electronics

MDPI

Article

Authenticated Key Release for Electronic Health Records During Migration to Post-Quantum Cryptography

Faisal Albalwy [1,2]

[1] Department of Cybersecurity, College of Computer Science and Engineering, Taibah University, Madinah 42353, Saudi Arabia; fbalwy@taibahu.edu.sa
[2] Energy, Industry, and Advanced Technologies Research Center, Taibah University, Madinah 42353, Saudi Arabia

Abstract

Migration to post-quantum cryptography must preserve the conditions for releasing electronic health record keys. This study evaluates a signed off-chain release protocol binding a trusted accepted event to the expected recipient and record, migration policy, and paired single-use state. It supports RSA, ML-KEM-only, and a custom dual-contribution hybrid. The inherited zero-knowledge proof does not bind requester and record identities; accepted events remain trusted inputs. Symbolic agreement holds for an external adversary and one honest recipient. Implementation tests exercise real cryptographic operations, rejection, and failure cases; 27 coordinated races between distinct releases for one grant each returned exactly one key. Issuer-side exposure records support an optional inventory gate conditional on truthful histories and consistent record-key identity; content-identifier alias tests reveal the limits of per-identifier exposure histories. Across five restarted native processes, validated preloading reduced median legacy and hybrid service latencies from 48.501 to 2.085 ms and from 48.824 to 2.404 ms relative to per-call RSA import; ML-KEM-only remained near 1.5 ms. These operational comparisons use unmatched security levels and exclude network, blockchain, concurrent-load, and durable-state costs. Static recipient keys provide no forward secrecy, and later post-quantum protection cannot repair earlier key exposure.

Keywords: post-quantum cryptography; ML-KEM; authenticated key release; downgrade resistance; electronic health records; zero-knowledge proofs; hybrid migration; replay protection

Check for updates

Academic Editor: Hung-Yu Chien

Received: 4 October 2026
Revised: 24 October 2026
Accepted: 25 October 2026
Published: 28 October 2026

Copyright: © 2026 by the author. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Electronic health record (EHR) sharing requires controlled access to protected records [1,2]. Long-lived encrypted information also faces a harvest-now-decrypt-later threat: an adversary may retain a release encrypted under Rivest–Shamir–Adleman (RSA) and recover its record key if a sufficiently capable quantum computer becomes available [3–5]. Access control determines whether a request is accepted, whereas key release determines whether the intended recipient can recover the encrypted record. A valid access proof or an auditable ledger does not remove the exposure associated with RSA-based key release.

The starting point for this study is the published Zero-Knowledge Electronic Health Record (ZK-EHR) prototype for cross-institutional sharing [6]. That work investigated proof-based policy enforcement and encrypted record retrieval. Its architecture combines

a Groth16 proof workflow, a blockchain verifier, encrypted off-chain storage, and RSA Optimal Asymmetric Encryption Padding (RSA-OAEP) for delivery of a content-encryption key (CEK) [7,8]. Building on that architecture, the present study examines migration of the RSA-based key-release mechanism after an accepted access request. It reuses the record-encryption and authorization interfaces to examine this operation separately. It does not introduce the earlier architecture as a new contribution or establish the soundness of its authorization statement.

The scientific problem is broader than replacing one encryption algorithm. During migration, a release must remain associated with the correct authority, requester, record, and access grant. The receiver must distinguish an authorized legacy mode from an attempted downgrade and prevent reuse of a release or grant. Moreover, a record key previously recoverable through RSA cannot acquire a new confidentiality history merely because the same key is later protected by a post-quantum mechanism. Related studies address group authentication and key agreement for industrial devices and lattice-based certificateless signcryption for smart healthcare [9,10]. The specific problem addressed here is preserving release conditions for existing record keys while legacy and post-quantum modes coexist.

The Module-Lattice-Based Key-Encapsulation Mechanism (ML-KEM) and Module-Lattice-Based Digital Signature Algorithm (ML-DSA) provide standardized post-quantum components for key establishment and signatures, respectively [11,12]. Key-encapsulation guidance and Hybrid Public Key Encryption (HPKE) clarify composition and authentication, but the application must still define its authorization context, replay rules, and migration policy [13,14]. The gap examined here is how to express and evaluate those rules at the post-authorization key-release boundary of the existing EHR prototype. This is an application-protocol contribution, rather than a new primitive or a claim that existing composition methods are insufficient.

We propose a signed release artifact that binds the authority and requester to the record and accepted blockchain event. Canonical encoding gives the authenticated fields an unambiguous representation [15]. A trusted registry selects one of three modes: legacy RSA, ML-KEM-only, or a hybrid in which independent RSA and ML-KEM contributions derive one wrapping key. Two replay indexes enforce single use of the release and its grant context. The policy rejects fallback, and new direct-RSA releases set an irreversible exposure marker for their record commitments. Consistent with managed migration guidance, this design implements a fixed profile with explicit transitions; it does not demonstrate arbitrary algorithm negotiation or a second interchangeable suite [16].

The study addresses four research questions:

RQ1. *Integration correctness:* Can ML-KEM-based release recover the expected key and plaintext through the inherited workflow, given a trusted accepted access event?

RQ2. *Rejection behavior:* Are representative cases of malformed input, tampering, context substitution, expiry, replay, and downgrade rejected without releasing a key?

RQ3. *Migration cost:* How do the implemented modes compare in issuer-to-requester service latency and transmitted artifact size, and how does RSA key loading affect that comparison?

RQ4. *Coexistence:* How do mode policy and a scoped historical-inventory gate distinguish exposed, unexposed, and unresolved record-key histories?

