International Journal of Computers and Informatics (IJCI)
Vol. (5), No. (11)
IJCI
November 2026
المجلة الدولية للحاسبات والمعلوماتية
الإصدار (5)، العدد (11)

Post-Quantum Lattice-Based Cryptographic Schemes for Long-Term Data-at-Rest Protection in Enterprise Database Systems

Haider H. Razzaq
Computer Department, College of Science, Mustansiriyah University, Baghdad, Iraq
haideritsec@uomustansiriyah.edu.iq

Abstract

Enterprise databases routinely hold records — financial ledgers, electronic health records, genomic biobank entries, government archives — whose confidentiality must survive for decades, yet the asymmetric key-wrapping operations that protect them today (RSA and elliptic-curve key exchange inside transparent data encryption, TDE) are efficiently broken by a sufficiently large quantum computer running Shor's algorithm. Published resource estimates for that attack have fallen roughly three orders of magnitude in thirteen years — from approximately 10^9 physical qubits in 2012 to under 10^6 in 2025 — and the harvest-now-decrypt-later (HNDL) threat means an adversary can exfiltrate encrypted backups today and decrypt them once such a machine exists. A record's effective confidentiality horizon is therefore bounded by whichever comes first: its retention obligation, or the arrival of a cryptographically relevant quantum computer (CRQC). This paper studies that boundary at the key-management layer specifically, since bulk symmetric encryption (AES-256) already resists Grover's algorithm and is out of scope. We present a lattice-based post-quantum envelope-encryption architecture that replaces the RSA/ECC key-wrapping tier of a conventional TDE key hierarchy with a hybrid ML-KEM/classical key-encapsulation combiner, authenticates key-rotation metadata with ML-DSA or FN-DSA signatures over a hash-chained audit log, and negotiates between classical-only, hybrid, and pure post-quantum modes under policy control. The design is grounded in the Module-LWE and NTRU-lattice hardness assumptions behind NIST's finalized ML-KEM (FIPS 203) and ML-DSA (FIPS 204) and the still-draft FN-DSA (FIPS 206); we give seven explicit algorithms — hybrid encapsulation and decapsulation, enterprise-scale rekeying, signed audit and verification, crypto-agility negotiation, and exposure-ranked migration scheduling — with a composed security argument and an analytical cost model calibrated against published lattice-scheme parameters. That model puts the hybrid combiner's overhead at a modeled 9.0% added latency per key-rewrap relative to an all-classical baseline (12% at the highest NIST security level); shows full-estate rekeying of ten million keys falling from over ten hours at one worker to under ninety seconds at 512 parallel workers; and projects a signed, ML-DSA-authenticated ten-year audit trail for one thousand tenants at roughly 400 MB — about thirteen times an ECDSA baseline — a gap FN-DSA's compact, still-draft signatures narrow to roughly 100 MB. Four illustrative case studies — a 30-year financial ledger, a 50-year health/device archive, a 60-year genomic biobank, and a multinational deployment under CNSA 2.0 and EU deadlines — show migration completing in well under an hour of modeled compute for every estate, yet the two longest-horizon records would carry 41 to 57 years of residual

exposure under Mosca's inequality if that migration were skipped: the key-management tier, not the migration's computational cost, is the binding constraint. The paper closes by positioning the framework against current mandates (NIST, NSA CNSA 2.0, the 2026 U.S. federal schedule, and the EU roadmap) and naming the engineering work — constant-time FN-DSA, HSM firmware support, and empirical validation — that separates this analytical architecture from a production deployment.

Keywords: Post-Quantum Cryptography, Lattice-Based Cryptography, ML-KEM, ML-DSA, FN-DSA, Transparent Data Encryption, Key Management, Envelope Encryption, Harvest-Now-Decrypt-Later, Crypto-Agility, Enterprise Databases.

1. Introduction

1.1 Background and Motivation:

Modern enterprise database platforms protect data at rest through transparent data encryption (TDE): every tablespace, column, or backup is encrypted under a symmetric data-encryption key (DEK), and that DEK is in turn wrapped by a key-encryption key (KEK) that a hardware security module (HSM) or cloud key-management service (KMS) never releases in the clear. This key hierarchy is the load-bearing structure of enterprise data-at-rest security, and today it is built almost universally on RSA or elliptic-curve (ECC) public-key operations for key wrapping, key exchange with the HSM, and the digital signatures that authenticate key-rotation events.

Shor's algorithm solves the integer-factorization and discrete-logarithm problems underlying RSA and ECC in polynomial time on a sufficiently large, fault-tolerant quantum computer **[1]**, and symmetric primitives such as AES-256 lose only half of their effective key length to Grover's quadratic search speed-up — a manageable, already-budgeted-for degradation rather than a break **[2]**. The practical consequence is asymmetric: the DEK-protects-the-data layer of a TDE deployment is essentially already quantum-resistant, while the KEK-wrapping and key-rotation-signing layer is not. A framework that ignores this asymmetry and treats “post-quantum data-at-rest protection” as a bulk re-encryption problem misdiagnoses where the actual exposure lies; the work in this paper is scoped deliberately to the key-management tier.

The urgency of that exposure is driven by the harvest-now-decrypt-later (HNDL) pattern: an adversary who can passively record encrypted backups, replication streams, or key-wrapping ciphertext today needs no quantum computer at capture time — only at decryption time, whenever a CRQC eventually exists **[3]**. For data with a short confidentiality requirement this is a manageable, distant risk. For data with a multi-decade retention or sensitivity horizon — financial records held under statutory retention, health and genomic records whose relevance can outlive the patient, government records under long declassification schedules — the arrival of a CRQC may fall well inside the period during which the data must remain confidential, even though CRQC timelines themselves remain genuinely uncertain. Fig. (1) illustrates the resulting exposure window for a representative 25-year record.

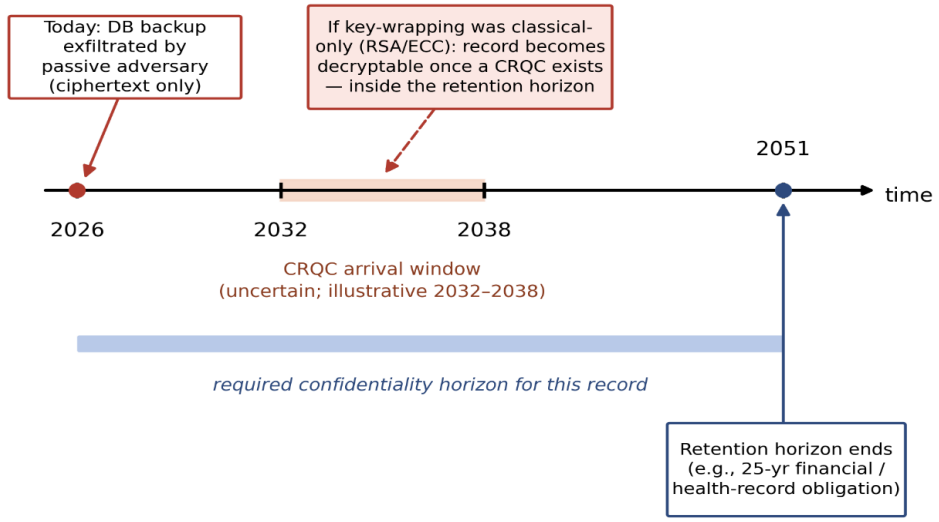


Fig. (1). Harvest-now-decrypt-later: an adversary who exfiltrates ciphertext today needs the quantum computer only at decryption time, which can fall inside a long-retention record's confidentiality horizon.

NIST finalized its first three post-quantum cryptography standards in August 2024: ML-KEM for key encapsulation (FIPS 203, standardizing CRYSTALS-Kyber) **[4]**, ML-DSA for digital signatures (FIPS 204, standardizing CRYSTALS-Dilithium) **[5]**, and SLH-DSA, a hash-based signature scheme included as a structurally independent fallback (FIPS 205) **[6]**. A fourth, compact lattice-based signature scheme, FN-DSA (standardizing Falcon), remains in draft as FIPS 206 at the time of writing, with finalization expected in late 2026 or early 2027 **[7]**. All three finalized algorithms and the draft fourth are lattice-based, resting on the Module-Learning-With-Errors (Module-LWE) and NTRU-lattice hardness assumptions rather than on the code-based, hash-based, or multivariate constructions that make up the rest of the post-quantum literature; this paper works exclusively within that lattice-based family, both because it is what NIST has actually standardized for key encapsulation and general-purpose signing, and because its performance and key-size profile is the most compatible with the latency and storage budgets of an operational TDE deployment.

1.2 Problem Statement:

Given an enterprise database deployment whose TDE key hierarchy currently wraps data-encryption keys under RSA- or ECC-protected key-encryption keys, and given data whose required confidentiality horizon may extend beyond the plausible arrival window of a cryptographically relevant quantum computer, this paper asks: how should the key-management layer of that hierarchy be redesigned so that (i) the confidentiality of already-wrapped keys does not depend on the hardness of factoring or discrete logarithm once a CRQC exists, (ii) the migration can proceed incrementally across a large, heterogeneous estate without a flag-day cutover, and (iii) every step of the redesign is accountable to an explicit, auditable trail rather than a silent algorithm swap. Section 3 develops the

system, threat, and cryptographic model this problem statement presupposes; Section 3.4 presents the proposed architecture.

1.3 Research Contributions:

This paper makes five contributions. First, it isolates the key-management tier of enterprise TDE as the correct target for post-quantum hardening, arguing that bulk symmetric protection is out of scope (§1.1, §3.3). Second, it specifies a lattice-based envelope-encryption architecture in which a hybrid ML-KEM/classical combiner protects the key-encryption key and ML-DSA or FN-DSA signatures authenticate every key-rotation event over a hash-chained audit log (§3.4). Third, it gives seven explicit algorithms — hybrid encapsulation and decapsulation, enterprise-scale rekeying, signed rotation audit and its verification, crypto-agility negotiation, and exposure-ranked migration scheduling — with a byte-level record format and usage limits for every key class (§3.4). Fourth, it develops a composed security argument grounded in Module-LWE and NTRU-lattice hardness and an analytical cost model, calibrated against published ML-KEM, ML-DSA and FN-DSA parameters, that projects key size, CPU overhead and rekeying latency from tens to tens of thousands of tenants (§4). Fifth, it tests the architecture against four case studies with different confidentiality horizons and against the regulatory landscape, including NIST's standards, CNSA 2.0 and the 2026 U.S. federal and EU mandates (§4.5, §5.2).

1.4 Scope and Non-Goals:

The paper is scoped to the confidentiality of data at rest inside an enterprise relational or document database and to the asymmetric key-management operations that protect it. It does not address data in transit (TLS key exchange is a separately standardized problem with its own hybrid literature, §2.3) or computation over encrypted data, which lattice-based homomorphic and searchable encryption address (§2.4). Bulk AES-256-GCM encryption is treated as adequate against Grover's algorithm and is not modified. The cost model is analytical, calibrated against published benchmarks rather than measured on a from-scratch implementation; §5.5 and §5.5 discuss what that does and does not limit.

1.5 Paper Organization:

Section 2 reviews the enterprise TDE/KMS landscape, the quantum threat, NIST's lattice-based standards, and hybrid and crypto-agile architectures. Section 3 presents the methodology: the system and threat model, lattice foundations and building blocks, the proposed architecture and its seven algorithms, and the security-analysis and cost-model approach. Section 4 reports results, Section 5 discusses regulatory, deployment and validity questions, and Section 6 concludes.

2. Literature Review

2.1 Enterprise Data-at-Rest Encryption Architectures:

Every major enterprise database engine implements some form of transparent data encryption. Oracle wraps tablespace and column keys under a master key in an external keystore or HSM **[27]**; SQL

Server wraps a database encryption key under a certificate or asymmetric key protected by the service master key or an external KMS **[28]**; PostgreSQL relies on extensions such as pg_tde that layer envelope encryption under the storage engine **[29]**; and MySQL/InnoDB uses a two-tier keyring architecture with a similar master-key/tablespace-key split. All follow the pattern of the classical hierarchy in Fig. (2): an HSM- or KMS-resident root key protects a key-encryption key (KEK) through RSA or ECC wrapping, and each KEK protects AES-256 data-encryption keys. Cloud key-management services (AWS KMS, Azure Key Vault, Google Cloud KMS) externalize the root tier but keep the same RSA/ECC-wrapped envelope pattern **[36]**, and none of these systems, as shipped at the time of writing, offers a lattice-based or hybrid alternative to that wrapping tier.

2.2 The Quantum Threat and NIST Post-Quantum Standardization:

Shor's algorithm factors integers and computes discrete logarithms in time polynomial in the input length on a fault-tolerant quantum computer, which breaks RSA, Diffie-Hellman, and elliptic-curve cryptography outright once a sufficiently large such machine exists **[1]**. Grover's algorithm gives only a quadratic speed-up for unstructured search, halving the effective security level of a symmetric cipher or hash function rather than breaking it **[2]**; this is the basis for treating AES-256 as already quantum-resistant at a 128-bit post-quantum security level. The open question is not whether a CRQC would break RSA/ECC — that is a settled matter of computational complexity — but when one will exist at sufficient scale and fidelity; timeline estimates surveyed by industry-academic consortia place meaningful probability mass across the 2030s **[32]**, which is precisely the uncertainty band this paper's case studies (§4.5) are built around rather than a single point estimate.

Resource estimates for Shor's algorithm have fallen steeply, and the trend matters as much as any single figure. Fowler et al. estimated about 10^9 physical qubits to factor a 2048-bit RSA modulus under surface-code error correction **[40]**; Gidney and Ekerå reduced this to roughly 20 million noisy qubits running about eight hours, assuming a 0.1% gate error rate and a 1 μs surface-code cycle **[38]**; and Gidney's 2025 analysis, under the same assumptions, brought it below one million qubits at under one week **[39]**. Elliptic-curve targets are cheaper still at the logical level, at about 2,330 logical qubits for a 256-bit prime-field curve **[41]**. Table 1 collects these estimates. The three-orders-of-magnitude reduction in thirteen years came from algorithmic and error-correction advances rather than hardware, which is why this paper treats CRQC arrival as a window rather than a point (§4.5); unrefereed 2026 preprints on quantum LDPC codes and neutral-atom architectures claim further reductions under different hardware models and are not directly comparable **[46]**.

Table (1). Published quantum resource estimates for the primitives relevant to a TDE key hierarchy

Target	Source	Estimated resources	Key assumptions
RSA-2048 (Shor)	Fowler et al., 2012 [40]	≈10^9 physical qubits	Surface code; earlier error-rate and hardware assumptions
RSA-2048 (Shor)	Gidney & Ekerå, 2019/2021 [38]	≈2×10^7 physical qubits; ≈8 h	0.1% gate error; 1 μs cycle; 10 μs reaction time
RSA-2048 (Shor)	Gidney, 2025 [39]	<10^6 physical qubits; <1 week	Same assumptions as 2019; approximate residue arithmetic, yoked surface codes, magic-state cultivation
ECC P-256 (ECDLP)	Roetteler et al., 2017 [41]	≈2,330 logical qubits; ≈1.3×10^11 Toffoli gates	Logical-level estimate; error-correction overhead not included
AES-256 (Grover)	Grover, 1996 [2]	≈(π/4)·2^128 serial iterations	Iterations are inherently serial; S processors give only a √S speed-up

Symmetric primitives fare differently. Exhaustive search of a k-bit key with Grover's algorithm needs about $(\pi/4)\cdot 2^{(k/2)}$ iterations of the cipher circuit, and because the iterations must run in sequence, S parallel quantum processors reduce the time only by a factor of √S,

$$N_{Grover} \approx (\pi/4)\cdot 2^{k/2}, \quad T_S \approx (\pi/4)\cdot 2^{k/2}/\sqrt{S} \tag{1}$$

so, for AES-256 the attack remains on the order of 2^128 sequential AES evaluations before any error-correction overhead, the basis for NIST's assignment of AES-256 to security category 5 and for treating the DEK-protects-data layer as already quantum-resistant. The appropriate response to Grover is therefore to size symmetric keys at 256 bits, not to replace the primitive — the opposite of the situation for RSA and ECC under Eq. (9).

