deserves separate comment because it is routinely missed. A signature applied today to a record retained for fifteen years must remain verifiable across that period. If the signing algorithm is

disallowed after 2035 [4], then records signed classically and retained beyond that point have a verification problem that no future migration fixes, because the signature already exists. The mitigation is to re-sign or to apply long-term timestamping before the deprecation date, and it must be planned early because it operates on the entire archive rather than on new records.

4. The migration architecture

The architecture is a phased, reversible process of ten stages, not a cutover. Every stage has an exit criterion and every cohort has a rollback trigger.

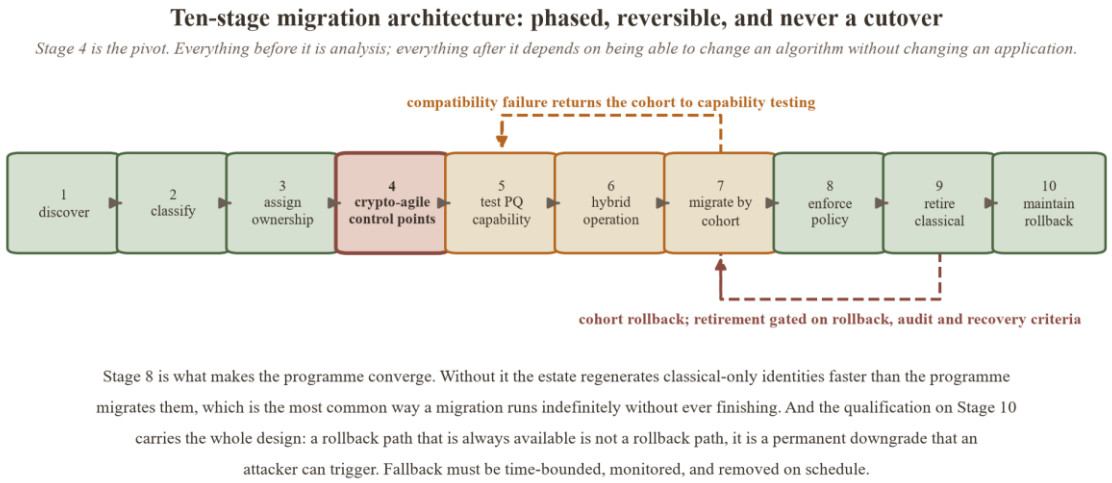

Figure 2. The ten-stage migration architecture. The forward path runs left to right; the two return paths are the compatibility-failure loop from Stage 7 back to Stage 5 and the cohort rollback loop from Stage 9. Stage 4, establishing crypto-agile control points, is drawn as the pivot because it is what makes every later stage reversible rather than a one-way change.

Stage 1, discover. Inventory agents, service identities, certificates, keys, algorithms, protocols, libraries, trust stores, token issuers, data flows, and dependent systems. Discovery tooling and method follow the NCCoE guidance [5]. Discovery is continuous rather than a phase, because the population churns.

Stage 2, classify. Rank each dependency by privilege, data sensitivity, confidentiality lifetime, exposure, business criticality, regulatory obligation, and ease of replacement. Confidentiality lifetime is the dimension that orders the work, because it is the one that decides what is already exposed to harvest-now-decrypt-later collection rather than what will be exposed later.

Stage 3, assign ownership. Link every cryptographic dependency to a technical owner, a business owner, a recovery owner, and an approved migration path. An unowned dependency cannot be migrated, and discovering that a dependency is unowned is itself a finding worth recording.

Stage 4, establish crypto-agile control points. Externalize algorithms and key parameters from application code, and use centralized policy, versioned profiles, and replaceable providers [8], [9]. This is the pivot stage. Everything before it is analysis; everything after it depends on being able to change an algorithm without changing an application.

Stage 5, test post-quantum capability. Validate supported algorithms, certificate and token sizes, handshake behavior, memory and CPU impact, latency, hardware compatibility, observability, and failure modes. Section 5.2 argues that the size tests matter more than the performance tests.

Stage 6, introduce hybrid operation. Where supported, run classical and post-quantum mechanisms together so

that compromise of either alone does not defeat the transition control, maintaining dual certificates, dual trust paths, or negotiated protocol profiles as appropriate [10], [11].