The study makes three contributions. First, it specifies a release-policy layer that binds an accepted grant to an authenticated artifact, a registry-selected mode, and paired release/grant single-use state. The issuer records direct-RSA exposure, and an optional inventory gate uses these records to classify record-key histories; the classification depends on truthful inventory and consistent record-key identity. Second, it evaluates the layer through

scoped symbolic analysis, integration and failure checks with real cryptographic providers, and coordinated concurrency tests. A qualitative comparison identifies which application rules an alternative KEM-based profile could retain and which require redefinition. Third, five process runs quantify how RSA key loading and object reuse affect local service cost. Together, these contributions define and evaluate the policy and state rules for key release that a cryptographic primitive alone does not specify. Synthetic records and a single host limit conclusions about clinical deployment and performance on other platforms.

Section 2 positions the study within the literature. Sections 3 and 4 define the system assumptions and protocol, and Section 5 describes the evaluation. Sections 6–8 present the results, security analysis, and interpretation. Limitations and conclusions follow. Appendix A summarizes the encoding and validation rules needed to interpret the construction.

2. Related Work

The relevant literature addresses cryptographic migration, composition of key-establishment mechanisms, and healthcare integration. Sources were identified through a targeted, non-systematic search of NIST CSRC, IETF Datatracker, IACR ePrint, and publisher-indexed healthcare and blockchain literature, with a final search date of 20 October 2026. Search terms included ML-KEM, RSA migration, hybrid or composite KEM, EHR, key release, and downgrade. Sources were selected for direct relevance to migration policy, authenticated key establishment, or the inherited healthcare workflow; unrelated blockchain applications and reports lacking an identifiable primary publication were excluded.

2.1. Post-Quantum Migration and Cryptographic Agility

Migration guidance places algorithm replacement within a wider process of inventory, risk assessment, and controlled transition. Joint Cybersecurity and Infrastructure Security Agency (CISA), National Security Agency (NSA), and NIST guidance emphasizes data lifetime and staged roadmaps. NIST publications address adaptable algorithms and deprecation, while RFC 9958 discusses key-encapsulation semantics, larger cryptographic objects, hybrid designs, and implementation risks [3,4,16,17].

FIPS 203 and FIPS 204 specify ML-KEM and ML-DSA, respectively, but do not define an EHR application's release policy [11,12]. The present study implements a limited form of managed transition: a trusted registry selects among three modes within one fixed profile. Explicit identifiers prevent the interpretation of an existing release from changing silently. A second interchangeable suite would require additional design and evaluation; it is not demonstrated here.

2.2. Authenticated Key Establishment and Hybrid Composition

A key-encapsulation mechanism establishes shared secret material but does not, by itself, identify the application authority that requested the release. SP 800-227 discusses KEM use; HKDF provides key derivation, and the JSON Canonicalization Scheme (JCS) defines reproducible serialized inputs [11,13,15,18]. HPKE combines a KEM, a key-derivation function, and authenticated encryption. Its Base mode does not authenticate the sender, whereas its other modes provide pre-shared-key or asymmetric authentication under their stated assumptions [14]. Thus, authority authentication is not unique to the present protocol. The distinction is its integration with a signed application artifact, accepted access grant, migration policy, and exposure history.

Hybrid composition also requires more than placing two ciphertexts in one message. Generic combiner research and hybrid authenticated key-exchange analysis examine the assumptions needed when component mechanisms fail independently [19,20]. X-Wing binds the traditional ciphertext and public key in its X25519/ML-KEM combiner. The LAMPS

composite-ML-KEM draft also incorporates traditional ciphertext and public-key material into derivation; the HPKE post-quantum draft specifies ML-KEM-based options [21–23]. Pathum and De Silva examine phased RSA-to-ML-KEM migration with HPKE integration, providing a closer migration comparator than a primitive standard alone [24].

The construction studied here, termed the *ZK-EHR dual-contribution hybrid*, derives one CEK-wrapping key from an ML-KEM secret and an independent RSA-protected contribution. Its wire identifier remains `hybrid-rsa-mlkem`. ML-DSA authenticates the ciphertext fields, but neither those ciphertexts nor public-key bytes enter the hybrid derivation directly. Payload binding therefore depends on correct signature verification. The scoped symbolic analysis in Section 7 does not supply a generic indistinguishability under adaptive chosen-ciphertext attack (IND-CCA) theorem.

Retaining this construction permits a controlled comparison at the inherited RSA/CEK interface. RFC 9180 Auth requires a KEM with `AuthEncap`/`AuthDecap`; the KEMs in the cited HPKE-PQ draft do not provide these functions [14,23]. A PQ-KEM HPKE profile would therefore require a separately specified authentication mechanism, such as an appropriate PSK or an external signature, rather than a drop-in use of Auth. X-Wing would replace the traditional component with X25519; LAMPS includes RSA-OAEP composites with different derivation and binding rules. Section 6.7 compares application-rule transfer qualitatively. A replacement needs a new profile and evaluation; no alternative implementation or performance advantage is demonstrated here.

2.3. Healthcare Systems and the Published Baseline

Blockchain-based EHR research separates ledger-supported access or audit from encrypted off-chain content, while key management and interoperability remain distinct concerns [2]. Recent healthcare studies address ML-KEM protection of medical images and DICOM binding, blockchain multifactor authentication with post-quantum signatures, and the CITADEL EHR architecture [25–27]. Olushola and Meenakshi examine an adjacent, general-purpose authenticated ML-KEM/ML-DSA session protocol rather than an EHR-specific workflow [28]. Hao et al. compare conventional protection with a Kyber-inspired hospital simulation; this is relevant to migration motivation but is not a full ML-KEM implementation comparator [29].