NIST's standardization process, launched in 2016 **[35], [37]**, produced three FIPS standards in August 2024: ML-KEM (FIPS 203), a Module-LWE key-encapsulation mechanism standardizing CRYSTALS-Kyber **[4], [12]**; ML-DSA (FIPS 204), a Module-LWE signature standardizing CRYSTALS-Dilithium **[5], [13]**; and SLH-DSA (FIPS 205), a hash-based signature included as a conservative, structurally independent fallback **[6]**. A fourth scheme, FN-DSA (FIPS 206), standardizes the NTRU-lattice Falcon signature and remains a draft, submitted for approval in August 2025 with finalization expected in late 2026 or early 2027; its signatures are roughly four to six times smaller than ML-DSA's at comparable security, at the cost of a floating-point Gaussian sampling step that is difficult to implement in constant time **[7]**. NIST's round-three status report documents the selection rationale and parameter choices on which this paper's cost model draws **[15]**.

The hardness assumptions behind these schemes trace to Regev's Learning-With-Errors (LWE) problem, which reduces worst-case lattice problems to an average-case learning problem and thereby to a public-key cryptosystem **[8]**. Ring-LWE restricts LWE to a polynomial ring to shrink key sizes **[9]**, and Module-LWE generalizes Ring-LWE with a module rank parameter that lets a single algebraic structure span multiple security levels — the choice ML-KEM and ML-DSA both make **[10]**. NTRU lattices, defined independently and considerably earlier by Hoffstein, Pipher, and

Silverman **[11]**, underlie FN-DSA/Falcon instead. Section 3.2 develops the specific hardness definitions this paper relies on.

2.3 Post-Quantum Key Management, Migration, and Crypto-Agile Architectures:

NIST SP 800-57 predates the PQC effort and does not natively address hybrid or crypto-agile hierarchies, though its key-lifecycle framework is the natural scaffold for a post-quantum key hierarchy **[24]**. Draft IR 8547 sets a civilian federal timeline that deprecates quantum-vulnerable public-key algorithms after 2030 and disallows them after 2035 **[19]**, and the Cloud Security Alliance's 2026 note documents the summer-2026 shift from advisory roadmaps to binding procurement and compliance mandates **[34]**, which §5.2 treats as the immediate driver for enterprise planning. Industry analysis of FN-DSA underscores that standardization and safe constant-time implementation are separate milestones **[33]**.

Because confidence in any single post-quantum assumption is necessarily lower than the decades of scrutiny given to RSA and ECC, the key-exchange literature converged on hybrid constructions that remain secure if either component holds **[21]**. Bindel et al. formalize hybrid KEM combiners that stay IND-CCA2-secure so long as one constituent is secure **[22]**, the result on which this paper's combiner (§3.3.4) and security argument (§4.1) build. The IETF's hybrid key-exchange design for TLS 1.3 applies the same idea to a widely deployed protocol **[23]**; it is supportive precedent for the key-wrapping layer of TDE, though key wrapping is a long-lived at-rest operation rather than a per-session handshake.

2.4 Alternative Post-Quantum Families and Related Constructions:

NIST's portfolio also includes code-based (Classic McEliece) and hash-based (SLH-DSA) schemes; the most prominent isogeny candidate, SIKE, was broken by classical cryptanalysis during standardization. Table (2) compares the families on the dimensions that matter for key wrapping: key and ciphertext size, computational cost and standardization maturity. Lattice schemes are not best on every axis — SLH-DSA offers a structurally independent security argument at the cost of much larger signatures — but they are the only family that is NIST-finalized for both key encapsulation and general-purpose signing while remaining competitive in size with the RSA/ECC keys they replace, which is why the proposed architecture is restricted to lattice-based schemes.

Table (2). Post-quantum cryptographic families compared for enterprise key-wrapping suitability

Family	NIST status (2026)	Representative scheme	Key/ct size	Suitability here
Lattice (LWE/NTRU)	ML-KEM, ML-DSA final; FN-DSA draft	Kyber, Dilithium, Falcon	Small–moderate	Primary choice (§3.2–§3.3)
Code-based	Classic McEliece: alternate KEM candidate	McEliece	Very large public key (100s of KB)	Poor fit: key size too large for KEK rotation at scale

Family	NIST status (2026)	Representative scheme	Key/ct size	Suitability here
Hash-based	SLH-DSA final (FIPS 205)	SPHINCS+	Small key, large/slow signature	Conservative fallback signer only (§5.3)
Multivariate	No NIST-finalized general-purpose scheme	Rainbow (broken)	Varies	Excluded: unresolved cryptanalytic history
Isogeny-based	SIKE candidate broken (2022)	SIKE	Small	Excluded: broken by classical cryptanalysis

Lattice hardness also underlies fully homomorphic encryption schemes such as BFV **[30]** and CKKS **[31]**, which permit computation over ciphertext, and a growing literature on lattice-based searchable encryption. These are often conflated with post-quantum data-at-rest protection because they share LWE-family assumptions, but they solve a different problem — confidential computation over data that must remain queryable, at substantial overhead — rather than confidential storage of data that is decrypted locally before use. The proposed architecture neither requires nor provides homomorphic properties; §5.4 returns to them only as a future direction.

2.5 Summary and Research Gap:

Table (3)positions the present work against the enterprise TDE/KMS landscape of §2.1 and the hybrid/migration literature of §2.3. Commercial TDE and KMS products protect data at rest but wrap keys exclusively under classical RSA/ECC operations; general PQC and hybrid key-exchange literature establishes the cryptographic building blocks and their combiner security but has not been worked through as an explicit, algorithmic redesign of an enterprise key hierarchy with a quantified rekeying cost at scale. The gap this paper addresses is precisely that translation: taking NIST-finalized lattice primitives and the hybrid-combiner security literature and turning them into explicit key-hierarchy algorithms, a composed security argument, and a cost model that an enterprise architect could use to plan a migration.

Table (3). Existing enterprise TDE/KMS approaches compared with the proposed framework

Approach	Key-wrap algorithm	Quantum-safe?	Key limitation
Oracle TDE / SQL Server TDE	RSA / ECC	No	No PQC key-wrap option shipped
PostgreSQL pg_tde	RSA / ECC via external KMS	No	Depends entirely on KMS-side algorithm
Cloud KMS (AWS/Azure/GCP)	RSA / ECC (some offer symmetric-only envelopes)	No	No customer-facing hybrid/PQC wrap mode
Hybrid TLS key exchange [21]–[23]	Classical + ML-KEM	Yes (session)	Solves transport, not at-rest key hierarchy
This framework	Hybrid classical + ML-KEM; ML-DSA/FN-DSA signed audit	Yes (key-management tier)	Analytical cost model, not yet empirically measured

3. Methodology

3.1 System Model, Key Hierarchy, and Threat Model:

The system under study is a single enterprise database instance or a small cluster of replicas, each storing one or more tenants' data across tablespaces or column-level encrypted stores, protected by the classical TDE key hierarchy of Fig. (2) an HSM- or KMS-resident root key protects one key-encryption key (KEK) per tenant or tablespace via RSA or ECC wrapping, and each KEK protects one or more AES-256 data-encryption keys (DEKs). The database engine, the KMS/HSM, and an append-only audit log are modeled as three cooperating but not fully mutually trusting components (Fig. 7); the database engine requests key operations but never sees unwrapped KEK material, the KMS/HSM performs wrapping/unwrapping and signs rotation metadata but never sees plaintext row data, and the audit log accepts signed, hash-chained append operations but cannot itself authorize a key change.

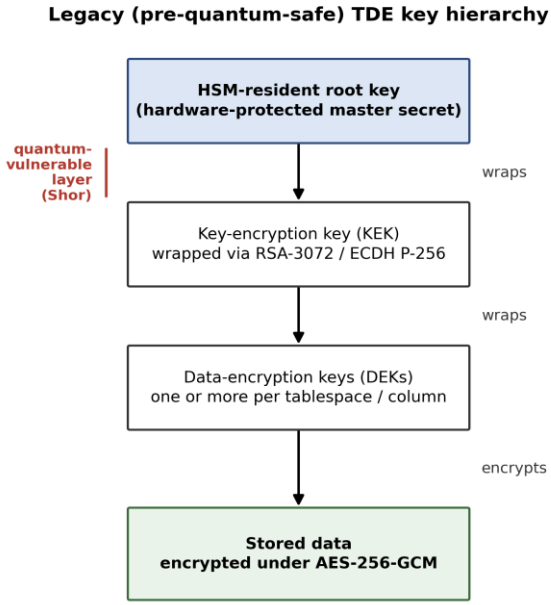


Fig. (2). The classical (pre-quantum-safe) TDE key hierarchy implemented, with minor variation, by every major enterprise database engine and cloud KMS.

Fig. (3) enumerates the threat actors this paper considers. A passive network eavesdropper can record TLS-protected traffic or replication streams today and attempt decryption later; a cloud storage or backup operator can read encrypted backups and snapshots at rest but not the keys that protect them; a malicious or compromised database administrator has query-level access to plaintext views through the database engine's normal access-control path but not direct access to raw HSM-resident key material; and — the threat this paper is organized around — a future cryptographically relevant quantum computer can retroactively break any RSA- or ECC-wrapped key material harvested by the

first two actors. The framework assumes the HSM/KMS itself is not compromised (a hardware root of trust is out of scope to re-derive) and that TLS or an equivalent channel protects data in transit; §3.3 governs only what protects a key once it is wrapped for storage or transmitted between the database engine and the KMS.

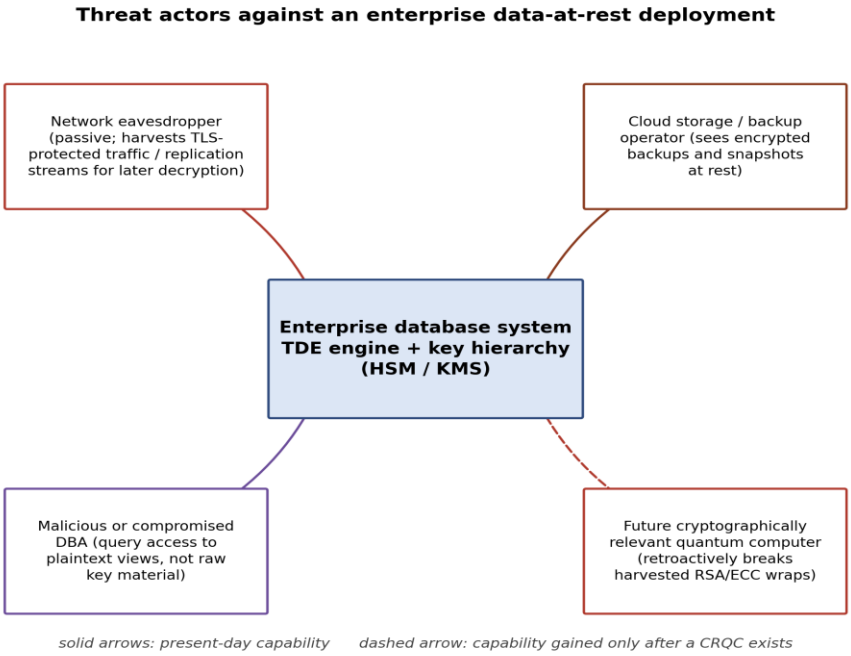


Fig. (3). Threat actors considered against an enterprise data-at-rest deployment. Only the future-CRQC actor's capability is not already present today.

3.2 Design Goals and Lattice Hardness Foundations:

Four requirements shape the proposed architecture. First, backward compatibility during migration: an enterprise estate with millions of existing RSA/ECC-wrapped KEKs cannot be re-keyed atomically, so the architecture must support incremental, per-tenant migration with both algorithm families active simultaneously. Second, no single point of cryptographic failure: because confidence in any one post-quantum hardness assumption is necessarily younger than decades of cryptanalysis on RSA/ECC, the default operating mode combines both rather than replacing one with the other outright (§3.3.4). Third, auditable key provenance: every key-rotation event must be signed and hash-chained so that a change to the active KEK is independently verifiable, not merely logged. Fourth, bounded operational overhead: the added latency and storage cost of hybrid or pure post-quantum operation must be small relative to existing TDE overhead at the tenant counts and rekeying frequencies enterprise deployments actually see (quantified in §4.2–§4.4).

The Learning-With-Errors (LWE) problem asks a solver to distinguish, given a uniformly random matrix A and a vector of noisy linear equations in a secret s,

$$b = As + e \pmod{q} \tag{2}$$

the resulting b from a uniformly random vector u of the same dimension, where e is drawn from a narrow error distribution and all arithmetic is modulo q. Regev showed that solving this decision problem on average is at least as hard as solving several worst-case lattice problems (the shortest-vector and shortest-independent-vectors problems) in the worst case **[8]**, giving the security guarantee its usual asymptotic form,

$$|\Pr[D(A,b) = 1] - \Pr[D(A,u) = 1]| \leq \varepsilon_{\mathrm{LWE}} \tag{3}$$

where the distinguishing advantage of any efficient adversary against decision-LWE is bounded by a negligible ε_LWE. Plain LWE's public matrix A grows quadratically with the security parameter, which is impractical for a real key-encapsulation mechanism; Ring-LWE restricts the same problem to a polynomial ring $\mathbb{Z}_x[x]/(x^n+1)$, so that a single ring element replaces an entire matrix,

$$b(x) = a(x)\cdot s(x) + e(x) \pmod{q, x^n+1} \tag{4}$$

shrinking keys from quadratic to quasi-linear in the security parameter at the cost of relying on the extra algebraic structure of the ring rather than a fully generic lattice **[9]**. Module-LWE generalizes Ring-LWE with a module rank k that lets one ring dimension n serve several security levels by varying k instead of n — the design choice both ML-KEM and ML-DSA make —

$$\mathrm{b} = \mathrm{As} + \mathrm{e} \pmod{q}, \quad \mathrm{A} \in R_q^{k\times k} \tag{5}$$

so that ML-KEM-512, -768, and -1024 (§3.3.1) share a single ring dimension and differ only in k **[10]**. FN-DSA instead builds on NTRU lattices, defined by a public ring element h that is the ratio of two secret, small-coefficient polynomials f and g,

$$h \equiv g\cdot f^{-1} \pmod{q}, \quad f, g \in \mathrm{Z}[x]/(x^n+1) \tag{6}$$

a structure introduced well before the LWE framework **[11]** and independently hard under the NTRU and, for the associated signature scheme, the closely related Short-Integer-Solution-over-NTRU-lattices assumption. Across all three constructions, concrete security level scales approximately with the lattice dimension n, the modulus q, and the achievable root Hermite factor δ_0 of the best known lattice-reduction attack,

$$\lambda \approx (n \log_2 q) / (\log_2(1/\delta_0))\cdot c \tag{7}$$

which is why NIST's five security categories are realized, within each scheme family, purely by varying (n, k, q) rather than by changing the underlying algebraic construction — the parameter sets in Table 4 and Table 6 are exactly this scaling in practice. For contrast, Grover's algorithm against a symmetric cipher of classical key length k reduces the effective post-quantum security level by exactly a factor of two,

$$k_{\text{eff}}^{Q} = k_{\text{classical}} / 2 \tag{8}$$

which is the formal statement behind treating AES-256 (k_eff = 128) as adequately quantum-resistant (§1.1), while Shor's algorithm's polynomial running time against RSA,

$$T_{\text{Shor}}(N) = \text{O}((\log N)^2(\log\log N)(\log\log\log N)) \tag{9}$$

is a qualitative break rather than a quantitative security-level reduction — the asymmetry between Eq. (8) and Eq. (9) is the formal version of the argument in §1.1 for scoping this paper to the key-wrapping tier alone.