Stage 7, migrate by cohort. Begin with low-risk test agents, then internal non-production agents, then production identities ordered by risk and dependency complexity.

Stage 8, enforce policy. Prevent creation of new non-compliant identities, require approved cryptographic profiles, and continuously detect algorithm and credential drift. Without this stage the estate regenerates classical-only identities faster than the programme migrates them, which is the most common way a migration programme fails to converge.

Stage 9, retire classical-only trust. Remove legacy keys, certificates, libraries, trust anchors, and fallback modes, but only after compatibility, rollback, audit, and recovery criteria are all satisfied.

Stage 10, maintain rollback. Preserve time-bounded fallback credentials, configuration versions, trust-store snapshots, and tested recovery procedures, so an incompatible agent can be restored without leaving an uncontrolled permanent downgrade path.

The qualification on Stage 10 is the one that separates a controlled migration from an indefinite one. A rollback path that is always available is not a rollback path; it is a permanent downgrade that an attacker can trigger. Fallback must be time-bounded, monitored, and removed on schedule.

5. Migrating without downtime in a regulated environment

5.1 Coexistence and cohorts

Zero-downtime migration depends on coexistence, segmentation, and evidence.

Agents first receive a new cryptographic profile while retaining a strictly time-limited classical path. Trust services accept both approved profiles during the migration window. Routing or policy controls move small cohorts to the new path, and health checks verify authentication success, token validation, latency, tool access, and downstream compatibility before the next cohort proceeds.

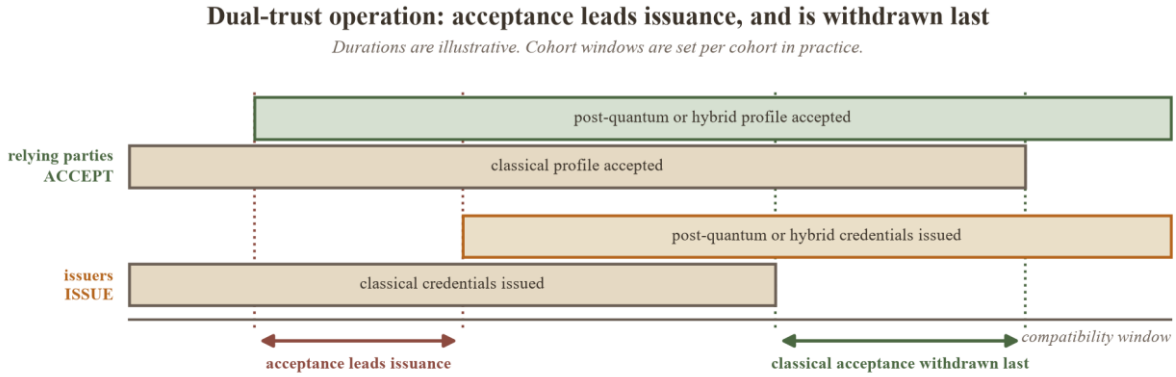

Figure 3. Dual-trust operation through the compatibility window. The upper band shows what issuers issue; the lower shows what relying parties accept. The two are deliberately offset: acceptance of the new profile must be established across the relying-party population before issuance switches, and acceptance of the old profile is withdrawn last. Compressing this offset is what produces the split-trust failure of Section 6.2. Durations are illustrative and are set per cohort in practice.

The offset in Figure 3 is the whole technique and it is easy to get backwards. Relying parties must accept the

new profile before issuers begin issuing it, because an assertion nobody can validate is an outage. Symmetrically, acceptance of the old profile is the last thing withdrawn, because withdrawing it early strands anything that has not yet moved.

5.2 Size is the binding constraint, not speed

The performance discussion around post-quantum cryptography usually centres on CPU cost. For agent identity systems the analysis here is that size is the binding constraint, and that the failures it produces are structural rather than gradual.

Table 2 gives the published parameter sizes.