Tan et al. study group authentication and key agreement with non-linkability and integrity for industrial Internet of Things devices [9], while Sourav et al. study lattice-based certificateless signcryption using blockchain for smart healthcare [10]. These studies provide context for authentication and key protection in adjacent settings. The present evaluation concerns the release of an existing CEK relative to a trusted accepted grant and its migration history. No cross-paper timing comparison is made.

The published ZK-EHR study provides the specific architectural context for this work [6]. Its research focus was proof-based policy enforcement and encrypted retrieval. The current study retains those interfaces and investigates authenticated key release after an accepted event. Release-level replay controls do not implement proof-level nullifiers or revocation. Accepting an event as input also does not establish that requester and record identity were proved within the Groth16 statement.

2.4. Coexistence and the Application-Level Gap

Hybrid terminology and protocol guidance require the security of a combination to be stated explicitly and downgrade behavior to be considered [30–32]. HybridTrust negotiates the strongest mutually supported mode, including classical ECC when it is the only common capability [33]. This selection supports legacy devices; failed authentication or integrity checks instead exclude the affected update. The present study addresses a

different boundary: authority-controlled record-key release with a fixed no-fallback policy for migrated requesters.

Coexistence also creates a historical constraint. If a record key remains independently recoverable from a retained RSA release, adding an ML-KEM release cannot change the protection of that same key. The service marks new direct-RSA releases by record commitment. The optional inventory gate evaluates admission under scoped, trusted history assertions; it does not independently establish the completeness of institutional records.

2.5. Comparative Positioning

Table 1 groups the selected sources by their research boundary rather than ranking them by features specific to our schema. Standards and neighboring applications provide components and alternatives; they were not designed to implement this particular EHR workflow.

Table 1. Selected related work and its relationship to the present study.

Source or Source Group	Primary Focus	Relationship to This Study
FIPS 203/204 and SP 800-227 [11–13]	Cryptographic primitives and KEM-use guidance.	Supply components; application grants and migration state require separate definition.
HPKE [14]	KEM/KDF/AEAD composition, with Base and authenticated modes.	Offers an alternative construction; application replay and downgrade policy remain necessary.
KEM combiners, X-Wing, and LAMPS [19,21,22]	Hybrid composition and component binding.	Motivate explicit assumptions; their guarantees are not transferred to our custom hybrid.
Phased RSA-to-ML-KEM migration [24]	Algorithm transition with HPKE integration.	Adjacent migration design, rather than an EHR grant and exposure-state protocol.
Healthcare systems and adjacent session protocols [25–28]	Medical-image and EHR protection, authentication, and general session establishment.	Provide application and protocol context; the evaluated objects differ from post-authorization record-key releases.
Hospital simulation [29]	Kyber-inspired comparative simulation.	Motivates migration; not a full ML-KEM implementation benchmark.
HybridTrust [33]	Healthcare-device coexistence and negotiated sessions.	Capability-based classical-mode selection differs from fixed migration policy; failed validation does not trigger fallback.
Published ZK-EHR [6]	Proof-based access and encrypted record retrieval.	Provides the architecture and trusted event interface extended by this study.
Present study	Authenticated key release during RSA/ML-KEM coexistence.	Evaluates grant/context binding, paired single-use state and mode policy, plus issuer-side exposure recording and an optional inventory gate.

The contribution is therefore the design and evaluation of migration rules at a defined application boundary. The results do not establish that the custom construction is superior to HPKE or composite KEMs. They examine which additional relationships an EHR key-release service must preserve when the cryptographic mechanism changes.

3. System Model and Threat Model

The protocol begins with an accepted `AccessGranted` event, treated as a *trusted external authorization assertion*, and ends when the requester passes the recovered CEK to the record decryptor. Authentication of a release establishes agreement with this assertion and the

independently expected context. It does not establish the correctness of the upstream access decision.

3.1. Relationship to the Published ZK-EHR System

The published system combines a Circom access-policy circuit, Groth16 proofs, a Solidity verifier, and encrypted records addressed by content identifiers (CIDs) through the InterPlanetary File System (IPFS) [6,8]. Records use the Advanced Encryption Standard in Galois/Counter Mode (AES-GCM), with a fresh CEK delivered through RSA-OAEP [7]. The current prototype retains these components and interfaces to isolate the effect of changing key release. Reuse does not imply that every file is identical to the publication-time implementation.

The new protocol adds authority authentication, a trusted registry, release and replay state, explicit migration modes, and a direct-RSA exposure marker for new releases (Figure 1). It transports the existing 32-byte CEK without changing the record cipher or proof system. The `legacy-rsa` comparator is also part of this new authenticated service; it is not a rerun of the earlier Node release implementation. Consequently, the present latency measurements should not be compared directly with the earlier study's proof-generation, verification, or storage timings.

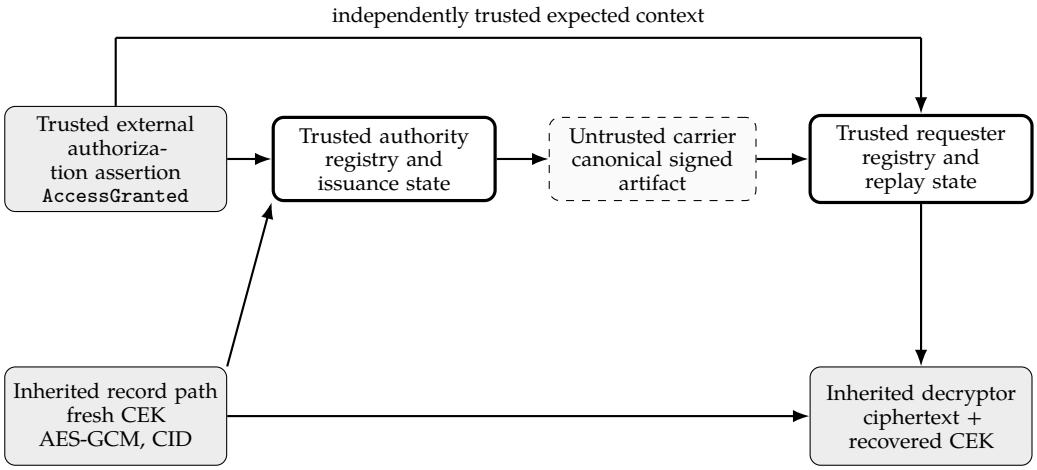

Figure 1. Key-release architecture. Shaded boxes identify retained workflow components; thick-bordered boxes identify the evaluated release services. The dashed carrier is untrusted. Accepted authorization and independently expected context remain trusted inputs.