3.2.1 Geometry of the underlying problems. A lattice is the set of integer combinations of a basis. Given a short, nearly orthogonal basis, the shortest-vector problem (SVP) and closest-vector problem (CVP) are solved by inspection or simple rounding; given a long, skewed basis of the very same lattice they appear hard, and recovering a good basis from a bad one is exactly what lattice-reduction algorithms attempt. LWE-based schemes publish a bad description of the lattice (the noisy equations) and keep a good one (the secret) hidden; NTRU-based schemes publish a bad basis of an NTRU lattice and keep the short pair (f, g) as a trapdoor. In dimension two the distinction is trivial. In the dimensions used here — n·k = 512 to 1,024 for ML-KEM — no known algorithm recovers a good basis faster than the exponential cost quantified next.

3.2.2 Concrete cryptanalysis. Asymptotic reductions do not fix parameters; concrete security is estimated from the cost of lattice reduction. The BKZ algorithm with block size β returns a basis whose first vector is shorter than the shortest-vector bound by a root Hermite factor

$$\delta_0(\beta) = ((\beta/2\pi e)\cdot(\pi\beta)^{1/\beta})^{1/(2(\beta-1))} \tag{10}$$

In the primal (unique-SVP) attack on an LWE instance with n secret coordinates, m samples and error standard deviation σ, reduction succeeds when **[43]**

$$\sqrt{\beta}\cdot\sigma \le \delta_0^{2\beta-d-1}\cdot q^{m/d}, \quad d = n + m + 1 \tag{11}$$

which fixes the smallest workable β for each parameter set. The cost of BKZ-β is dominated by calls to an SVP oracle in dimension β, and the core-SVP methodology charges a single such call, giving classical and quantum sieving costs of **[12], [44]**

$$C_{\text{SVP}}^{\text{cl}}(\beta) = 2^{0.292\beta}, \quad C_{\text{SVP}}^{q}(\beta) = 2^{0.265\beta} \tag{12}$$

The resulting block sizes β ≈ 404, 627 and 877 for ML-KEM-512, -768 and -1024 correspond to classical core-SVP costs of about 2^118, 2^183 and 2^256 (Fig. 4). Two caveats matter. First, core-SVP is a deliberately conservative lower bound — it ignores the many SVP calls in a BKZ run, memory-access costs and quantum-circuit overheads — and NIST's category assignments rest on more detailed gate-count analyses **[15]**. Second, ML-KEM-512's estimate of 2^118 sits below the nominal 2^128 AES-128 reference line of Fig. 4; continuing refinement of lattice attacks, notably of dual attacks, has narrowed the margin at this parameter set without changing the category assignments

[15]. This is one reason the framework defaults to ML-KEM-768 and reserves ML-KEM-1024 for confidentiality horizons beyond roughly 40 years (§4.5).

3.2.3 Polynomial arithmetic and the NTT. The efficiency that makes ML-KEM and ML-DSA practical inside a KMS comes from the ring R_q = Z_q[x]/(x^256 + 1). ML-KEM uses q = 3329 and ML-DSA uses q = 8380417, both with n = 256 **[4], [5]**. Multiplication in R_q costs O(n log n) with the number-theoretic transform (NTT) instead of O(n^2). Because 256 divides q − 1 = 3328 = 13·256 but 512 does not, ML-KEM's NTT is incomplete: x^256 + 1 splits only into 128 quadratic factors (with ζ = 17 a primitive 256th root of unity),

$$R_q \simeq \prod_{i=0}^{127} Z_q[x]/(x^2 - \zeta^{2\cdot \mathrm{br}(i)+1}) \tag{13}$$

and products are computed pairwise in the quadratic residue rings. ML-DSA's modulus satisfies q ≡ 1 (mod 512), which admits a complete NTT to degree-zero factors. These transforms, together with the centered-binomial sampling and compression of §3.3, account for nearly all of the arithmetic in a decapsulation — and, as §4.1 discusses, for most of its side-channel surface.

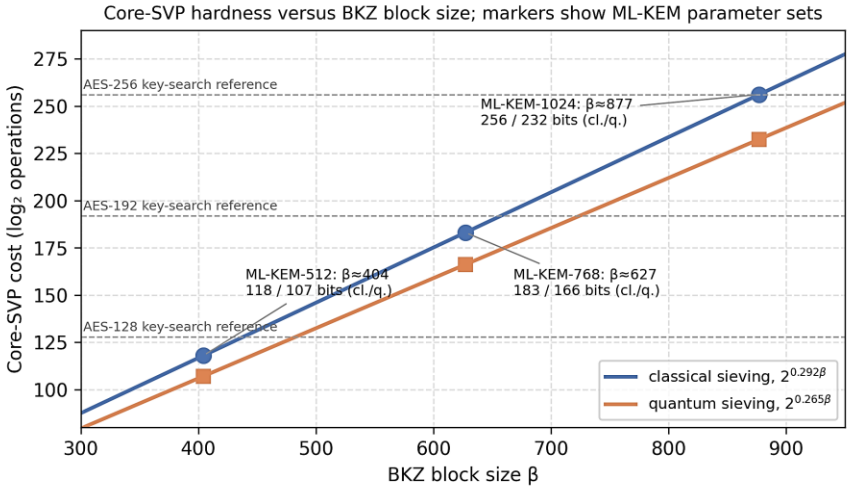


Fig. (4). Core-SVP cost as a function of BKZ block size under classical (0.292β) and quantum (0.265β) sieving exponents, with the three ML-KEM parameter sets marked.

3.3 Lattice-Based Cryptographic Building Blocks:

3.3.1 ML-KEM. ML-KEM (FIPS 203) is an IND-CCA2-secure key-encapsulation mechanism built from a Module-LWE public-key encryption scheme via a Fujisaki-Okamoto-style transform **[4], [12]**. Encapsulation against a public key produces both a ciphertext and a shared secret,

$$(K, ct_{\mathrm{pqc}}) \leftarrow \mathrm{Encaps}(pk_{\mathrm{pqc}}) \tag{14}$$

and decapsulation recovers the same shared secret from the ciphertext and the matching secret key, falling back to a pseudorandom (but deterministic and indistinguishable-from-valid) value under

implicit rejection if the ciphertext fails re-encryption verification, which is what gives ML-KEM its CCA2 security rather than merely CPA security,

$$K' = \text{Decaps}(sk_{pqc}, ct_{pqc}) \quad (15)$$

Table (4) lists the three standardized parameter sets. ML-KEM-768 is NIST's recommended default for most applications and is the parameter set this paper's baseline cost-model results (§4.2–§4.4) use unless otherwise stated.

Table (4). ML-KEM (FIPS 203) parameter sets

Parameter set	NIST category	Public key	Ciphertext	Secret key
ML-KEM-512	1 (≈AES-128)	800 B	768 B	1,632 B
ML-KEM-768	3 (≈AES-192)	1,184 B	1,088 B	2,400 B
ML-KEM-1024	5 (≈AES-256)	1,568 B	1,568 B	3,168 B

Internally, ML-KEM composes a Module-LWE public-key encryption scheme (K-PKE) with the Fujisaki–Okamoto (FO) transform, as Fig. (5) shows. Key generation expands a public matrix A ∈ R_q^(k×k) from a seed ρ, samples a secret s and error e from a centered binomial distribution CBD with parameter η_1, and publishes t = As + e. Encapsulation draws a random 32-byte message m, derives the shared secret K and the encryption randomness r from G(m ‖ H(pk)), and emits a ciphertext (u, v) whose coefficients are compressed to d_u and d_v bits,

$$\text{Compress}_d(x) = \lfloor (2^d/q)x + 1/2 \rfloor \pmod{2^d} \quad (16)$$

which is why an ML-KEM ciphertext is far smaller than an uncompressed vector of ring elements. Compression adds rounding noise that is absorbed into the decryption error budget, so the decryption-failure probability

$$\delta_{\text{fail}} = \Pr[\|\text{noise}\|_\infty \geq q/4] \approx 2^{-164} \quad (k = 3) \quad (17)$$

is a design parameter rather than a defect: the standardized sets achieve 2^-139, 2^-164 and 2^-174 for k = 2, 3 and 4 **[4]**. Decapsulation re-encrypts the recovered message and compares the result with the received ciphertext; on mismatch it returns a pseudorandom value keyed by the implicit-rejection secret z,

$$K = K' \text{ if } c = c', \quad K = \text{J}(z \parallel c) \text{ otherwise} \quad (18)$$

so an attacker who submits malformed ciphertexts to an HSM learns nothing from success or failure. Algorithm (5) preserves this property at the hybrid layer. Table (5) lists the internal parameters behind the sizes of Table (4).

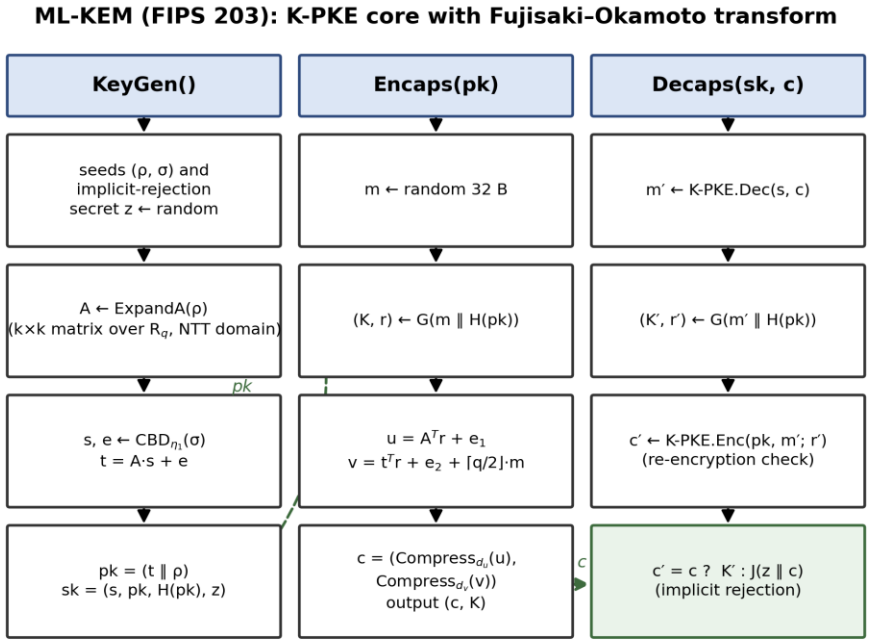


Fig. (5). ML-KEM structure: an IND-CPA Module-LWE encryption core (K-PKE) wrapped by the Fujisaki–Okamoto transform, whose re-encryption check gives IND-CCA2 security and implicit rejection.

Table (5). ML-KEM internal parameters (q = 3329, n = 256 throughout)

Parameter	ML-KEM-512	ML-KEM-768	ML-KEM-1024
Module rank k	2	3	4
Noise parameters (η_1, η_2)	(3, 2)	(2, 2)	(2, 2)
Compression bits (d_u, d_v)	(10, 4)	(10, 4)	(11, 5)
Public key = 384k + 32 (B)	800	1,184	1,568
Ciphertext = 32(d_u·k + d_v) (B)	768	1,088	1,568
Secret key = 768k + 96 (B)	1,632	2,400	3,168
Decryption-failure probability	2^-139	2^-164	2^-174
Core-SVP, classical / quantum (bits)	118 / 107	183 / 166	256 / 232
NIST security category	1	3	5

3.3.2 ML-DSA and FN-DSA. ML-DSA (FIPS 204) is a Module-LWE/Module-SIS signature scheme built via the Fiat-Shamir-with-aborts paradigm **[5], [13]**; FN-DSA (FIPS 206, draft) is an NTRU-lattice signature built via a hash-and-sign paradigm over a trapdoor sampler, standardizing Falcon **[7], [14]**. Table (6) compares their parameter sets: ML-DSA is finalized and has broad library and hardware support, while FN-DSA's much smaller signatures make it attractive for exactly the certificate-chain and audit-log overhead this paper's key-rotation design accumulates over time (§3.4.3, §4.4), at the cost of a signing procedure whose floating-point Gaussian sampling is genuinely difficult to implement safely in constant time — the reason this paper treats FN-DSA as available but not yet the default (§5.3).

Table (6). ML-DSA (FIPS 204, final) and FN-DSA (FIPS 206, draft) parameter sets

Scheme	NIST category	Status (2026)	Public key	Signature
ML-DSA-44	2	Final	1,312 B	2,420 B
ML-DSA-65	3	Final	1,952 B	3,309 B
ML-DSA-87	5	Final	2,592 B	4,627 B
FN-DSA-512	1	Draft (→FIPS 206)	897 B	≈666 B
FN-DSA-1024	5	Draft (→FIPS 206)	1,793 B	≈1,280 B

ML-DSA follows the Fiat–Shamir-with-aborts paradigm: the signer samples a masking vector y with coefficients in $(-\gamma_1, \gamma_1]$, computes $w = Ay$, derives a sparse challenge c with τ coefficients in $\{-1, +1\}$ by hashing the message digest with the high bits of w, and releases $z = y + cs_1$ only if the result cannot leak the secret, that is, only if **[5], [13]**

$$\|\mathbf{z}\|_\infty < \gamma_1 - \beta \quad \text{and} \quad \|\mathbf{r}_0\|_\infty < \gamma_2 - \beta, \quad \beta = \tau\eta \tag{19}$$

where $\beta = \tau\eta$ bounds the size of cs_1; otherwise the attempt is discarded and resampled. The expected number of attempts,

$$E[\#\text{attempts}] = 1/\Pr[\text{accept}] \in \{4.25, 5.1, 3.85\} \tag{20}$$

makes signing latency variable while verification costs a fixed amount. For the audit log of §3.4.3 the asymmetry is favorable: signing happens once per rotation event, whereas verification recurs at every audit. Table (7) lists the internal parameters. FN-DSA differs structurally — hash-and-sign over an NTRU trapdoor with n = 512 or 1,024 and q = 12,289, using a fast-Fourier Gaussian sampler **[14]** — and that sampler's floating-point arithmetic is what makes constant-time implementation difficult (§5.3).

Table (7). ML-DSA internal parameters (q = 8,380,417, n = 256, d = 13 dropped bits throughout)

Parameter	ML-DSA-44	ML-DSA-65	ML-DSA-87
Matrix dimensions (k, l)	(4, 4)	(6, 5)	(8, 7)
Secret coefficient bound η	2	4	2
Challenge weight τ	39	49	60
$\beta = \tau\eta$	78	196	120
Mask range γ_1	2^17	2^19	2^19
Low-order rounding range γ_2	(q−1)/88	(q−1)/32	(q−1)/32
Hint weight bound ω	80	55	75
Expected signing attempts	4.25	5.1	3.85
Public key / signature (B)	1,312 / 2,420	1,952 / 3,309	2,592 / 4,627

3.3.3 Notation. Table (8) fixes the notation used from this point forward.

Table (8). Notation summary

Symbol	Meaning
KEK, DEK	Key-encryption key; data-encryption key
pk_c, pk_p / sk_c, sk_p	Classical / post-quantum public and secret keys
ct_c, ct_p	Classical key-exchange value; ML-KEM ciphertext
K_hyb	Combined hybrid shared secret

Symbol	Meaning
n, k, q	Lattice dimension, module rank, modulus
λ	Target statistical/computational security parameter
N, P	Number of DEKs/KEKs to rekey; parallel workers
H_req	Required confidentiality horizon of a record (Eq. 38)
X, Y, Z, E	Confidentiality lifetime; migration duration; time to CRQC; residual exposure (years), Eq. 28
W	Maintenance window (s), Eq. 34
t, t_harvest, t_migration	Calendar time; harvest time; migration-completion time

3.3.4 Hybrid KEM combiner. Following the construction and security treatment of Bindel et al. **[22]**, the framework never relies on ML-KEM alone during the migration window; it combines a classical Diffie-Hellman exchange (ECDH over P-384, chosen for its larger classical margin than P-256 given the multi-decade horizons in §4.5) with ML-KEM through a key-derivation function applied to both shared secrets and both ciphertexts,

$$K_{\text{hyb}} = \text{KDF}(K_{\text{classical}} \parallel K_{\text{pqc}} \parallel ct_c \parallel ct_p) \tag{21}$$

binding the transcript (both ciphertexts) into the derivation so that an adversary who tampers with either ciphertext independently cannot cause the two parties to derive matching keys from mismatched inputs. Algorithm 1 specifies this combiner precisely; §4.1 gives the corresponding security argument.