Algorithm	Public key (bytes)	Signature or ciphertext (bytes)	Standard
ECDSA P-256	64	64 (signature)	Classical baseline
RSA-2048	256	256 (signature)	Classical baseline
X25519	32	32 (key share)	Classical baseline
ML-KEM-512	800	768 (ciphertext)	FIPS 203 [1]
ML-KEM-768	1184	1088 (ciphertext)	FIPS 203 [1]
ML-KEM-1024	1568	1568 (ciphertext)	FIPS 203 [1]
ML-DSA-44	1312	2420 (signature)	FIPS 204 [2]
ML-DSA-65	1952	3309 (signature)	FIPS 204 [2]
ML-DSA-87	2592	4627 (signature)	FIPS 204 [2]
SLH-DSA-128s	32	7856 (signature)	FIPS 205 [3]
SLH-DSA-128f	32	17088 (signature)	FIPS 205 [3]

Table 2. Published key, ciphertext, and signature sizes for the standardized algorithms and their classical baselines. These are exact parameters from FIPS 203, 204, and 205, not measurements. The ratios, rather than the absolute values, are what drive the compatibility analysis below.

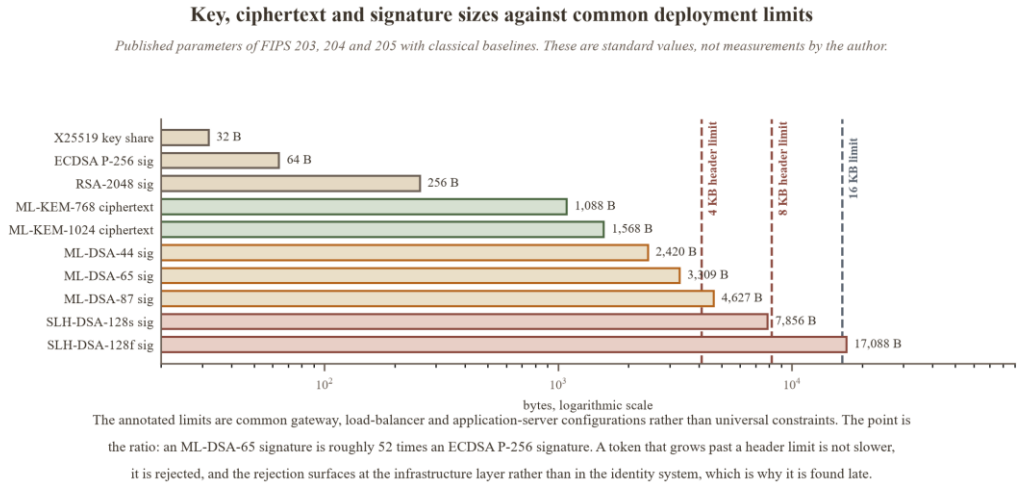

Figure 4. Signature and key sizes on a logarithmic scale, with three structural limits annotated. Values are published standard parameters, not measurements. The annotated limits are common deployment constraints rather than universal ones, and the point of the figure is the relationship: an ML-DSA-65 signature is approximately 52 times an ECDSA P-256 signature, and a conservative SLH-DSA signature is larger still by more than two further orders of magnitude relative to the classical baseline.

The operational consequences follow from the ratios and they are worth stating concretely.

A signed token carrying an ML-DSA-65 signature is roughly three kilobytes larger than its classical equivalent before encoding overhead. Tokens are routinely transported in HTTP headers, and header size limits are commonly configured in the four to eight kilobyte range at gateways, load balancers, and application servers. A token that grows past the limit does not degrade; the request is rejected. That is a hard failure, and it appears at the infrastructure layer rather than in the identity system, which is why it is frequently discovered late.

Certificate chains compound the effect. A chain with a leaf, an intermediate, and a root, each carrying a post-quantum signature and public key, grows by roughly the sum of those increases, and the growth lands in the TLS handshake where it can push the flight past the initial congestion window and add a round trip on high-latency paths.

Hardware-backed key stores present a different constraint again. Support for the new algorithms, and availability of validated modules, is set by the vendor, and where a device does not support ML-DSA there is no configuration that produces post-quantum signing from it.

The practical rule this analysis produces for Stage 5 is that size compatibility must be tested against gateways, load balancers, application servers, header limits, embedded runtimes, and hardware modules before any cohort is scheduled, and that the test must exercise the full chain rather than the endpoint.

5.3 Evidence for regulators and auditors

For a regulated environment, the evidence package is a deliverable of the migration rather than a byproduct.