3.2. Actors, Assets, and Trust Boundaries

The system distinguishes the party that encrypts a record, the authority that releases its CEK, and the endpoint that decrypts it. This separation prevents record-ciphertext integrity from being treated as authorization to release the key. Table 2 states the actors and the trust placed in each one.

The principal confidential assets are the CEK, plaintext, requester private keys, ML-KEM/RSA secret contributions, and derived wrapping keys. Integrity assets include the artifact, authority identity, registry snapshot, event and record context, capability state, and issuance/replay/exposure ledgers. Availability matters because a fail-closed store or provider failure prevents release, but this local study does not claim a production availability design.

Build and dependency provenance form an additional privileged boundary for the experiment operator.

Table 2. System components, roles, and trust assumptions.

Component	Role and Trust Assumption
Record producer and store	Produce encrypted records and supply ciphertext. Retrieved content requires digest and AES-GCM verification; transport is untrusted.
Prover, verifier, and chain	Supply the accepted event and receipt coordinates. Their authorization and finality are input assumptions.
Release authority	Bind the event and record, select policy, and sign the release. Its endpoint and ML-DSA key are trusted.
Registry	Bind owners, keys, activation windows, and capability state. A coherent snapshot is trusted and cannot be overridden by artifact fields.
Artifact carrier	Convey signed bytes. Fully untrusted: Observation, delay, replacement, modification, duplication, and replay are allowed.
Requester	Validate the release and recover the key. Its private keys, clock, independently expected context, and process are trusted.
State stores	Enforce identifier uniqueness, direct-RSA exposure marking for new releases, and paired replay state. Local in-memory and isolated SQLite semantics are trusted.

3.3. Adversary Capabilities and Trusted Assumptions

The artifact adversary knows the algorithms, schemas, identifiers, public keys, blockchain events, and ciphertext locations. It can retain RSA releases for a harvest-now-decrypt-later attack; create chosen ML-KEM encapsulations to public requester keys; mutate or replace signed fields and cryptographic payloads; submit malformed, non-canonical, expired, cross-context, or mixed-mode objects; race duplicate deliveries; reuse a grant with a new release identifier; substitute a requester key; and attempt a legacy downgrade. The model does not assume that the adversary can break ML-KEM-768, ML-DSA-65, HKDF-SHA-256, AES-256-GCM, or SHA-256 directly.

The analysis relies on the following assumptions:

- The expected release authority and its signing key, the requester endpoint and private keys, the trusted registry, replay state, local clock, and configured chain/contract are honest and uncompromised.
- The selected cryptographic libraries and the inherited AES-GCM record implementation behave as specified.
- The local prototype receives one accepted, finalized `AccessGranted` event and its exact receipt coordinates as trusted evidence.
- The event requester, CID-derived commitment, deployment coordinates, and deterministic grant identifier supplied to the release service (schema version 2) are the values that the issuer and requester independently expect.

The last two assumptions define the trusted-authorization boundary. In the inherited prototype, requester and record/grant identity are not public inputs to the Groth16 statement, the event accepts a separately supplied `recordIdHash`, and the circuit's equality expression is not a general field-equality test. The release protocol can reject divergence from the event and expected context, but it cannot distinguish a semantically valid event from a malicious event that the inherited layer nevertheless accepts. A mined receipt on the ephemeral local chain also does not establish public-chain finality.

The two replay indexes introduced in this study operate on releases and accepted event contexts. They do not implement proof-level nullifiers or revoke previously issued credentials. If the inherited layer accepts a repeated proof in a new transaction and emits a new event, that event can identify a new grant. The protocol therefore cannot compensate for missing proof-level replay protection. To compose authorization with release in a deployment, the upstream assertion must bind the approved requester and record/key version and prevent proof reuse when the access policy requires it; a proof-level nullifier is one possible mechanism. These are requirements on that interface, not repairs implemented in the inherited circuit.

3.4. Security Goals and Explicit Non-Goals

Within these assumptions, the design targets five properties. First, every release is authenticated by the expected authority before any requester private-key operation. Second, the signed artifact and derived wrapping inputs bind the release to its profile, algorithms, authority, recipient, event, grant, record commitment, and validity interval. Third, two replay indexes enforce one successful CEK return per release and per finalized grant within the trusted state store. Fourth, current trusted capability state selects exactly one migration mode and an error never triggers fallback. Fifth, the issuer records direct-RSA exposure under the literal record commitment, and an optional inventory gate uses that record. The signed mode and its result label describe how the key was established; they do not certify the CEK's complete exposure history to the requester.

Single use serves grant-level accountability: one accepted event permits at most one successful key return within the trusted store. It does not prevent an authorized recipient from retaining the delivered CEK, nor does it provide forward secrecy.