3.3.5 Combiner design details. Three choices in Algorithm (1) carry security weight. First, the KDF input contains both ciphertexts as well as both shared secrets; binding the full transcript defeats mix-and-match attacks in which an adversary substitutes one component's ciphertext, a concern analyzed for generic hybrid KEM combiners in **[22]**. Second, the KDF is domain-separated by a context string containing algorithm identifiers, tenant identifier and record version, so a K_hyb derived for one KEK wrap can never be valid for another. Third, the classical share is ECDH over P-384: it is expected to fall first once a CRQC exists, but during migration it hedges against an unforeseen lattice attack at a cost of only the roughly 140 µs of Fig. 18. The IETF's TLS 1.3 hybrid designs use the same construction **[23]**. The combiner is a commercial-sector hedge: national-security deployments under CNSA 2.0 move to ML-KEM-1024 alone on the schedule of §5.2 **[16]**.

3.4 Proposed Framework: Post-Quantum Envelope-Encryption Architecture:

Fig. (6) shows the redesigned key hierarchy. The HSM-resident root key is unchanged; it now protects a hybrid KEM combiner rather than a bare RSA/ECC key-wrapping step. The combiner's output wraps the KEK, and every wrap is accompanied by an ML-DSA (or, once available, FN-DSA) signature over the wrap's metadata — algorithm identifiers, key identifiers, and a timestamp — rather than over the KEK material itself, so that the signature can be verified without ever reconstructing the unwrapped key outside the HSM boundary. The KEK continues to wrap one or more DEKs exactly as before, and the DEKs continue to protect AES-256-GCM-encrypted data unchanged; §1.1's

asymmetry argument is visible directly in the figure as the difference between the top two boxes (redesigned) and the bottom two (untouched).

3.4.1 Hybrid key encapsulation. Algorithm (1) specifies the combiner introduced in §3.3.4. Fig. (7) shows it embedded in the larger rekeying exchange between the database engine, the KMS/HSM, and the audit log: the engine requests a rewrap, the KMS returns its classical and post-quantum public keys, the engine (or, in the equivalent HSM-internal variant, the HSM itself) runs HybridEncapsulate, and the resulting hybrid secret wraps the KEK before a signed rotation record is appended to the audit log and verified by the engine prior to activation.

Algorithm 1 *HybridEncapsulate(pk_c, pk_p)*

```
Output: ciphertexts (ct_c, ct_p); shared secret K_hyb
1: (ct_c, K_c) ← ECDH-Encaps(pk_c)        // classical share
2: (ct_p, K_p) ← ML-KEM.Encaps(pk_p)      // lattice share
3: K_hyb ← KDF(K_c ‖ K_p ‖ ct_c ‖ ct_p)   // Eq. (21), transcript-bound
4: zeroize K_c, K_p in local memory
5: return (ct_c, ct_p, K_hyb)
```

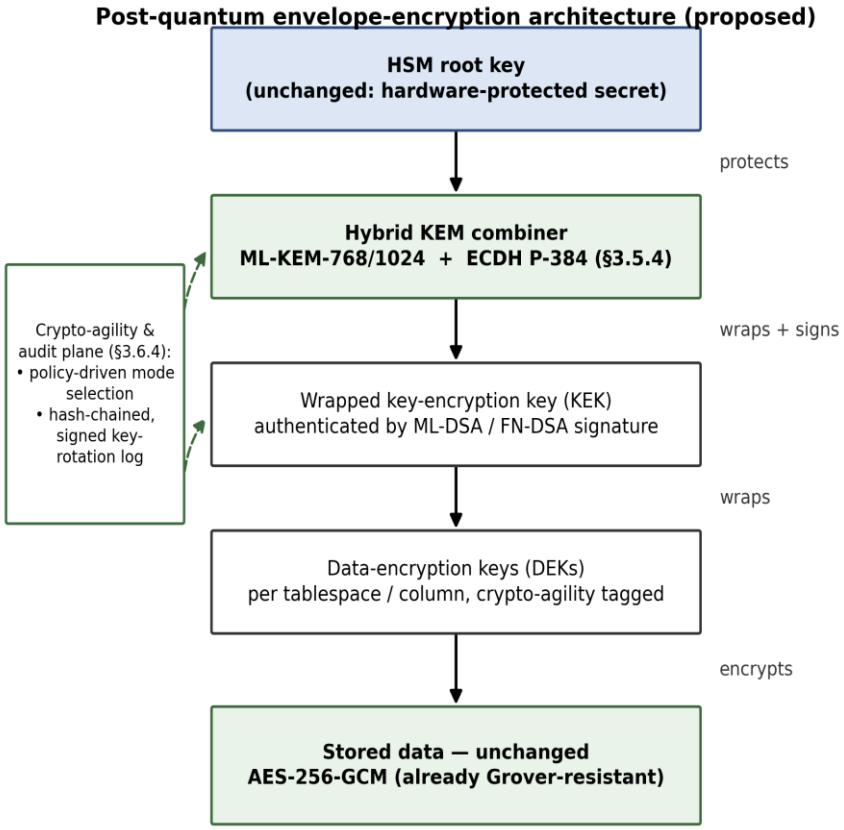


Fig. (6). Proposed post-quantum envelope-encryption architecture. Only the key-wrapping tier changes; stored data and its AES-256-GCM protection are untouched.

The corresponding decapsulation runs ECDH and ML-KEM.Decaps on the receiving (KMS/HSM) side and applies the identical KDF; per §4.1, security requires only that at least one of the two component mechanisms remains unbroken, which is exactly the "no single point of cryptographic failure" requirement of §3.2.

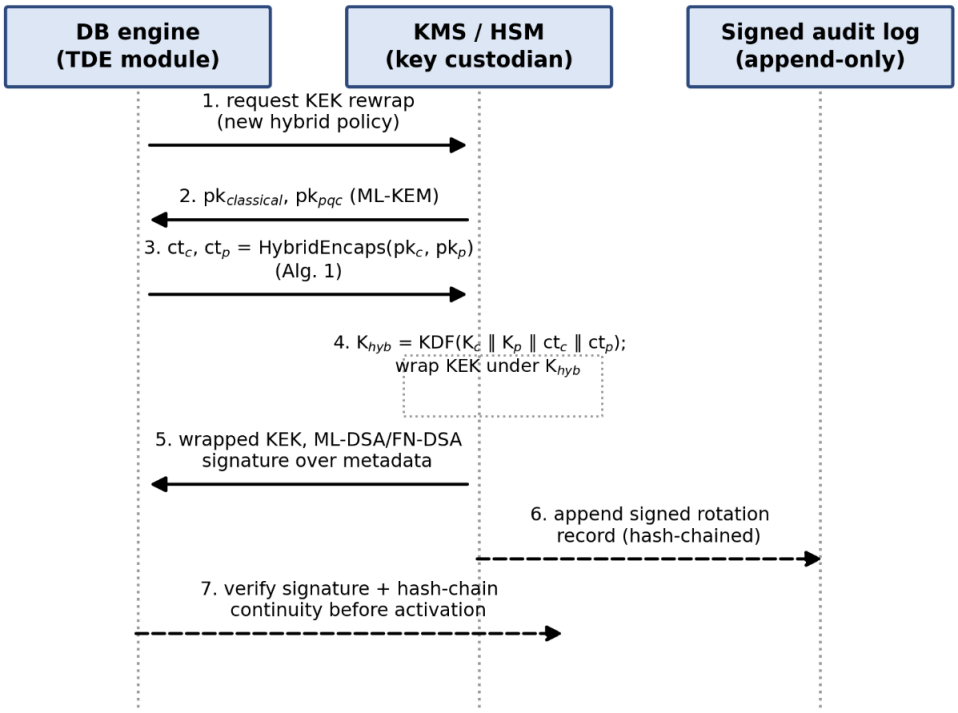


Fig. (7). Message-sequence diagram for the hybrid key-wrap exchange between the database engine, the KMS/HSM, and the signed audit log.

3.4.2 Enterprise-scale rekeying. A single hybrid encapsulation protects one KEK; an enterprise estate may hold anywhere from thousands to millions of KEKs across tenants and tablespaces, none of which can be rewrapped in a single atomic operation without unacceptable downtime. Algorithm (2) rewraps one KEK, and Fig. (8) shows the three-pass pipeline that applies it across an entire estate: an inventory pass ranks existing RSA/ECC-wrapped KEKs by urgency using each record's required confidentiality horizon (§4.5); a hybrid-wrap pass applies Algorithm (1) and Algorithm (2) to each KEK, batched across independent tenants for parallelism; and a verify-and-commit pass validates the new signature chain before atomically swapping the active KEK pointer, only then retiring the legacy KEK per the site's retention policy for cryptographic material.

Algorithm 2 *HybridRewrapKEK(KEK_plain, pk_c_new, pk_p_new)*

```
Output: new wrapped record (ct_c, ct_p, ciphertext_KEK)
1: (ct_c, ct_p, K_hyb) ← HybridEncapsulate(pk_c_new, pk_p_new)   // Alg. 1
2: ciphertext_KEK ← AES-256-GCM.Encrypt(K_hyb, KEK_plain)
3: meta ← {alg_ids, key_ids, timestamp, tenant_id}
```

```
4: h_new ← SignedKeyRotationAudit(meta, sk_sign, h_prev)        // Alg. 3
5: return (ct_c, ct_p, ciphertext_KEK, h_new)
```

Sizing this pipeline requires the wrapped-record's size overhead relative to the classical scheme it replaces. Dominated by the ML-KEM ciphertext of Table 4, the ciphertext expansion ratio is

$$\rho_{ct} = |ct_{pqc}| / |ct_{classical}| \tag{22}$$

— roughly a fourfold expansion at ML-KEM-768 relative to a 256-byte RSA-2048 ciphertext — and the full wrapped-record size sums the KEM ciphertext, the AES-wrapped KEK, and the authentication tag,

$$S_{wrap} = |ct_{KEM}| + |DEK|_{AES} + |tag|_{MAC} \tag{23}$$

both of which are immaterial next to typical DEK and metadata sizes but are exactly the effect §4.4 quantifies at audit-log scale. The wall-clock cost of rewrapping one KEK is the sum of unwrapping under the old scheme, wrapping under the new one, and signing the rotation record,

$$t_{DEK} = t_{unwrap}^{old} + t_{wrap}^{new} + t_{sign} \tag{24}$$

which §3.5 turns into a full-estate rekeying-time model.

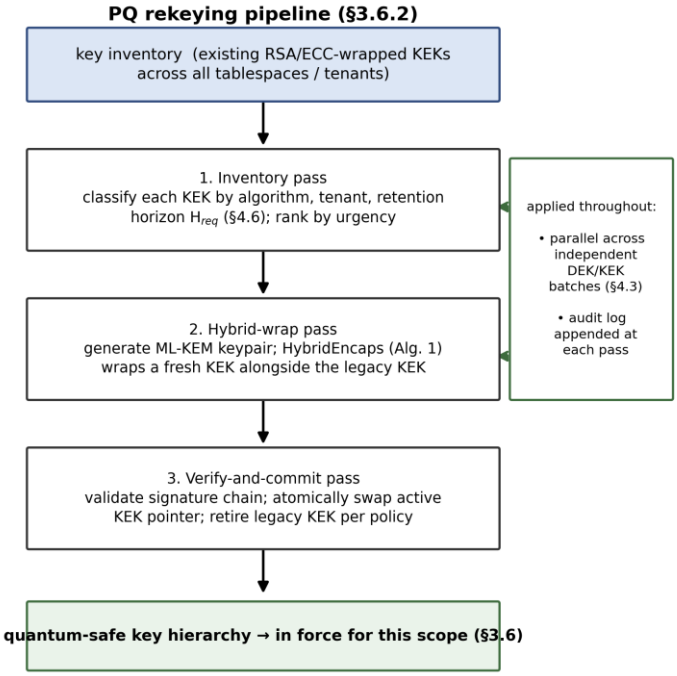


Fig. (8). The three-pass enterprise-scale rekeying pipeline built from Algorithm 2, applied across an inventory of existing KEKs.

3.4.3 Signed key-rotation audit. Every rewrap, and every crypto-agility mode change (§3.4.4), is

appended to a hash-chained, ML-DSA-signed (or FN-DSA-signed) log rather than merely recorded in a mutable metadata table. Algorithm (3) gives the append operation; because each record's hash covers the previous record's hash, an attempt to retroactively alter or delete a past rotation event is detectable by any party who re-verifies the chain, including an auditor who was not present at the time of the original rotation.

Algorithm 3 *SignedKeyRotationAudit(meta, sk_sign, h_prev)*

```
Output: new chain head h_new; appended, independently verifiable record
1: h ← Hash(h_prev ‖ serialize(meta))
2: σ ← ML-DSA.Sign(sk_sign, h)      // or FN-DSA.Sign once FIPS 206 is final
3: record ← (meta, h, σ)
4: append record to audit log (append-only store)
5: return h        // becomes h_prev for the next rotation
```

Because ML-DSA and FN-DSA signatures are larger than the ECDSA signatures such logs use today (Table 6), the audit log's own storage grows at a rate driven by the signature scheme and the rotation cadence,

$$S(t) = S_0 + r \cdot t, \quad r = f_{\text{rot}} \cdot |\sigma_{\text{FN-DSA}}| \tag{25}$$

where f_rot is the rotation frequency; §4.4 evaluates this growth numerically for ML-DSA-65 and FN-DSA-512 and is precisely why FN-DSA's compact signatures (§3.3.2) matter operationally even before FIPS 206 is finalized.

3.4.4 Crypto-agility negotiation. Because a large estate migrates incrementally, the framework must decide, for each key-management operation, whether to operate in classical-only, hybrid, or pure post-quantum mode. Algorithm (4) makes this decision from local policy, the peer or engine's advertised capability, and the outcome of the last successful negotiation, logging every decision (including any policy-permitted fallback) through Algorithm (3). Fig. (9) shows the resulting state machine: policy pressure (an internal mandate or an external one such as CNSA 2.0, §5.2) drives transitions toward pure PQC, while a capability mismatch can drive a logged, policy-bounded fallback in the other direction — but, per §3.2's second design goal, classical-only operation is disallowed for new keys once an operator's policy date has passed, regardless of peer capability.

Algorithm 4 *CryptoAgilityNegotiate(policy, peer_caps, last_mode)*

```
Output: negotiated mode ∈ {classical, hybrid, pure_pqc}
1: if policy.mandate = pure_pqc and pure_pqc ∈ peer_caps:
2:     mode ← pure_pqc
3: elif policy.mandate ≥ hybrid:
4:     if hybrid ∈ peer_caps: mode ← hybrid
5:     elif policy.allow_exception and now < policy.hard_deadline:
6:         mode ← classical   // logged fallback, Alg. 3
7:     else: raise PolicyViolation
8: else: mode ← last_mode
9: SignedKeyRotationAudit({mode, rationale}, sk_sign, h_prev)  // Alg. 3
10: return mode
```

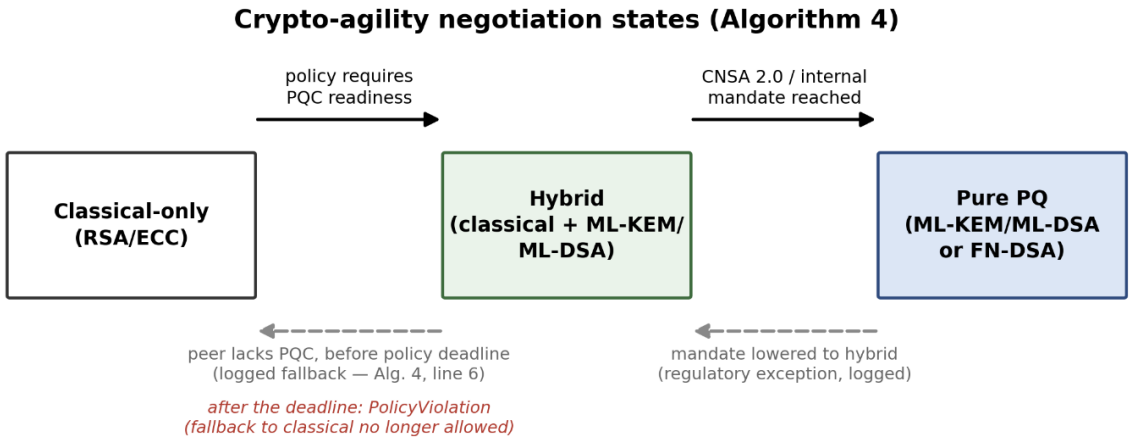


Fig. (9). Crypto-agility negotiation states. Transitions are driven by policy pressure and peer capability and are always logged (Algorithm 4).