Artifact	What it demonstrates
Cryptographic inventory with coverage statement	What was in scope, and, importantly, what was known not to be covered
Risk classification and prioritization basis	That sequencing followed confidentiality lifetime and exposure rather than convenience
Approved cryptographic profiles, versioned	That algorithm choice is governed and current
Change records per cohort	That migration was a controlled change, not a configuration drift
Test cases and results, including size and compatibility tests	That capability was validated before deployment
Performance results before and after	That impact was measured rather than assumed
Exception register with dated entries and owners	That unsupported dependencies are tracked, not forgotten
Rollback test results	That the reversibility claim was exercised, not asserted
Access approvals and key custody records	That key material was handled under dual control
Certificate issuance and revocation history	That the trust chain is accountable end to end
Monitoring results and drift detections	That policy enforcement is operating
Formal sign-off per phase	That accountability is recorded at each gate

Table 3. The regulated-environment evidence package. Controls should map to the applicable regulatory and enterprise requirements rather than

merely record that a new algorithm was enabled, and the coverage statement in row one is the artifact auditors most often find missing.

5.4 Where support does not exist

Where a cloud, appliance, or third party lacks support, the architecture applies compensating controls: isolate the dependency, shorten credential lifetime, strengthen key protection, minimize privilege, restrict network paths, add transaction-level authorization, increase monitoring, and record a dated exception with a vendor or replacement roadmap.

Shortening credential lifetime is the most useful of these and the reasoning is worth making explicit. If a dependency cannot be made quantum-resistant, reducing how long each credential is valid reduces the value of any single captured credential, which does not solve the confidentiality problem for data already collected but does limit the authentication exposure. It is a genuine mitigation rather than a gesture, and it is available almost everywhere.

The governance rule attached is that an unsupported component must not silently become a permanent blocker. An exception without a date is an unrecorded acceptance of risk.

6. Where this breaks down

6.1 Uneven support and moving parts

Standards, implementations, and product support move at different speeds. Even where a standard is stable, protocol integration, validated modules, certificate ecosystems, hardware security modules, identity providers, API gateways, and cloud-native services may not support the same algorithms or the same transition patterns. The asymmetry noted in Section 2.1, where hybrid key exchange is widely deployed while post-quantum signatures in the certificate chain are not, is the current instance of this and it will not be the last.

6.2 Split trust

A migration that fails mid-flight produces split trust: some agents present new credentials while downstream services recognize only the old chain.

This is the characteristic failure of this migration and it deserves precision, because it is not a simple outage. In a delegation chain of the kind agent systems routinely build [14], a partial migration can break attribution rather than connectivity. If one hop validates the new profile and the next does not, the chain may continue to function while the identity assertions carried across it are no longer end-to-end verifiable. The system keeps working and the evidence quietly stops being sound, which is worse than a failure, because a failure is noticed.