These goals are examined by symbolic models and implementation tests under explicit assumptions. Named key-disclosure and stale-policy models examine selected assumption failures; they do not model complete endpoint compromise. Compromised hospitals, registry administrators, clocks, or implementation hosts remain outside the protection claim. The study also excludes upstream authorization and ledger issues (blockchain consensus or governance failure, Groth16 setup, and authorization repair); privacy, memory-handling, and side-channel properties (metadata privacy, traffic analysis, certified erasure, and side-channel certification); operational resilience (arbitrary storage corruption, distributed crash recovery, Internet-facing abuse controls, and deployment-scale concurrency); production or clinical certification; and experiments executing ML-KEM on quantum hardware or demonstrating quantum advantage. These exclusions limit the analysis to off-chain key release.

4. Authenticated Key-Release Protocol and Migration Design

The design links three decisions that must remain consistent during migration: which authority may release a key, which accepted request the release serves, and which cryptographic mode the policy permits. A signed artifact binds these decisions to one requester and record. The requester authenticates the artifact before comparing its expected context and processing replay state; it returns the CEK only after the single-use state has been consumed.

4.1. Notation and Cryptographic Suite

Let JCS(x) denote the RFC 8785 canonical 8-bit Unicode Transformation Format (UTF-8) encoding of x [15]; H denote SHA-256; $\|$ denote byte concatenation; and $0x00$ denote one zero byte. `CEK` is the inherited 32-byte record content key, ss_K is the 32-byte ML-KEM shared secret, and r_R is a fresh 32-byte RSA contribution used only in hybrid mode. Additional

authenticated data (AAD) is the context-bound, authenticated non-ciphertext input to AES-256-GCM. Table 3 gives the only accepted suite.

Table 3. Cryptographic suite for authenticated key release.

Purpose	Wire Identifier	Exact Construction
Profile and canonicalization	`ZKEHR-KR-v2-MLKEM768-MLDSA65; JCS-RFC8785`	Schema version 2; closed, mode-discriminated canonical JSON.
Post-quantum key establishment	`ML-KEM-768`	FIPS 203 parameter set; 1184-byte public key, 1088-byte ciphertext, and 32-byte shared secret [11].
Key derivation	`HKDF-SHA-256`	RFC 5869 extract-and-expand, 32-byte salt and output, with fixed domain-separated `info` [18].
CEK wrapping	`AES-256-GCM`	32-byte key, 12-byte nonce, context AAD, and 16-byte authentication tag.
Authority authentication	`ML-DSA-65`	Mandatory pure randomized FIPS 204 signature in every mode; 1952-byte public key and 3309-byte signature [12].
Legacy transport	`RSA-OAEP-SHA-256`	RSA-2048, exponent 65537, SHA-256, MGF1-SHA-256, and an empty OAEP label.

The suite admits no aliases, implicit defaults, or fallback. A future suite requires a new profile identifier. The implementation rejects unknown profiles and supports key retirement and revocation within the current profile. Communicating profile deprecation to deployed requesters would require authenticated administrative configuration, an explicit local allowlist or minimum-version policy, and an update mechanism. These are deployment requirements; profile-revocation distribution and bounded propagation are not implemented here.

4.2. Artifact Encoding and Processing

A release is a canonical JSON object with required common fields and one closed mode branch. Appendix A defines the fields and encoding constraints. The requester rejects non-canonical bytes rather than normalizing them, ensuring that the authenticated and measured representations coincide.

The issuer validates the trusted event and record binding, records one clock reading, acquires one coherent registry snapshot, and resolves the expected authority before selecting the requester mode and active keys. It then reserves a fresh UUIDv4 release identifier, generated from a separate 16-byte operating-system random draw. A collision or uncertain insertion aborts without retry. In legacy mode, it persists and reads back the record's exposure marker before encryption or publication.

Salts, nonces, and hybrid contributions use separate exact-length operating-system random draws. The issuer executes one release mode, signs the complete artifact, validates it, and verifies its own signature. Publication succeeds only when the stored state is `I_PUBLISHED` (successful immutable publication) and readback matches the exact canonical bytes. This readback can reconcile a lost acknowledgement. Otherwise, the service returns no successful issuance, attempts to record `I_ABORTED` (terminal failed issuance),

and never reuses the release identifier or clears exposure. An uncertain committed write is not assumed to have been rolled back.

The requester follows a fixed sequence: wire admission; clock reading and registry snapshot; candidate issuer-key resolution; signature verification; expected-context comparison; requester policy; time policy; paired replay reservation; private-key operations; paired consumption; and CEK return. The compact validation table in Appendix A records the corresponding stages. A candidate key named by an artifact cannot override expected authority policy: even a valid alternate-authority signature fails context comparison. A known mode mismatch defers only the mode-specific key-set decision to the policy stage.

4.3. Authentication and Context Binding

The mode-selected context C_m contains the schema, profile, mode, canonicalization and applicable algorithm identifiers; release identifier and validity window; issuer and signing-key identifiers; requester address and applicable key identifiers; chain and contract; transaction and log; access-grant identifier; and record commitment. It excludes the cryptographic payload and `authority_signature`. Define

$$d_C = H(\mathrm{ASCII}(\texttt{ZKEHR-KeyRelease-v2/context}) \| 0x00 \| \mathrm{JCS}(C_m)), \tag{1}$$

$$\mathrm{info} = \mathrm{ASCII}(\texttt{ZKEHR-KeyRelease-v2/wrap}) \| 0x00 \| d_C, \tag{2}$$

$$\mathrm{AAD} = \mathrm{ASCII}(\texttt{ZKEHR-KeyRelease-v2/aad}) \| 0x00 \| d_C. \tag{3}$$

The signature input is

$$\mathrm{ASCII}(\texttt{ZKEHR-KeyRelease-v2/signature}) \| 0x00 \| \mathrm{JCS}(A \setminus \{\texttt{authority_signature}\}), \tag{4}$$

where A is the complete artifact. Unlike C_m, it includes the ciphertexts, salt, nonce, and wrapped CEK. Pure randomized ML-DSA-65 uses the additional FIPS 204 context bytes `ZKEHR-KeyRelease-v2`. The wire field `kdf_info` is the fixed ASCII identifier `ZKEHR-KeyRelease-v2/wrap`; the actual derivation input also includes the zero byte and d_C in Equation (2).