3.4.5 Integration with database engines. The architecture is designed to sit beneath the same external-keystore or KMS-callback interfaces that Oracle, SQL Server, and PostgreSQL's pg_tde already expose (§2.1): none of these engines needs to know that its keystore now performs a hybrid encapsulation rather than a bare RSA unwrap, since the interface contract — hand the engine an unwrapped KEK or DEK on request — is unchanged. The practical integration work is therefore concentrated in the KMS/HSM layer and in the crypto-agility and audit components of Fig. (6), not in the database engine itself; §5.4 discusses the software stack this implies.

Table (9) lists the documented external-key interfaces through which each platform obtains keys, and the point at which the hybrid unwrap is introduced.

Table (9). External-key interfaces through which a hybrid unwrap can be introduced without modifying the database engine

Platform	External key interface (as documented)	Where the hybrid unwrap is inserted	Engine-side change
Oracle Database TDE	PKCS#11 HSM; Oracle Key Vault (KMIP)	KMS/HSM back end answering unwrap requests	None
Microsoft SQL Server TDE	Extensible Key Management (EKM) provider	EKM provider library	None (provider replaced)
PostgreSQL with pg_tde	KMIP or Vault key provider	Key-provider service	None
MySQL / InnoDB TDE	Keyring component (KMIP-backed)	Keyring component or KMIP server	None (component replaced)
Cloud KMS, customer-managed keys	Provider API; bring-your-own-key import	Customer-held wrapping key and import format	Depends on provider PQC support

3.4.6 Unwrap path and record format. The unwrap path mirrors Algorithm (1) inside the HSM boundary. Algorithm (5) runs both component decapsulations unconditionally, so timing does not

reveal which component failed, derives K_hyb with the identical KDF, and treats an AEAD tag failure as the only observable rejection: ML-KEM's implicit rejection guarantees that an invalid ct_p still yields a pseudorandom K_p, and the authenticated wrap converts the resulting mismatch into a uniform failure rather than a decryption oracle.

Algorithm 5 *HybridDecapsulate(sk_c, sk_p, ct_c, ct_p, wrapped_KEK)*

```
Output: KEK, or ⊥ (uniform failure); runs inside the HSM boundary
1: K_c ← ECDH-Decaps(sk_c, ct_c)          // always executed
2: K_p ← ML-KEM.Decaps(sk_p, ct_p)        // implicit rejection: never aborts
3: K_hyb ← KDF(K_c ‖ K_p ‖ ct_c ‖ ct_p)    // Eq. (21)
4: KEK ← AES-256-GCM.Decrypt(K_hyb, wrapped_KEK)
5: if tag check fails: audit-log(reject, record_id); return ⊥
6: zeroize K_c, K_p, K_hyb; return KEK
```

Table (10) and Fig. (10) give the byte-level layout of what is stored per rotation. The wrapped-KEK record is dominated by the 1,088-byte ML-KEM ciphertext; the size S_wrap of Eq. (23) is the 1,136-byte subtotal of ct_p, the wrapped key and the tag, and hybrid operation adds 165 bytes of classical share, nonce, header and timestamp for a 1,301-byte record. The audit record is 3,509 bytes, of which the ML-DSA-65 signature is 94%, which is the record size assumed by the storage model of §4.4.

Table (10). Byte-level record layouts (ML-KEM-768, ECDH P-384, ML-DSA-65)

Record	Field	Size (B)	Purpose
Wrapped KEK	Header: version, algorithm IDs, key IDs, tenant ID	48	Crypto-agility tag; KDF context
	Timestamp	8	Ordering; replay window
	ct_c: P-384 uncompressed point	97	Classical ECDH share
	GCM nonce	12	AES-256-GCM IV
	ct_p: ML-KEM-768 ciphertext	1,088	Lattice share
	Wrapped KEK payload	32	AES-256-GCM ciphertext of the KEK
	GCM tag	16	Authentication
Audit record	S_wrap subtotal (ct_p + payload + tag)	1,136	Eq. (23)
	Record total	1,301	Stored per KEK per rotation
	Rotation metadata	168	Event type, IDs, tenant, timestamp
	Chain hash h_i	32	Link to predecessor
	ML-DSA-65 signature	3,309	Non-repudiation
	Record total	3,509	Appended to audit log

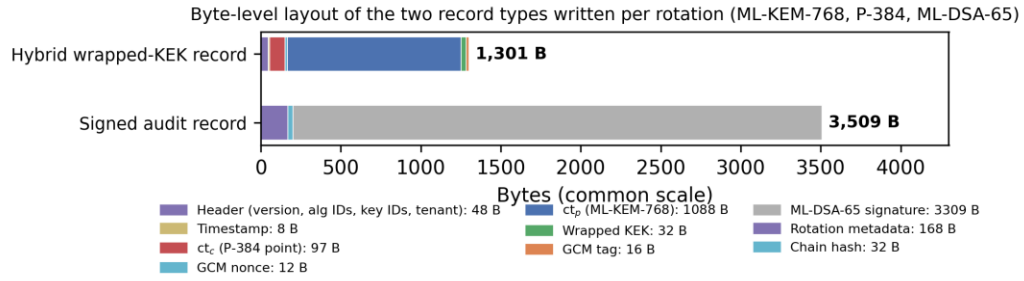


Fig. (10). The two record types written per rotation, drawn to a common byte scale: the signature dominates the audit record, and the ML-KEM ciphertext dominates the wrapped-KEK record.

3.4.7 Usage limits and rotation triggers. Post-quantum wrapping does not remove the classical limits on the symmetric layer, and a rotation policy that ignores them would be incomplete. With AES-256-GCM and random 96-bit initialization vectors, the probability that two of q_inv invocations under one key share an IV is approximately

$$p_{\text{coll}}(q_{\text{inv}}) \approx q_{\text{inv}}^2/2^{97} \le 2^{-32} \iff q_{\text{inv}} \lesssim 2^{32.5} \quad (26)$$

and NIST SP 800-38D caps random-IV use at 2^32 invocations per key **[45]**. The framework rotates a DEK at 2^30 invocations, a factor of four inside the cap, where the collision probability is about 2^-37. Table (11) applies the same reasoning to the other keys in the hierarchy. The hybrid wrap has a structural advantage here: because K_hyb is derived from fresh encapsulations, each K_hyb encrypts exactly one KEK and IV reuse under the wrapping key is impossible by construction.

Table (11). Usage limits and rotation triggers for each key class in the hierarchy

Key	Primitive / IV scheme	Per-key limit	Rotation trigger	Residual IV-collision probability
DEK (bulk data)	AES-256-GCM, random 96-bit IV	2^32 invocations [45]	2^30 invocations or 90 days	≈2^-37
KEK (wraps DEKs)	AES-256-GCM, random 96-bit IV	2^32 invocations	10^6 wraps or annual rotation	≈6×10^-18
K_hyb (wraps one KEK)	AES-256-GCM, one-shot key	1 invocation	Single use, fresh per wrap	0 (no key reuse)
Audit signing key	ML-DSA-65	No scheme-imposed limit	Annual; on suspected compromise	n/a
KMS hybrid key pair	ML-KEM-768 + ECDH P-384	No scheme-imposed limit	Annual, aligned with KEK rotation	n/a

3.4.8 Audit-chain verification. Algorithm (3) appends; Algorithm (6) is its counterpart, run by an independent auditor or by the engine before activating a new KEK. Each record commits to its predecessor and carries a signature,

$$h_i = \mathrm{H}(h_{i-1} \,\|\, \mathrm{ser}(m_i)), \quad \mathrm{Vfy}(pk, h_i, \sigma_i) = 1 \;\; \forall i \quad (27)$$

so verification is a single pass of n hash evaluations and n signature verifications; at roughly 0.1–0.2 ms per ML-DSA-65 verification, the 120,000-record ten-year log of §4.4 verifies in about 12–24 seconds on one core. Two details matter. The chain head must be anchored out of band — published to a separate write-once store — because an operator who controls both the log and the signing key could otherwise truncate and re-sign it; and signing-key rotation is itself a chain event, in which the outgoing key certifies the incoming one so that verification can cross the key boundary.

Algorithm 6 *VerifyAuditChain(log[1..n], pk_sign, h_0, h_head)*

```
Output: OK, or FAIL(i, reason)
1: h ← h_0                                  // trusted genesis / anchor
2: for i = 1 to n:
3:     (meta_i, h_i, σ_i) ← log[i]
4:     if h_i ≠ Hash(h ‖ serialize(meta_i)): return FAIL(i, chain)
5:     if not ML-DSA.Verify(pk_sign, h_i, σ_i): return FAIL(i, signature)
6:     if meta_i is a key-rotation event: pk_sign ← certified next key
```

```
7:     h ← h_i
8: if h ≠ h_head: return FAIL(n, head-mismatch)   // externally anchored head
9: return OK
```

3.4.9 Exposure-ranked migration scheduling. Which KEKs to migrate first has a principled answer. Mosca's inequality states that data with confidentiality lifetime X, migrated over a program of duration Y, is at risk if a CRQC arrives at time $Z < X + Y$ **[3]**; the exposure that remains after the CRQC arrives is measured in years of ciphertext still requiring protection,

$$X+Y>Z \implies E = \max(0, X+Y-Z) \tag{28}$$

Fig. (11) maps E over the (Z, X) plane for a three-year program: every record whose horizon exceeds $Z - Y$ lies in the exposed region, which for the financial, health and genomic horizons of §4.5 covers the entire illustrative CRQC window. Algorithm (7) turns this into a schedule that ranks KEKs by exposure and fills maintenance windows greedily, so the largest exposures migrate first.

Algorithm 7 *PrioritizeMigration(inventory, Y_est, Z_min, W, P)*

```
Output: ordered maintenance windows W_1, W_2, ...
1: for each KEK k in inventory:
2:     E_k ← max(0, X_k + Y_est − Z_min)        // Eq. (28)
3: sort inventory by (E_k descending, data volume descending)
4: cap ← P·(W − t_fixed) / t_DEK                // KEKs per window, Eq. (31)
5: for j = 1, 2, ...: W_j ← next cap KEKs in sorted order
6:     run Algorithm 2 on W_j in parallel; check with Algorithm 6
7: return [W_1, W_2, ...]
```

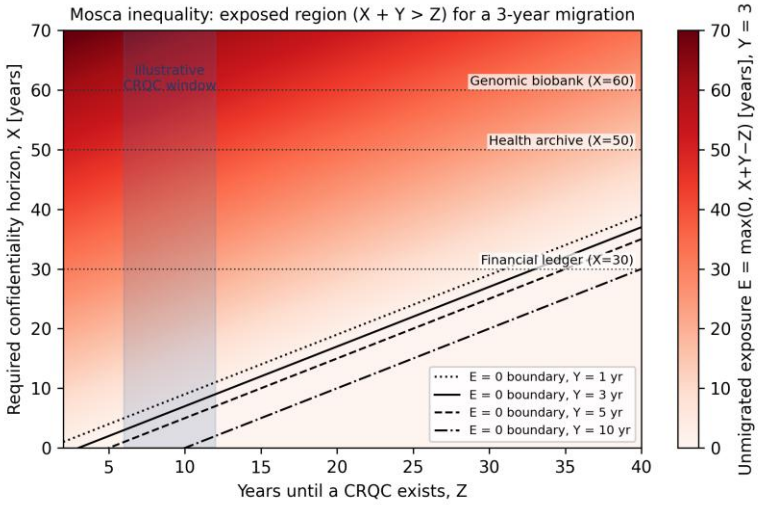


Fig. (11). Unmigrated exposure E over the (Z, X) plane for a three-year migration program. Case-study horizons lie in the exposed region across the whole illustrative CRQC window.

3.5 Security Analysis, Cost Model, and Case Study Design:

The security argument in §4.1 is a composition sketch rather than a full mechanized proof, following the same reduction style used for the underlying primitives. For the audit log, ML-DSA's EUF-CMA

security bounds the probability that an adversary who observes q_s signed rotation records can forge a new one,

$$\Pr[\text{forge}] \leq q_s \cdot \varepsilon_{\text{MSIS}} + 2^{-\lambda} \tag{29}$$

where ε_MSIS is the Module-SIS advantage at the chosen parameter set and λ is the target security parameter; this is the formal basis for treating a verified hash-chain as tamper-evident in §3.4.3. For the hybrid combiner, IND-CCA2 security of the wrapped KEK follows if at least one of the classical or post-quantum shared secrets is indistinguishable from random, since the KDF of Eq. (21) is modeled as a random oracle over its inputs; formally,

$$\text{Adv}_{\text{IND-CCA2}}^{\text{hyb}} \leq \text{Adv}^{\text{classical}} + \text{Adv}^{\text{MLWE}} + \text{negl}(\lambda) \tag{30}$$

so that a break requires simultaneously breaking ECDH over P-384 and Module-LWE at the chosen ML-KEM parameter set — the "no single point of failure" property of §3.2 stated as a security reduction. Section 4.1 also states explicitly what the framework does **not** protect against — side-channel leakage from a non-constant-time implementation (particularly relevant to FN-DSA's floating-point signing step, §5.3), an insider with direct plaintext access through the database engine's own access-control path, and data already harvested and later decrypted because it was never migrated before a CRQC existed.

The cost model in §4.2–§4.4 is analytical: it composes the per-operation sizes and times fixed by the standardized parameter sets of Table 4 and Table 6 (and, for encapsulation/decapsulation latency, published benchmark figures for those parameter sets) into estate-scale projections, in the same spirit as §3.4.2's per-KEK cost model but generalized across an inventory of N keys rewrapped by P parallel workers,

$$T(N,P) = \lceil N/P \rceil \cdot t_{\text{DEK}} + t_{\text{fixed}} \tag{31}$$

where t_fixed captures one-time overhead such as HSM session establishment and new keypair generation. Dividing the total estate rekeying time by the number of keys rekeyed gives an amortized per-key migration cost useful for capacity planning against a cloud KMS's usage-based pricing,

$$C_{\text{amortized}} = (T(N,P) \cdot \text{cost}_{\text{CPU}}) / N \tag{32}$$

Because this is a model calibrated against published component benchmarks rather than measurements of a from-scratch implementation of the full pipeline, §5.5 is explicit about the resulting threats to validity; the model is offered as a planning tool and a falsifiable prediction for the empirical validation identified as future work (§5.4), not as a substitute for it.

Section 4.5 evaluates four case studies chosen to span both the confidentiality-horizon dimension (§1.1) and the regulatory-driver dimension (§5.2): a financial ledger under a 30-year statutory retention requirement; a post-marketing medical-device and electronic-health-record archive with an approximately 50-year relevance horizon; a genomic biobank record whose family-relevance horizon is modeled at roughly 60 years; and a multinational cloud deployment whose binding constraint is

not data retention but a regulatory migration deadline (CNSA 2.0 and the EU coordinated roadmap, §5.2). For each, §4.5 computes a required-confidentiality condition against the illustrative CRQC arrival window used throughout this paper, and reports the projected migration cost from the model of §3.5 at a representative federation size for that sector.