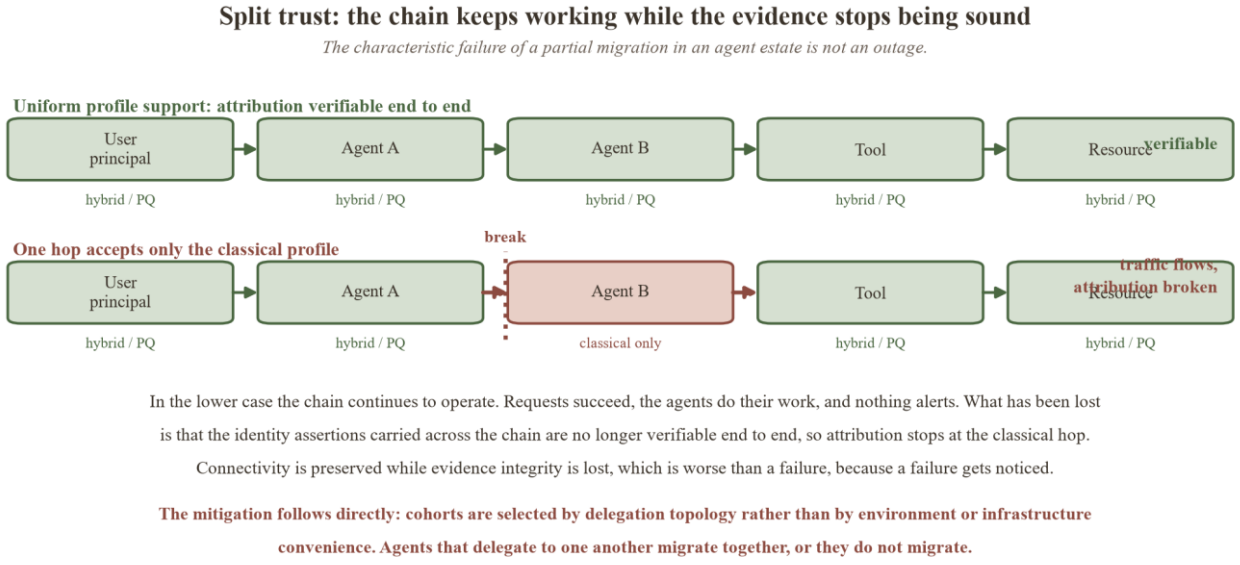

Figure 5. Split trust in an agent delegation chain. In the upper case, uniform profile support preserves end-to-end verifiability. In the lower case, one hop accepts only the classical profile: the chain continues to operate, but the

attribution path is broken at that hop and the assertion is no longer verifiable end to end. Connectivity is preserved while evidence integrity is lost, which is why cohort selection must follow delegation topology rather than infrastructure convenience.

The mitigation is that cohorts are selected by delegation topology rather than by environment or convenience. Agents that delegate to one another migrate together or not at all.

6.3 Hybrid complexity

Hybrid operation is the right transition mechanism and it is not free. Two trust mechanisms mean more keys, more policies, more telemetry, more renewal paths, and considerably more failure combinations. Every cohort needs dependency testing, transaction monitoring, automatic stop conditions, and a tested rollback that does not normalize indefinite downgrade.

6.4 The real limits

The limits are incomplete cryptographic visibility, third-party dependency, legacy code, unavailable validated implementations, uncertain performance at scale, and absent ownership for machine identities [15]. The last of these is the one that most often stops a migration before it starts: a credential nobody owns is a credential nobody can approve changing.

7. Contribution boundary

Two companion manuscripts in preparation by the author address detection-ready telemetry pipelines and non-human identity governance architecture respectively. This paper does not restate either. It assumes an inventory of non-human identities and their ownership, which the governance work specifies, and treats that inventory as the input to Stage 1 rather than as a contribution here.

8. What a pilot would measure

No completed enterprise migration is claimed. Table 4 is the measurement design.

Category	Metrics
Discovery coverage	Share of agent identities, certificates, keys, token issuers, libraries and protocols inventoried; and the size of the known-unknown remainder
Readiness	Share of dependencies supporting post-quantum, hybrid, classical-only, or unknown profiles
Migration progress	Number and share of agents migrated by environment, business criticality and cloud
Compatibility	Failed integrations, unsupported libraries, certificate-chain issues, token-size rejections, hardware limitations
Performance	Authentication latency, handshake time, CPU and memory use, bandwidth growth, token and certificate size, throughput, before and after
Reliability	Downtime, failed authentications, rollback events, mean time to recover, error rates per cohort
Governance	Exceptions opened and closed, non-compliant identity creations prevented, credentials rotated, trust anchors retired, evidence artifacts completed
Security	Reduction in classical-only exposure, credential lifetime

	reduction, revocation time, coverage of high-value long-retention data paths

Table 4. Pilot measurement design. The known-unknown remainder in the first row is the metric most often omitted and the one that determines whether every other percentage is meaningful, since a share of an unknown denominator is not a measurement.

The components to build next are an enterprise cryptographic dependency graph tied to the non-human identity inventory; automated discovery and policy-as-code; a compatibility test harness exercising full chains rather than endpoints; hybrid certificate and token experiments; performance benchmarks; a migration evidence ledger; and a risk dashboard showing exposure, readiness, exceptions, and retirement status.

9. Conclusion

The case for treating agent identity as the unit of post-quantum migration rests on a structural observation rather than a preference. Cryptographic dependencies in an agent system are not distributed across endpoints that can be upgraded independently. They are bound to identities, and those identities carry trust relationships that cross clouds, providers, and organizational boundaries, so the achievable profile for any relationship is set by its least capable participant. An endpoint-ordered migration plan cannot see that constraint. An identity-ordered one is built around it.