The inherited binding is derived independently at both ends:

$$\texttt{record_commitment} = \texttt{0x} \| \,\mathrm{hex}\big(\mathrm{Keccak256}_{Ethereum}(\mathrm{UTF8}(CID))\big), \tag{5}$$

$$\texttt{access_grant_id} = \texttt{0x} \| \,\mathrm{hex}\big(H(\mathrm{ASCII}(\texttt{ZKEHR-KeyRelease-v2/grant-id}) \| 0x00 \| \mathrm{JCS}(G))\big). \tag{6}$$

Here, G contains exactly `chain_id`, `contract_address`, `log_index`, and `transaction_hash`. The adapter uses Ethereum Keccak-256, not NIST SHA3-256, and recomputes the commitment from the literal CID text, bounded to 256 UTF-8 bytes. It does not accept a precomputed hash or normalize equivalent CID versions or multibase spellings; both parties must use the same text.

4.4. Expiry, Replay, and Failure State Machine

The issuer emits whole-second UTC timestamps and a default lifetime of 300 s. At reception, a lifetime that is non-positive or greater than 300 s violates the lifetime policy and returns `EXPIRED`. Issuance later than `now + 30 s` is not yet valid, while `expires_at` $\leq$ `now` is expired. The same immutable clock snapshot governs key-activation windows, these comparisons, and replay reservation time; future skew never extends expiry. The 30 s allowance is a fixed prototype policy choice, not an empirically calibrated clock bound. Deployment would require an operational clock policy and evaluation of its effect on admission.

Replay state uses two independent uniqueness indexes:

$$K_R = (\texttt{issuer_id}, \texttt{release_id}),$$
$$K_G = H(\mathrm{ASCII}(\texttt{ZKEHR-KeyRelease-v2/grant}) \| 0x00 \| \mathrm{JCS}(G_R)),$$

where G_R contains exactly the access-grant identifier, chain, contract, transaction, log, record commitment, and requester address. It excludes the release identifier, key identifiers, and algorithms, so a fresh artifact identifier, key rotation, or mode change cannot reuse the same finalized grant.

Figure 2 shows the paired replay-state transitions.

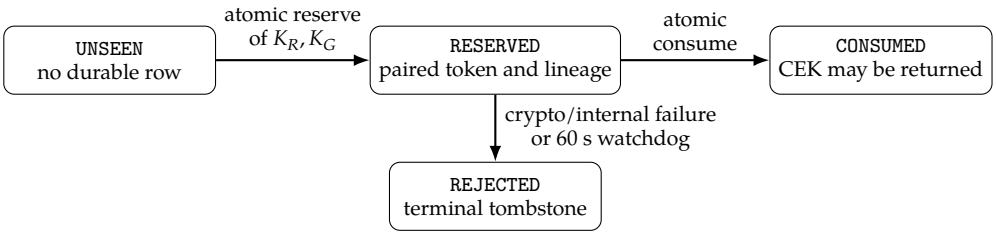

Figure 2. Paired replay-state transitions under trusted atomic store semantics.

Both indexes move in one transaction under the same unpredictable token, context digest, partner reference, and lineage. An existing coherent pair is `REPLAY_DETECTED`; a physically orphaned or internally inconsistent pair is `REPLAY_STATE_FAILURE`. Post-reservation cryptographic failure sets both rows to `REJECTED`. A process left in `RESERVED` is tombstoned after 60 s, never reopened. Consume uncertainty returns no CEK.

Public errors identify the failed validation category without exposing secrets or component-specific cryptographic detail. The archive specifies the exact rejection codes asserted by the tests.

4.5. Registry Policy and Downgrade Control

Each registry entry binds a key to its owner, identifier, algorithm, status, and activation window. An operation uses one immutable, version-coherent snapshot; stale compare-and-swap updates fail. Retirement and revocation are terminal, while rotation creates a new entry and atomically updates the trusted pointer. Administrative capability state determines the signed mode as shown in Table 4.

Table 4. Requester capabilities and corresponding key-release modes.

Capability State	Derived Wire Mode	Exact Active Requester Keys	Result Label
`legacy-only`	`legacy-rsa`	One RSA-2048 key; no active ML-KEM release key	`classical-legacy-confidentiality`
`hybrid-required`	`hybrid-rsa-mlkem`	One RSA-2048 key and one ML-KEM-768 key	`hybrid-both-required`
`pq-only`	`mlkem-only`	One ML-KEM-768 key; every RSA release key retired or revoked	`pq-key-establishment-authenticated`

Allowed state transitions are `legacy-only` → `hybrid-required` → `pq-only`, with a direct transition from `legacy-only` to `pq-only`. A same-state transaction is reserved for

atomic key rotation. Backward transitions are forbidden. Issuance exposes no caller-controlled mode, algorithm, fallback, or "best available" flag. On reception, signed `legacy-rsa` under either migrated state is `DOWNGRADE_DETECTED`; another known state/mode mismatch is `CAPABILITY_MISMATCH`; unknown identifiers fail in the earlier wire phases. No rejection selects another engine.

4.6. Legacy, ML-KEM-Only, and Hybrid Mode Semantics

In `legacy-rsa`, RSA-OAEP encrypts the CEK directly. Mandatory ML-DSA still authenticates the artifact, but it cannot change the classical confidentiality of the RSA-wrapped CEK. The exact record commitment is marked `legacy_exposed` before the release can be published.