4. Results

4.1 Security Analysis Results:

Composing the reductions of §3.5 gives the overall security posture of the proposed architecture. Confidentiality of a wrapped KEK reduces to the hybrid combiner's IND-CCA2 security (Eq. (30)), which in turn reduces to the harder of breaking ECDH over P-384 or Module-LWE at the deployed ML-KEM parameter set; because these rest on unrelated mathematical structures (the discrete-logarithm problem on an elliptic curve versus worst-case lattice problems), a single cryptanalytic advance is very unlikely to compromise both simultaneously, which is the concrete content behind §3.2's "no single point of cryptographic failure" requirement. Integrity and non-repudiation of the key-rotation history reduces to ML-DSA's EUF-CMA security (Eq. (29)) composed with the collision resistance of the hash function chaining successive audit records; an adversary who cannot forge a signature or find a hash collision cannot insert, delete, or reorder a rotation event without detection at the next verification pass (§3.4.2's verify-and-commit step).

Three things the architecture does **not** protect against should be stated precisely. First, side-channel leakage: the reductions say nothing about an implementation that leaks its key through timing or power, and FN-DSA's floating-point Gaussian sampler is a known, unresolved hazard (§5.3). Second, insider access through the engine's own query path: a DBA with SELECT privileges sees plaintext regardless of what protects the KEK, because TDE was never designed to stop authorized-path access (§3.1). Third, data already harvested under a classical-only wrap remains exactly as exposed as before — the framework protects from migration forward, not retroactively, which is why §4.5 tracks migration-completion time against each record's horizon.

Implementation risk deserves concrete treatment because it is where deployed lattice cryptography has actually failed. KyberSlash showed that secret-dependent division timings in the reference code and several optimized ML-KEM (Kyber) implementations allowed secret-key recovery within minutes to hours on embedded platforms, and was fixed by patching to constant-time arithmetic **[42]**. The lesson is not that the scheme is weak but that the reductions of §3.5 describe the mathematical object, and an HSM-side implementation must be validated separately for constant-time behavior. Table (12) maps each threat class to its mitigation and states the residual risk.

Table (12). Threat, mitigation and residual-risk matrix

Threat	Component	Mitigation	Residual risk
Timing leakage in decapsulation (cf. KyberSlash [42])	HSM / KMS provider	Constant-time builds; instrumented timing tests in CI; validated modules only	Novel microarchitectural leaks
Floating-point sampler leakage in FN-DSA signing	Audit signer	Default to ML-DSA; adopt FN-DSA only in validated modules	Until FIPS 206 implementations mature
Fault injection during decapsulation or signing	HSM	FO re-encryption check; redundant computation; tamper-resistant enclosure	Attacker with invasive physical access
Downgrade to classical-only mode	Negotiation (Algorithm 4)	Hard policy deadline; signed mode log; pinned minimum mode per tenant	Logged exceptions before the deadline
Audit-log truncation or rollback	Audit store	Hash chain plus externally anchored head (Algorithm 6)	Collusion of store operator and signing-key holder
AES-GCM IV collision on a hot DEK	Bulk encryption	Rotation at 2^30 invocations (Table 11)	Negligible (≈2^-37)
Pre-migration ciphertext already harvested	Backups, replicas	Re-wrap every live copy; crypto-shred retired KEKs	Copies exfiltrated earlier remain exposed

4.2 Key and Signature Size Results:

Fig. (12) compares public-key and ciphertext sizes across classical and ML-KEM parameter sets. ML-KEM-768's 1,184-byte public key and 1,088-byte ciphertext are roughly three to four times larger than the RSA-3072 or ECC P-256 values they would sit alongside in a hybrid wrap, but remain small in absolute terms — well under 2 KB — relative to typical network and storage budgets in an enterprise KMS exchange. Fig. (13) compares signature sizes: ML-DSA's signatures (2.4–4.6 KB depending on security level) are more than an order of magnitude larger than the ECDSA or RSA signatures a legacy audit trail would use, which is the direct motivation for Table 6's inclusion of FN-DSA as a compact alternative — FN-DSA-512's roughly 666-byte signature is smaller than an RSA-3072 signature and nearly seven times smaller than ML-DSA-87's, at the cost of FN-DSA's draft standardization status and harder-to-implement signing procedure (§2.2, §5.3).

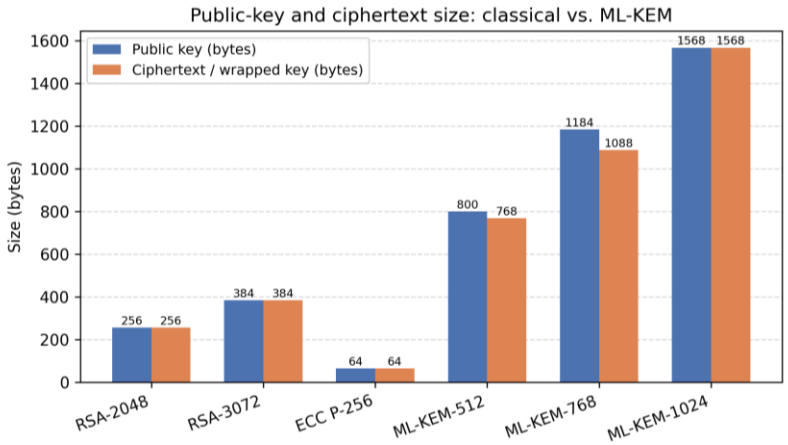


Fig. (12). Public-key and ciphertext size: classical RSA/ECC versus ML-KEM at each NIST security category.

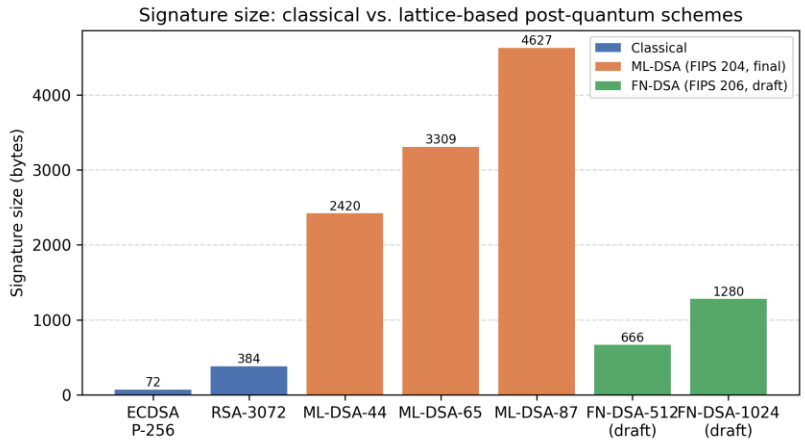


Fig. (13). Signature size: classical schemes versus lattice-based ML-DSA (final) and FN-DSA (draft).

4.3 Rekeying Throughput, Latency, and Planning Sensitivity:

Fig. (14) applies the full-estate rekeying model of §3.5 across DEK counts from one thousand to ten million and parallelism levels from one to five hundred and twelve workers. At the smallest scale tested, even a single worker completes the migration in a few seconds; the practically interesting regime is above roughly one hundred thousand DEKs, where the gap between low and high parallelism widens from single-digit multiples to more than two orders of magnitude — at ten million DEKs, the modeled time falls from over ten hours at P = 1 to under ninety seconds at P = 512. This is the direct, actionable output of the cost model for a migration planner: for a federation above roughly one hundred thousand keys, worker parallelism dominates every other lever available to reduce migration time, including algorithm choice within the lattice family.

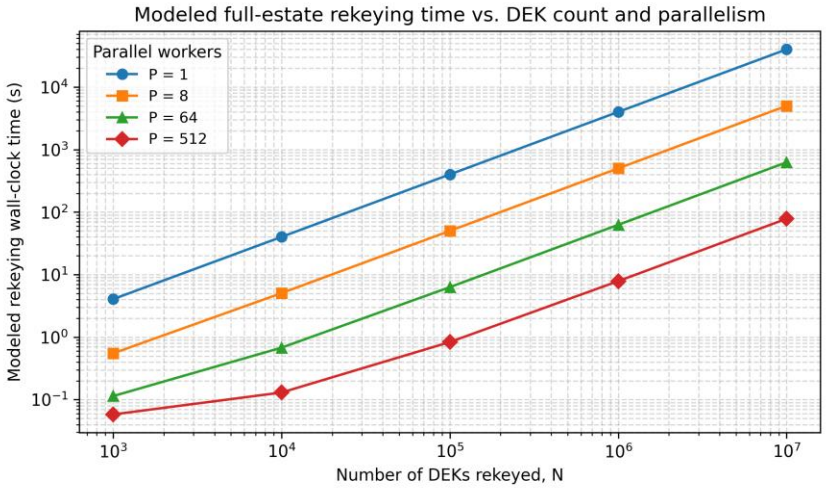


Fig. (14). Modeled full-estate rekeying time as a function of DEK count and parallel-worker count (Eq. (31)).

Fig. (15) looks at the same operation from a steady-state, multi-tenant perspective rather than a one-time migration burst: the modeled CPU overhead that ongoing key-management operations (periodic rotation, on-demand rewraps) add to a shared KMS/HSM cluster as tenant count grows. Pure ML-KEM/ML-DSA operation adds a modeled 1–12% CPU overhead across the tested range of ten to ten thousand tenants; hybrid operation, which performs both a classical and a post-quantum operation on every call, roughly doubles that overhead — consistent with the doubling relationship formalized in §4.4's Eq. (37) — reaching a modeled 24% at ten thousand co-located tenants. Neither curve is prohibitive at the scales tested, but the gap between them is the concrete price of the hybrid mode's stronger "no single point of failure" guarantee (§3.2), and is one input to the hybrid-versus-pure-PQC discussion in §5.3.

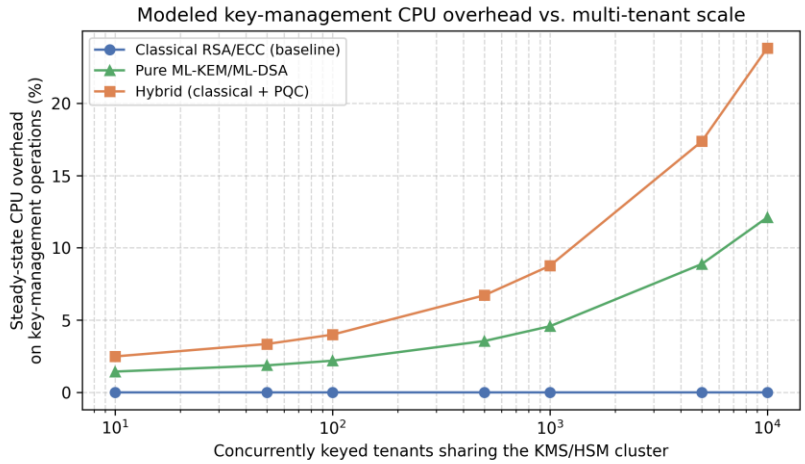


Fig. (15). Modeled steady-state CPU overhead of key-management operations versus the number of tenants sharing a KMS/HSM cluster.

Cluster-wide propagation of a new KEK — for example, across R read replicas that must all recognize the newly active key before old-key retirement — adds a communication term proportional to the number of replicas and to each replica's need to receive and verify the new wrap and its signature. Modeling this propagation as a balanced broadcast tree gives a communication volume of

$$B_{\text{cluster}} = R \cdot (|ct_{\text{KEM}}| + |\sigma|) \cdot \log_2 R \tag{33}$$

which, for a wrapped record and signature of roughly 1.1 KB and 3.3 KB respectively (ML-KEM-768 and ML-DSA-65) across a sixteen-replica cluster, projects under 300 KB of total replication traffic per rekeying event — negligible next to normal replication-stream volume, and the reason §4.4's storage results, not this communication term, dominate the framework's long-run operational cost.

The scale results above hide a structural fact that Table (13) makes explicit: per-KEK rewrap time is dominated not by post-quantum arithmetic but by HSM transport and legacy-scheme unwrap, which

together account for about 3.5 of the 4.0 ms. Moving from a classical RSA-3072/ECDSA rewrap (3.67 ms) to hybrid ML-KEM-768/ML-DSA-65 (4.00 ms) adds 9.0%; pure post-quantum operation adds 5.2%; and even the highest security level adds only about 12%. The t_DEK = 4 ms used throughout §4 is therefore insensitive to the choice of lattice parameter set, and the sensitivity that matters for planning is to parallelism (§4.3).

Table (13). Per-KEK rewrap latency components (ms), modeled from published primitive benchmarks and a fixed HSM round-trip

Component (ms)	Classical	Pure PQ (L3)	Hybrid L1	Hybrid L3	Hybrid L5
HSM transport / session	2.60	2.60	2.60	2.60	2.60
Legacy-scheme unwrap	0.90	0.90	0.90	0.90	0.90
New KEM wrap	0.06	0.04	0.17	0.18	0.20
KDF + AES-GCM	0.02	0.02	0.02	0.02	0.02
Signature	0.05	0.26	0.18	0.26	0.35
Audit append	0.04	0.04	0.04	0.04	0.04
Total t_DEK	3.67	3.86	3.91	4.00	4.11
Overhead vs. classical	—	+5.2%	+6.5%	+9.0%	+12.0%

A planning question follows directly from this sensitivity: because parallelism dominates migration time, it is usually inverted in practice — given a maintenance window W, how many workers are needed? Solving Eq. (31) for P gives

$$P_{\min} = \lceil (N \cdot t_{\mathrm{DEK}}) / (W - t_{\mathrm{fixed}}) \rceil \tag{34}$$

For N = 10^7 keys and t_DEK = 4 ms, a one-hour window needs P_min = ⌈40,000 / 3,599.95⌉ = 12 workers and a fifteen-minute window needs 45; a 10^8-key estate needs 112 workers for a one-hour window. Fig. (16) tabulates T(N, P) over six orders of magnitude in each dimension and outlines the region that fits a one-hour window. The feasible boundary is a straight staircase in log-log space, as Eq. (31) implies: each tenfold growth in N requires a tenfold growth in P to hold the window constant. Estates up to 10^5 keys finish in under seven minutes on a single worker, so parallelism engineering matters only for the largest estates. In practice the binding constraint is HSM concurrency rather than CPU — session slots and PKCS#11 parallelism cap the usable P — which is why §5.4 recommends horizontally scaled HSM pools and why the transport term of Table (13) dominates.

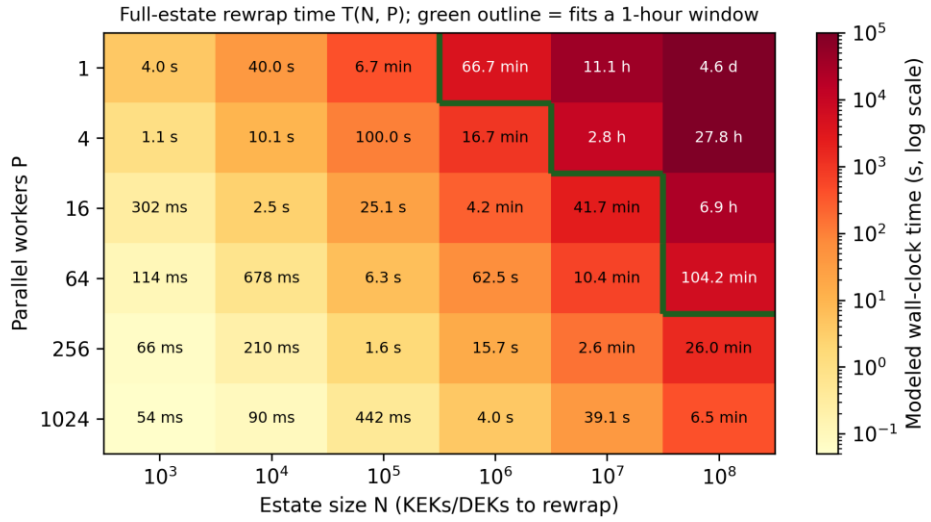


Fig. (16). Modeled full-estate rewrap time T(N, P) for t_DEK = 4 ms. The green boundary encloses the (N, P) combinations that complete within a one-hour maintenance window.