Three things follow that a protocol-centric plan would not produce. Sequencing is driven by confidentiality lifetime, because harvest-now-decrypt-later exposure is present today while forgery exposure is not, which puts key establishment ahead of signatures on risk grounds even though signatures are the more visible work. Compatibility testing is driven by size rather than speed, because the analysis in Section 5.2 shows the binding constraints are structural limits on tokens, headers, and certificate chains, and those fail hard rather than gradually. And cohorts are selected by delegation topology, because the characteristic failure of a partial migration in an agent estate is not an outage but a silently broken attribution chain that continues to carry traffic.

Crypto agility is the durable contribution rather than the migration itself. An estate that can change algorithm without changing application code has solved a recurring problem; an estate that has completed one migration by hand has solved it once. Stage 4 is therefore the stage worth over-investing in, and it is the one most likely to be skipped under schedule pressure because it produces no visible migration progress.

What remains is measurement, and the paper is deliberate about saying so. The architecture is designed and its failure modes are analyzed. The parameter sizes it reasons from are published standards and are exact. The migration percentages, the downtime, the performance impact, and the count of systems that cannot be migrated are not known, are not estimated here, and would come from a pilot of the kind Section 8 specifies. Until that pilot exists this is a design, and it should be read as one.

References

1. National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS 203, Aug. 2024. DOI: 10.6028/NIST.FIPS.203.
2. National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard," FIPS 204, Aug. 2024. DOI: 10.6028/NIST.FIPS.204.
3. National Institute of Standards and Technology, "Stateless Hash-Based Digital Signature Standard," FIPS 205, Aug. 2024. DOI: 10.6028/NIST.FIPS.205.
4. National Institute of Standards and Technology, "Transition to Post-Quantum Cryptography Standards," NIST IR 8547, 2024.
5. National Cybersecurity Center of Excellence, "Migration to Post-Quantum Cryptography," NIST SP 1800-38, Volumes A-C.
6. P. W. Shor, "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer," SIAM Journal on Computing, vol. 26, no. 5, pp. 1484-1509, 1997. DOI: 10.1137/S0097539795293172.
7. L. K. Grover, "A Fast Quantum Mechanical Algorithm for Database Search," in Proc. 28th Annual ACM Symposium on Theory of Computing (STOC), 1996,

pp. 212-219. DOI: 10.1145/237814.237866.

8. N. Rameshan and G. Messmer, "An Assessment Framework for Application-Level Cryptographic Agility," arXiv:2606.13425, 2026.

9. N. Rameshan and G. Messmer, "Intent-Based Cryptographic API Design for Cryptographic Agility," arXiv:2606.13445, 2026.

10. H. Balaji, A. Varshney, P. Ravi, S. Jain, R. Foe, J. Seet, H. Wang, K.-Y. Lam, and A. Chattopadhyay, "Operationalising Post Quantum TLS: Automated Configuration Profiling and Hybrid PQC Deployment in Financial Infrastructure," arXiv:2605.17955, 2026

11. Jaikissoon, "Hybrid Horizons: Policy for Post-Quantum Security," arXiv:2510.02317, 2025.

12. R. Chandramouli and Z. Butcher, "A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments," NIST SP 800-207A, Sep. 2023. DOI: 10.6028/NIST.SP.800-207A.

13. S. T. Avirneni, "Identity Control Plane: The Unifying Layer for Zero Trust Infrastructure," arXiv:2504.17759, 2025.

14. K. Tallam, "Authorization Propagation in Multi-Agent AI Systems: Identity Governance as Infrastructure," arXiv:2605.05440, 2026.

15. OWASP Foundation, "OWASP Top 10 Non-Human Identity Risks," 2025.

16. E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, Aug. 2018. DOI: 10.17487/RFC8446.

17. D. Fett, B. Campbell, J. Bradley, T. Lodderstedt, M. Jones, and D. Waite, "OAuth 2.0 Demonstrating Proof of Possession (DPoP)," RFC 9449, Sep. 2023. DOI: 10.17487/RFC9449.