In `mlkem-only`, the issuer encapsulates to the active registered ML-KEM key, derives

$$K_W = \text{HKDF-SHA-256}(\texttt{salt}, ss_K, \text{info}, 32),$$

and applies AES-256-GCM to the CEK with the artifact's fresh 12-byte nonce and the AAD from Equations (2) and (3). This protects the release under the stated ML-KEM and endpoint assumptions, without changing prior exposure.

In `hybrid-rsa-mlkem`, RSA-OAEP encrypts the independent contribution r_R, never the CEK. The exact 99-byte HKDF input key material is

$$\text{ASCII}(\texttt{ZKEHR-KeyRelease-v2/hybrid-ikm}) \| 0x00 \| \text{uint16be}(32) \| ss_K \| \text{uint16be}(32) \| r_R. \quad (7)$$

Here, uint16be encodes a length as an unsigned two-byte big-endian integer. Both contributions are therefore required to derive K_W, and AES-GCM wraps the CEK exactly once. This is the application-specific ZK-EHR dual-contribution hybrid, not a standardized composite KEM. Its symbolic key-disclosure results are given in Section 7.2. Before either RSA decryption or ML-KEM decapsulation, the requester verifies ML-DSA over the entire artifact, including both component ciphertexts. Since these ciphertexts and their public keys are absent from the KDF input, this ordering is essential to payload binding. RSA-OAEP decryption or AES-GCM authentication failure returns `CRYPTOGRAPHIC_REJECTION` without fallback. ML-KEM uses implicit rejection rather than an explicit ciphertext-authentication verdict; an incorrect resulting secret cannot authenticate the CEK wrapper [11]. Unexpected provider or operation faults are sanitized as `INTERNAL_FAILURE`.

4.7. Scoped Historical Inventory and Admission

Direct-RSA issuance writes and reads back a monotonic exposure marker before wrapping or publication. A missing marker does not establish that no earlier release occurred. An optional local inventory gate therefore accepts a trusted operator assertion for an exact record commitment: the evidence source, whether direct RSA was observed, whether that source is complete for the commitment, and whether the same CEK was independently wrapped under both mechanisms. The core requester does not consult this history store, so its mode label is not a historical confidentiality verdict.

Complete scoped evidence with no direct RSA and no existing marker permits an unexposed classification within that scope. Known RSA exposure remains exposed. Missing or incomplete history blocks admission unless the operator explicitly chooses conservative backfill, which marks and reads back exposure before permitting an exposed release. Same-CEK dual wrapping retains its unsafe classification. A later negative inventory assertion cannot clear known exposure.

The gate closes admission before reconciliation and activates it only after successful persistence and exposure readback. Every new gate instance requires reconciliation; stored

rows alone cannot reactivate admission. The gate binds issuance to the same validated event and CID. It is an external wrapper around the core issuer, so applications must use it; it does not prevent direct API bypass or provide a distributed two-store transaction. The synthetic evaluation in Section 6.3 tests these local rules, not the truth or global completeness of an institutional inventory.

Historical classification also requires a consistent record-key identity. Equivalent CID text encodings can yield different commitments in Equation (5), although they address identical bytes under the same CEK. An application claiming key-wide history must retain one authoritative locator per CEK/version or carry the key's history across aliases. The release layer does not enforce that mapping or normalize aliases silently. Section 6.4 evaluates this boundary; a complete-history assertion for one textual commitment alone cannot establish the absence of RSA exposure through another.

A new protected record version requires retiring legacy release capability and generating a fresh CEK, nonce, tag, ciphertext, CID, and commitment with a newer version. That full migration workflow and corpus-scale re-encryption were not evaluated; older exposure entries and harvested material remain relevant.

5. Materials and Methods

The evaluation separates functional integration, rejection and failure behavior, symbolic properties, and service performance. Test methods, generated examples, and evidence mappings are different units and are not added together.

5.1. Implementation and Runtime

A Node/Python adapter connects the retained authorization and record workflow to the authenticated release service. It supplies trusted event and record context and returns the recovered CEK to the record decryptor. The three release modes use ML-DSA-65 signatures, registry policy, issuance state, direct-RSA exposure marking, and paired replay indexes. A separate admission wrapper evaluates historical inventory; it is not part of the timed service.

The native performance experiment used an Apple M4 Pro with 14 logical CPUs and 64 GiB memory, connected to AC power, running macOS/Darwin 25.5.0. CPython 3.12.4 ran natively on arm64 with `cryptography` 48.0.0 and OpenSSL 4.0.0. Both `liboqs` and its Python wrapper used version 0.16.0. The `liboqs` build disabled its optional OpenSSL integration. Open Quantum Safe supplies ML-KEM-768 and randomized ML-DSA-65; `cryptography` supplies RSA-OAEP, HKDF-SHA-256, and AES-256-GCM. Runtime checks reject missing or mismatched mechanisms without automatic provider installation. Background activity, thread quality of service, core placement, frequency, and thermal state were not continuously recorded or controlled.

Node.js 22.10.0, npm 10.9.0, Hardhat 2.28.6, and ethers 6.16.0 support separate local integration tests, without a public RPC endpoint or live IPFS network. Mutable CEKs and intermediate buffers are overwritten where possible; immutable Python values and native-provider copies preclude certified erasure.

5.2. Integration, Generated Inputs, and Failure Tests

Synthetic records are encrypted under fresh CEKs and associated with matching mined `AccessGranted` events on an ephemeral Hardhat chain. The adapter checks deployment, requester, receipt coordinates, and CID-derived commitment before release. Successful recovery requires CEK equality, ciphertext-digest and AES-GCM verification, and exact recovered plaintext and source-digest equality.