4.4 Storage, Audit-Log, and Hybrid-versus-Pure-PQ Cost:

Because every rekeying and every crypto-agility decision appends a signed record (§3.4.3), the audit log's storage grows continuously rather than as a one-time migration cost. For an estate of n_t tenants each holding n_d DEKs, the storage committed to wrapped-key material alone is

$$C(n_t,n_d) = n_t \cdot n_d \cdot S_{\text{wrap}} + n_t \cdot |\text{chain}| \tag{35}$$

and, projecting this across a year of operation at a given rotation cadence,

$$C_{\text{annual}} = \sum_{i=1}^{12} T(N,P_i) \cdot \text{cost}_{\text{CPU}} \cdot \Delta_i \tag{36}$$

Fig. (17) evaluates the audit-log component of this cost numerically for 1,000 tenants under monthly rotation over a ten-year horizon. ECDSA-signed rotation records, included as a classical baseline, accumulate to roughly 31 MB over ten years — a trivial storage cost, but one that offers no post-quantum protection for the rotation history itself. ML-DSA-65-signed records accumulate to roughly 400 MB over the same period, still a modest figure for enterprise storage budgets but nearly thirteen times the classical baseline; FN-DSA-512-signed records, despite being a draft standard, accumulate to only about 100 MB — a quarter of ML-DSA's footprint — which is the clearest numerical case in this paper for tracking FN-DSA's finalization closely even while deploying ML-DSA as the default signer today (§5.3).

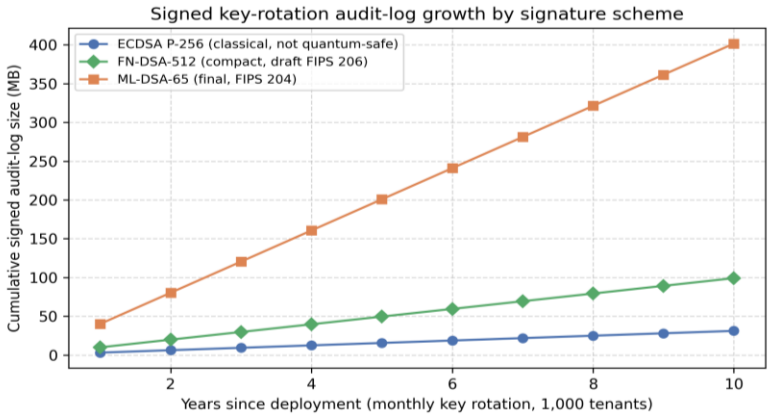


Fig. (17). Cumulative signed key-rotation audit-log size over ten years, by signature scheme, for 1,000 tenants under monthly rotation.

Fig. (18) isolates the per-operation cost difference between hybrid and pure-ML-KEM operation across the three NIST security categories. Because the hybrid combiner performs a full ECDH exchange in addition to ML-KEM encapsulation and decapsulation, its modeled per-operation time is consistently higher by roughly 140 microseconds — the dominant cost of the classical operation itself — across all three levels, closely matching the doubling-order relationship

$$C_{\text{hybrid}} = O(2 \cdot C_{\text{pure-PQC}}) \tag{37}$$

already visible in Fig. (15)'s steady-state overhead curves. At NIST Level 1 (ML-KEM-512) the absolute gap is largest in relative terms (pure PQC at 52 µs versus hybrid at 192 µs, a nearly fourfold difference) because the ECDH term is a larger fraction of a smaller base cost; at Level 5 the same 140-µs addition is proportionally smaller (118 µs versus 258 µs). Since §3.2 requires hybrid operation as the default during migration regardless of security level, this result's practical implication is that the absolute overhead of the "no single point of failure" guarantee is small and roughly constant in absolute terms across security levels, even though it looks larger in percentage terms at the lowest level.

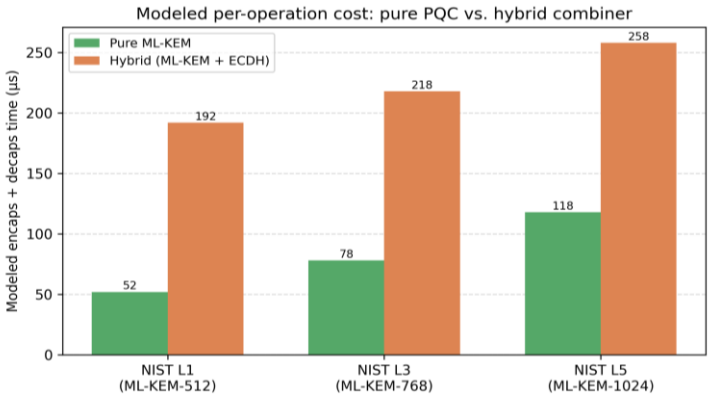


Fig. (18). Modeled per-operation cost of pure ML-KEM versus the hybrid ML-KEM/ECDH combiner, across NIST security categories.

4.5 Case Study Results:

A record's migration is timely if it completes before the record's required confidentiality horizon is threatened by CRQC arrival — formally, if the time remaining until the retention obligation ends still exceeds the time needed to complete migration,

$$H_{\text{req}} = t_{\text{retention}} - t_{\text{migration}} > 0 \tag{38}$$

and the risk this condition is meant to avoid grows, for an unmigrated record, with every year that passes after harvesting begins and before migration completes — writing κ for the harvest timestamp,

$$P_{\text{break}}(t) = 1 - (1 - p_{\text{CRQC}})^{t - \kappa} \tag{39}$$

Fig. (19) plots each case study's confidentiality horizon against the illustrative CRQC arrival window used throughout this paper, and Table (14) reports the modeled migration cost for each at a representative estate size using the cost model of §3.5. The financial ledger (30-year retention, ending 2056) and the multinational cloud deployment (bound by a 2033 regulatory deadline rather than a data-retention horizon) both complete comfortably within the illustrative CRQC window under the modeled parallelism; the 50-year health archive — whose confidentiality obligations derive from HIPAA in the United States and the GDPR in the European Union **[25], [26]** — and the 60-year genomic biobank cases both extend the confidentiality requirement well past the CRQC window regardless of when within it a CRQC actually arrives, which is the strongest form of the paper's motivating argument — for these two cases, H_req in Eq. (38) is satisfied only because migration is projected to complete in well under an hour of modeled processing time, not because the underlying data's exposure risk is otherwise small.

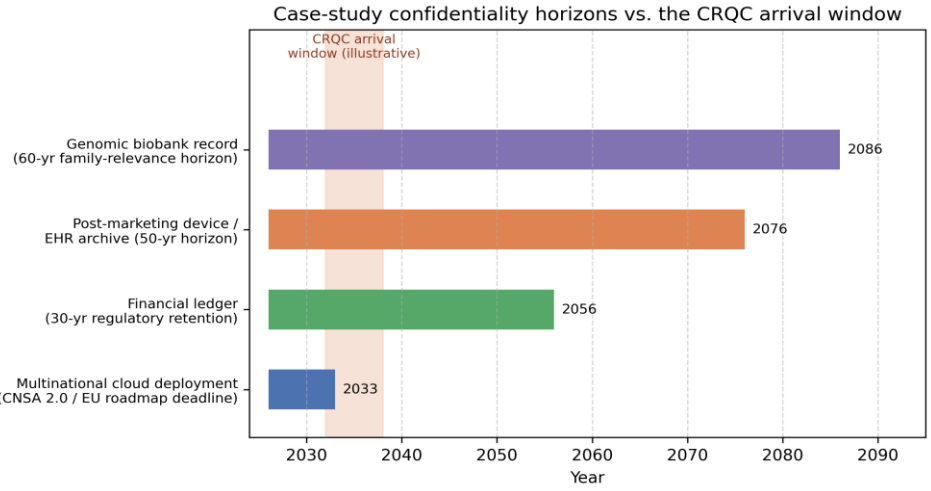


Fig. (19). Case-study confidentiality horizons plotted against the illustrative CRQC arrival window of Fig. 1.

Table (14). Case-study parameters and modeled migration cost

Case study	Estate scale (DEKs)	Horizon / deadline	Configuration	Modeled migration time
Financial ledger (30-yr retention)	2,000,000	Ends 2056	ML-KEM-768 + ML-DSA-65 hybrid	≈125 s @ P=64
Health/device archive (50-yr horizon)	10,000,000	Ends ≈2076	ML-KEM-1024 + ML-DSA-87 hybrid	≈641 s @ P=64
Genomic biobank (60-yr horizon)	500,000	Ends ≈2086	ML-KEM-1024 + ML-DSA-87 (FN-DSA-1024 pending)	≈32 s @ P=64
Multinational cloud (CNSA 2.0 / EU deadline)	50,000,000	Deadline 2033	ML-KEM-768 + ML-DSA-65 hybrid	≈391 s @ P=512

Table (15) extends this result across both the security-level and estate-scale dimensions independently, giving a migration planner a single reference for the combination closest to their own deployment.

Table (15). Extended parameter grid: wrapped-record size and modeled full-estate rekey time

Configuration	Wrapped-record size	Estate size N	Rekey time @ P=64
ML-KEM-512 + ML-DSA-44	≈816 B	100,000	≈6.1 s
ML-KEM-768 + ML-DSA-65	≈1,136 B	100,000	≈6.3 s
ML-KEM-1024 + ML-DSA-87	≈1,616 B	100,000	≈6.5 s
ML-KEM-768 + ML-DSA-65	≈1,136 B	10,000	≈0.7 s
ML-KEM-768 + ML-DSA-65	≈1,136 B	1,000,000	≈63 s
ML-KEM-768 + ML-DSA-65	≈1,136 B	10,000,000	≈625 s

Table (16) evaluates Eq. (28) for the four case studies over the illustrative CRQC window of Fig. (19) (Z between 6 and 12 years from 2026) with a three-year program duration Y. Every long-horizon case is exposed across the whole window: if migration were never performed, E would be 21–27 years for the financial ledger, 41–47 for the health archive and 51–57 for the genomic biobank. The analytical migration times of Table (14) — minutes of compute — are far smaller than the three-year program Y, which is dominated by change control, testing and HSM firmware qualification rather than by cryptographic work. The mandate-driven deployment is the only case in which an early and a late CRQC give different answers, since its X is a deadline (seven years) rather than a retention horizon.

Table (16). Worked evaluation of Mosca's inequality for the case studies (Y = 3 years)

Case study	X (yr)	Y (yr)	X + Y	E at Z = 6	E at Z = 12
Financial ledger (30-yr retention)	30	3	33	27	21
Health / device archive (50-yr horizon)	50	3	53	47	41
Genomic biobank (60-yr horizon)	60	3	63	57	51
Multinational cloud (deadline-driven)	7	3	10	4	0

5. Discussion

5.1 Comparative Positioning and Crypto-Agility as an Architectural Property:

Table (17) positions the proposed framework against the closest existing bodies of work identified in §2. Commercial TDE/KMS products (§2.1) solve envelope encryption but wrap keys exclusively under classical RSA/ECC operations, with no post-quantum path. Generic hybrid-KEM combiner results such as Bindel et al. **[22]** supply the security definition the combiner of §3.4 relies on, but stop at the primitive: they do not address key-rotation audit, estate-scale rekeying, or migration scheduling. The IETF's hybrid key exchange for TLS 1.3 **[23]** demonstrates the same combiner idea in wide production use, but for a per-session handshake rather than a long-lived, at-rest key hierarchy that must be inventoried, audited, and migrated incrementally. NIST SP 800-57 and draft IR 8547 **[19], [24]** set governance timelines and key-lifecycle expectations but are policy documents, not architectures — they say a migration must happen and by when, not how to sequence or verify it. None of these four points of comparison, individually or combined, gives an enterprise architect an algorithmic answer to "which key rotates first, what does the record look like, and how is a rotation later proven not to have been tampered with." That gap is what §3's seven algorithms, byte-level record format, and exposure-ranked scheduling are built to close.

Table (17). Comparison with the closest existing approaches (✓ = provided, ✗ = not provided, ~ = partial)

Capability	Classical TDE/KMS [27– 29]	Hybrid-KEM combiner theory [22]	TLS 1.3 hybrid key exchange [23]	NIST SP 800-57 / IR 8547 [19, 24]	This work
Post-quantum key encapsulation	✗	✓ (generic)	✓	guidance only	✓ (ML-KEM, hybridized)
Signed, hash-chained rotation audit	~ (vendor logging)	✗	✗	guidance only	✓ (Algorithms 3, 6)
Enterprise-scale rekeying model (10^3–10^8 keys)	✗	✗	✗	✗	✓ (Alg. 2, §4.3)
Exposure-ranked migration scheduling	✗	✗	✗	✗	✓ (Mosca-based, Alg. 7)
Crypto-agility negotiation (classical/hybrid/pure-PQC)	✗	~ (fixed combiner)	~ (session-only)	guidance only	✓ (Alg. 4)
Byte-level record format & per-key usage limits	✗	✗	✗	✗	✓ (§3.4.6–§3.4.7)
Analytical cost model at enterprise scale	✗	✗	✗	✗	✓ (§3.5, §4)
Designed against real TDE engine interfaces	✓ (native)	✗	✗	✗	✓ (§3.4.5, 9)

The results of §4.3 and §4.4 make a further case that is easy to state but easy to under-value: the largest lever over migration cost is not which lattice parameter set an enterprise chooses, but whether its key hierarchy was built, from the outset, to run two algorithm families side by side and to switch between them on policy rather than on a rewrite. Algorithm 4's negotiation state machine and

Algorithm 3's signed audit trail exist specifically so that a decision to move from hybrid to pure post-quantum mode — or a temporary, logged fallback in the other direction — is a configuration change with a permanent record, not a redeployment. This is the same lesson the TLS 1.3 hybrid key-exchange literature reached for transport security **[23]**, applied here to a key hierarchy that, unlike a TLS session, persists for years and must be renegotiated against an inventory rather than a single handshake.

5.2 Regulatory and Governance Implications:

Table (18) aligns the standardization and mandate timelines cited throughout. NIST finalized ML-KEM, ML-DSA and SLH-DSA in August 2024 **[4]–[6]**; FN-DSA remains a draft, submitted for approval in August 2025 with finalization expected in late 2026 or early 2027 **[7]**. On the mandate side, NSA's CNSA 2.0 suite specifies ML-KEM and ML-DSA for national security systems, with default use by 2030 and exclusive use by 2033, ahead of the White House's 2035 government-wide target **[16], [17]**; the EU roadmap asks member states to begin transition by the end of 2026 and cover critical infrastructure by 2030 **[18]**. Most immediately, U.S. federal civilian agencies are working against a 2026 schedule that requires a migration plan in late October 2026, with 2030–2031 deadlines for high-value assets **[20], [34]**, a shift the Cloud Security Alliance describes as moving from roadmap to rule **[34]**. None of these mandates targets commercial databases directly, but any enterprise selling into or operating alongside a government or defense supply chain should expect evidence of exactly this kind of key-hierarchy redesign to become a procurement and audit requirement well before the hard deadlines.

Table (18). Regulatory and standardization timeline summary

Instrument	Applies to	Key date(s)
NIST FIPS 203/204/205	General-purpose PQC standards	Finalized August 2024
NIST FIPS 206 (FN-DSA)	Compact signature standard	Draft; final expected late 2026 / early 2027
NSA CNSA 2.0	National security systems	Default by 2030; exclusive by 2033
White House NSM-10	U.S. government-wide	Quantum-resistance target by 2035
EU coordinated roadmap	EU member states	Begin 2026; critical infrastructure by 2030
2026 U.S. federal migration schedule	U.S. federal civilian agencies	Migration plans due ≈Oct. 2026; HVAs 2030–31

5.3 The Hybrid-versus-Pure-Post-Quantum Question:

Section 4.4's roughly twofold overhead of hybrid operation raises the obvious question of why not move directly to pure post-quantum operation now that ML-KEM and ML-DSA are final. Three considerations argue for hybrid as the default through the medium term. First, Module-LWE has roughly two decades less cryptanalytic scrutiny than factoring and discrete logarithms, and a hybrid combiner stays secure if a weakness emerges that the current parameter sets do not anticipate. Second, CNSA 2.0 does **not** approve hybrid constructions for national-security systems and requires pure post-quantum algorithms on its exclusive-use timeline **[16]**, so an enterprise selling into that market

must eventually support pure-PQC mode; Algorithm 4 accommodates this as a policy transition rather than a redesign. Third, FN-DSA's draft status (§2.2, §5.2) means the smallest lattice signature is not yet a stable target; an architecture that runs ML-DSA now and adopts FN-DSA after FIPS 206 finalizes, without another key-hierarchy redesign, is preferable to committing early. The recommendation is to default to hybrid until an operator's risk tolerance or an external mandate requires otherwise, exactly as Algorithm 4 encodes.

5.4 Future Work, Implementation, and Deployment Considerations:

Four extensions follow from this paper's scope decisions. First, empirical validation of the cost model of §3.5 against a from-scratch implementation built on liboqs **[21]** and a real TDE-capable engine is the most direct next step. Second, FN-DSA's finalization (§5.2) should be tracked given §4.4's storage results, with constant-time validation prioritized ahead of any deployment (§4.1). Third, the case studies assume single-database deployments; vertically partitioned or schema-heterogeneous federations, and searchable or homomorphic encryption over lattices where a workload must compute on still-encrypted records (§2.4) **[30], [31]**, are larger follow-on problems. Fourth, the threat model treats the HSM/KMS as trusted (§3.1); tolerating a partially compromised or multi-vendor key-custodian layer, in the spirit of threshold constructions, would relax that assumption at a round-trip cost the current model does not capture.

The most direct implementation path combines existing components. The Open Quantum Safe project's liboqs and its OpenSSL 3.x provider already implement ML-KEM and ML-DSA and can be loaded into a KMS or HSM cryptographic stack without modifying the engine, consistent with §3.4.5 **[21]**. HSM firmware support is the more variable dependency: several vendors have announced ML-KEM and ML-DSA support, but FN-DSA support is essentially absent given its draft status, an operational reason, independent of §5.3, to plan FN-DSA as a later upgrade. The audit component needs only an append-only store that most enterprises already operate for compliance logging; the work is wiring Algorithms (3) and 6 into it.

5.4.1 Reference topology. Fig. (20) shows a reference deployment: a three-node KMS cluster hosting the PQC crypto provider sits between the database tier and an HSM pool whose root keys are unchanged, with a separate append-only audit store and an independent verifier running Algorithm (6). The KMS nodes speak KMIP or PKCS#11 to the engines over mutually authenticated TLS, itself configured with a hybrid key exchange so that the channel carrying wrapped keys is not the weak link of an otherwise quantum-safe design **[23]**; the audit store should share no failure domain with the database or the KMS.

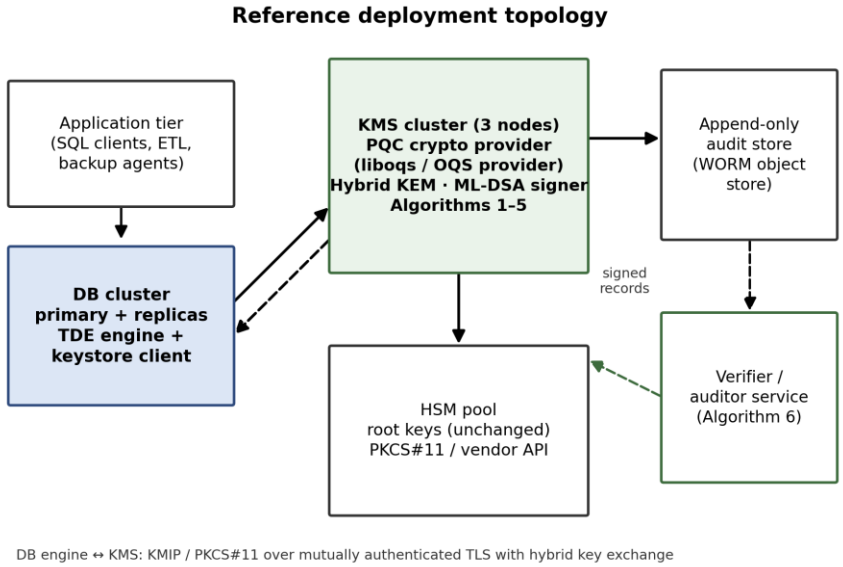


Fig. (20). Reference deployment topology: the PQC crypto provider lives in the KMS tier; engines, HSM root keys and audit storage are unchanged in role.

5.4.2 Phased roadmap. Table (19) sequences the migration into five phases whose exit criteria are measurable rather than calendar-driven. Inventory produces the (X_k, data-volume) ranking consumed by Algorithm (7); the pilot runs hybrid wraps in shadow mode, verifying every unwrap against the legacy path before any production key is switched; the rollout migrates in exposure order; pure-PQ readiness is gated on FIPS 206 finalization and HSM firmware qualification; and legacy retirement disables classical-only mode once the last logged exception has expired. Phase boundaries are anchored to the CNSA 2.0 default (2030) and exclusive (2033) dates, but Algorithm (7) compresses the schedule for long-horizon tenants.

Table (19). Phased migration plan with measurable exit criteria

Phase	Window	Scope	Exit criterion
P0 Inventory and ranking	Q4 2026–Q2 2027	Enumerate KEKs, algorithms and retention horizons; compute E_k	≥99% of KEKs classified; ranking approved
P1 Hybrid pilot	Q2–Q4 2027	One tenant group; shadow verification against legacy unwrap	Zero unwrap mismatches over 30 days; overhead within ≈10% of Table 13
P2 Hybrid rollout	2028–2029	All tenants in descending E_k order	100% of KEKs hybrid-wrapped; audit chain verified end to end
P3 Pure-PQ readiness	2029–2030	Pure mode on staging; FN-DSA evaluation after FIPS 206	Malformed-ciphertext and downgrade tests pass; HSM firmware qualified
P4 Legacy retirement	2030–2033	Disable classical-only mode; retire legacy KEKs	No classical-only records; all exceptions expired

5.4.3 Validation. Conformance and side-channel evidence should be gathered before production keys move: known-answer tests for ML-KEM and ML-DSA against NIST's published vectors and, where the module vendor supports it, algorithm validation through NIST's ACVP process; constant-time analysis of the provider build (§4.1); fuzzing of the wrapped-record parser, whose variable-length ML-KEM and ML-DSA fields enlarge the parsing surface relative to fixed-size RSA blobs; and a fault-injection test that kills the KMS mid-rewrap to confirm that the atomic pointer swap of §3.4.2 leaves either the old or the new KEK active, never neither.

5.5 Limitations and Threats to Validity:

Four limitations bound the conclusions. First, the cost model of §3.5 is analytical, calibrated against published component benchmarks, so §4's absolute figures are projections to be validated, not measurements. Second, the security argument of §3.5 and §4.1 is a composition sketch rather than a machine-checked proof. Third, FN-DSA's parameters could still change before FIPS 206 finalizes, so figures using it (Table 6, Fig. 13, Fig. 17) are provisional. Fourth, the four case studies are illustrative constructions spanning a range of horizons and regulatory drivers, not measurements of any named deployment.

Construct validity: the cost model's inputs — record sizes from the standards and the illustrative timings of §4.3, §4.4 and Table (13) — come from published specifications and typical benchmark ranges rather than one authoritative source, so a specific stack could shift absolute numbers while leaving the qualitative trends intact (parallelism dominates at scale, hybrid roughly doubles per-operation cost, FN-DSA cuts audit-log growth). External validity: the CRQC window used in Fig. (1), Fig. (19) and Eq. (39) is illustrative, reflecting unresolved expert disagreement about timelines **[32]**; the conclusion that long-retention data should migrate its key-management tier well before that window does not depend on any date within it. Measurement validity does not strictly apply because §4 reports a model, which is why §5.4 names empirical validation as the direct next step.

6. Conclusion

This research set out to answer a narrower question than “how should enterprise data be protected against quantum computers,” and that narrowing is its central argument: because AES-256 already resists Grover's algorithm, the exposure in an enterprise database's data-at-rest protection sits in the RSA- and ECC-based key-wrapping, key-exchange and signing operations above the data, not in the data's own encryption. The paper presented an envelope-encryption architecture that replaces exactly that tier with a hybrid ML-KEM/classical combiner, authenticates every key-rotation event with ML-DSA or FN-DSA signatures over a hash-chained audit log, and negotiates between classical, hybrid and pure post-quantum modes under logged policy control. Seven algorithms and a byte-level record format made the architecture concrete; a composition-style argument grounded it in Module-LWE, NTRU-lattice and Module-SIS hardness; and an analytical cost model projected key size, per-tenant overhead and full-estate rekeying time across federations from tens to tens of thousands of tenants. The four case studies made the motivating claim concrete: the financial ledger's 30-year retention and

the multinational deployment's 2033 deadline both complete migration comfortably within the illustrative quantum-computing window, but the 50-year health archive and 60-year genomic record extend their required confidentiality well past it whenever a cryptographically relevant quantum computer actually arrives — for these, migration times of well under an hour of modeled compute, and a program measured in years (Table 16), are the entire margin of safety. Against the regulatory landscape of §5.2, the practical recommendation is not to wait for a mandate naming a specific database product but to treat the key-management-tier redesign as the well-defined, boundable piece of post-quantum migration that can start now, while the empirical validation, FN-DSA finalization and HSM firmware support identified in §5.4 and §5.4 mature around it. The cost model is an analytical planning tool rather than a measured system, and closing that gap with a from-scratch implementation is the direct next step.

7. References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," SIAM J. Comput., vol. 26, no. 5, pp. 1484–1509, 1997.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in Proc. 28th Annu. ACM Symp. Theory of Computing (STOC), 1996, pp. 212–219.

[3] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?," IEEE Security & Privacy, vol. 16, no. 5, pp. 38–41, 2018.

[4] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard," FIPS 203, Aug. 2024.

[5] National Institute of Standards and Technology, "Module-Lattice-Based Digital Signature Standard," FIPS 204, Aug. 2024.

[6] National Institute of Standards and Technology, "Stateless Hash-Based Digital Signature Standard," FIPS 205, Aug. 2024.

[7] National Institute of Standards and Technology, "FFT-over-NTRU-Lattice-Based Digital Signature Standard (FN-DSA)," FIPS 206 (draft), submitted for approval Aug. 2025; final expected late 2026 / early 2027.

[8] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in Proc. 37th Annu. ACM Symp. Theory of Computing (STOC), 2005, pp. 84–93.

[9] V. Lyubashevsky, C. Peikert, and O. Regev, "On ideal lattices and learning with errors over rings," in Advances in Cryptology — EUROCRYPT 2010, LNCS vol. 6110, Springer, 2010, pp. 1–23.

[10] A. Langlois and D. Stehlé, "Worst-case to average-case reductions for module lattices," Des. Codes Cryptogr., vol. 75, no. 3, pp. 565–599, 2015.

[11] J. Hoffstein, J. Pipher, and J. H. Silverman, "NTRU: A ring-based public key cryptosystem," in Algorithmic Number Theory (ANTS-III), LNCS vol. 1423, Springer, 1998, pp. 267–288.

[12] J. Bos et al., "CRYSTALS — Kyber: A CCA-secure module-lattice-based KEM," in Proc. IEEE Eur. Symp. Security and Privacy (EuroS&P), 2018, pp. 353–367.

[13] L. Ducas et al., "CRYSTALS — Dilithium: A lattice-based digital signature scheme," IACR Trans. Cryptogr. Hardw. Embed. Syst. (TCHES), vol. 2018, no. 1, pp. 238–268, 2018.

[14] P.-A. Fouque et al., "Falcon: Fast-Fourier lattice-based compact signatures over NTRU," NIST Post-Quantum Cryptography Standardization submission, 2020.

[15] G. Alagic et al., "Status report on the third round of the NIST post-quantum cryptography standardization process," NISTIR 8413, 2022.

[16] National Security Agency, "Announcing the Commercial National Security Algorithm Suite 2.0," CNSA 2.0 Advisory, Sep. 2022.

[17] The White House, "National Security Memorandum on Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems," NSM-10, May 2022.

[18] European Commission, "Recommendation on a Coordinated Implementation Roadmap for the Transition to Post-Quantum Cryptography," Apr. 2024, formalized with EU member states Jun. 2025.

[19] National Institute of Standards and Technology, "Transition to Post-Quantum Cryptography Standards," NIST IR 8547 (draft), 2024.

[20] U.S. Office of Management and Budget, memorandum on federal agency post-quantum cryptography migration planning, 2026.

[21] D. Stebila and M. Mosca, "Post-quantum key exchange for the internet and the Open Quantum Safe project," IACR Cryptol. ePrint Arch., Rep. 2016/1017, 2016.

[22] N. Bindel, J. Brendel, M. Fischlin, B. Goncalves, and D. Stebila, "Hybrid key encapsulation mechanisms and authenticated key exchange," in Post-Quantum Cryptography (PQCrypto 2019), LNCS vol. 11505, Springer, 2019, pp. 206–226.

[23] D. Stebila, S. Fluhrer, and S. Gueron, "Hybrid key exchange in TLS 1.3," IETF Internet-Draft draft-ietf-tls-hybrid-design, 2023 (work in progress).

[24] National Institute of Standards and Technology, "Recommendation for Key Management, Part 1: General," NIST SP 800-57, rev. 5, 2020.

[25] U.S. Department of Health and Human Services, "Standards for Privacy of Individually Identifiable Health Information (HIPAA Privacy Rule)," 45 C.F.R. Parts 160 and 164, 2002 (as amended).

[26] European Parliament and Council of the European Union, "Regulation (EU) 2016/679 (General Data Protection Regulation)," Official Journal of the European Union, L119, 4 May 2016.

[27] Oracle Corporation, "Advanced Security: Transparent Data Encryption," Oracle Database Documentation, 2024.

[28] Microsoft Corporation, "Transparent Data Encryption (TDE)," SQL Server Documentation, 2024.

[29] PostgreSQL Global Development Group / Percona, "pg_tde: Transparent Data Encryption for PostgreSQL," project documentation, 2024–2026.

[30] J. Fan and F. Vercauteren, "Somewhat practical fully homomorphic encryption," IACR Cryptol. ePrint Arch., Rep. 2012/144, 2012.

[31] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in Advances in Cryptology — ASIACRYPT 2017, LNCS vol. 10624, Springer, 2017, pp. 409–437.

[32] Global Risk Institute, "Quantum Threat Timeline Report," M. Mosca and M. Piani, eds., annual report series, 2023.

[33] Encryption Consulting, "FN-DSA (FIPS 206): standardization status and implementation considerations," industry analysis, 2026.

[34] Cloud Security Alliance, "PQC compliance mandate convergence," CSA Research Note, Jul. 2026.

[35] National Institute of Standards and Technology, "Post-Quantum Cryptography Project" overview and call for proposals, 2016.

[36] Amazon Web Services, "AWS Key Management Service cryptographic details," AWS Documentation, 2024.

[37] L. Chen et al., "Report on post-quantum cryptography," NISTIR 8105, 2016.

[38] C. Gidney and M. Ekerå, "How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits," Quantum, vol. 5, p. 433, 2021.

[39] C. Gidney, "How to factor 2048 bit RSA integers with less than a million noisy qubits," arXiv:2505.15917, May 2025.

[40] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," Phys. Rev. A, vol. 86, no. 3, p. 032324, 2012.

[41] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, "Quantum resource estimates for computing elliptic curve discrete logarithms," in Advances in Cryptology — ASIACRYPT 2017, LNCS vol. 10625, Springer, 2017, pp. 241–270.

[42] D. J. Bernstein et al., "KyberSlash: Exploiting secret-dependent division timings in Kyber implementations," IACR Trans. Cryptogr. Hardw. Embed. Syst. (TCHES), vol. 2025, no. 2, pp. 209–234, 2025.

[43] E. Alkim, L. Ducas, T. Pöppelmann, and P. Schwabe, "Post-quantum key exchange — a new hope," in Proc. 25th USENIX Security Symp., 2016, pp. 327–343.

[44] M. R. Albrecht et al., "Estimate all the {LWE, NTRU} schemes!," in Security and Cryptography for Networks (SCN 2018), LNCS vol. 11035, Springer, 2018, pp. 351–367.

[45] M. Dworkin, "Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC," NIST SP 800-38D, Nov. 2007.

[46] The Quantum Insider, "Q-Day just got closer: Three papers in three months are rewriting the quantum threat timeline," industry analysis, Mar. 2026.