The integrated Python suite contains 242 test methods: 219 in the core suite and 23 distributed across service-failure (six), property-based (eight), and history-workflow (nine) modules. Its 602 successful subtests are nested executions, not additional independent experiments. A source-and-execution audit identifies 50 methods that invoke at least one actual provider KEM, signature, key-derivation, encryption, or decryption operation, either directly or through a helper. The count excludes methods whose only provider operation is class-wide key generation, hash/parser-only cases, rejection before a primitive, and fake-provider-only paths. A method combining real operations and injected behavior counts once. The per-method ledger is included in the Supplementary Verification Materials. Hypothesis 6.170.1 generated 64 examples in each of eight properties (512 total), using recorded seeds and no example database. These properties cover Base64url round trips, a bounded flat-JSON canonicalization subset, non-canonical member ordering, cross-mode fields, validly signed context substitutions, ciphertext modification, validation precedence, and oversized inputs. They do not exhaust JSON or cryptographic input spaces.

The six service methods record 39 scenarios. These include two-thread duplicate-delivery races with in-memory and SQLite stores, two-process SQLite races, distinct release identifiers for one grant submitted sequentially, injected acknowledgement loss after consumption, and process termination after committed reservation or consumption. Process-failure tests kill only the test worker. Reservation-timeout checks advance a controlled clock; they do not simulate power loss or distributed storage.

The nine history methods record 31 scenarios using trusted, record-scoped inventory assertions. They cover known RSA exposure, complete clear histories, missing or incomplete histories, conservative backfill, same-key dual wrapping, monotonic markers, context substitution, restart reconciliation, and write/read/acknowledgement failures. The gate wraps the core release API and can be bypassed by a caller that does not adopt the wrapper.

Table 5 distinguishes the accounting units. The 33 negative-control categories and 53 mappings for 19 claims have executed owners; mapping rows are not additional experiments. Of 55 negative wire fixtures, 49 are wire-owned and six require context/time service checks. The migration matrix's 116 rows, 22 requirements, and 74 identifiers provide structural traceability. Some inherited controls use doubles, whereas the signed-context and ciphertext properties use the actual providers. The Mendeley Data archive maps categories, wire fixtures, and claims to their owning tests; the Supplementary Verification Materials contain the per-method real-provider ledger.

Table 5. Evidence units in the integrated implementation test suite.

Unit	Count	Interpretation
Python test methods	242	One declared executed method per count.
Real-provider method subset	50	Actual operation on a tested body/helper path; subset of 242.
Nested successful subtests	602	Parameterized executions inside methods, not added to 242.
Generated property examples	512	Eight properties with 64 examples each; overlap methods/subtests.
Service/history scenarios	39/31	Logged outcomes from six/nine methods, respectively.
Claim mappings/wire fixtures	53/55	Traceability links and inputs, not independent method counts.
Migration-matrix rows	116	Structural checks; no claim of 116 cryptographic executions.

5.3. Grant Contention and CID-Identity Tests

A three-method test module evaluates grant contention and CID identity using the release implementation and native runtime described above. It contains 18 thread races (three modes, memory or two SQLite connections, three repetitions), nine two-process SQLite races (three per mode), and two backend executions of an 11-step CID diagnostic. Each race uses two valid artifacts with different release indexes but the same grant index and CEK. Coordination holds the first memory lock or SQLite write transaction until the competitor attempts reservation; SQLite trace callbacks identify entry into `BEGIN IMMEDIATE`. The leading artifact alternates across repetitions. Observers check key equality, one mode-specific recovery-engine invocation, loser rejection/retry, terminal state, and SQLite reopen. Invocation counts do not count the individual RSA/ML-KEM primitives inside hybrid recovery. These are bounded controlled schedules, not exhaustive interleaving coverage.

The CID diagnostic uses one encrypted synthetic content object and two valid multibase representations of identical CIDv1/raw/SHA-256 bytes. It checks separate commitments, real RSA exposure and ML-KEM recovery of the same CEK, unknown versus asserted-complete inventory, correct history backfill, context/signature rejection, and a fresh-key/content control. Its 22 dependent scenario records are not 22 independent tests. Regression checks re-executed 63 methods selected from the integrated suite in the same native runtime, covering replay/crash, inventory, binding, registry, and migration-service behavior. These counts overlap the integrated 242-method suite and are reported separately, not added to its method count.

5.4. Paired Service Benchmark

Five sequentially restarted native processes each generated a fresh requester/authority key set. Within a process, the same keys were used for six paired cells: three modes under per-call RSA loading and validated preloading. Each cell retained 100 warm-ups and 1000 primary observations, giving 30,000 measured primary releases. Separate instrumented calls provided 300 diagnostic observations per cell and 100 diagnostic warm-ups, giving 9000 measured diagnostic releases and 6000 warm-ups across both streams. Primary summaries never pool diagnostic observations.

The per-call path imports PKCS #8 DER using `load_der_private_key(encoded, password=None)`, retaining its default RSA key validation, then re-serializes and compares canonical bytes, validates modulus and exponent, and decrypts using RSA-OAEP. In the preload condition, the harness performs these key-admission checks once before timing and accepts only the identical DER bytes and validated provider object thereafter. Every release still undergoes signature, expected-context, policy, time, and replay checks. ML-KEM-only performs no RSA operation and serves as a loading-condition control. Preloading is an experimental condition, not a change to the protocol's validation requirements.

Services persist within each process. Each sample receives fresh synthetic event/CID context, CEK, and in-memory state outside the timer. Issuer timing begins at trusted-context binding and ends after publication-state verification; requester timing covers expected-context binding through recovered-key return. An independent outer timer measures their total. Key generation, preload admission, fixture/store creation, cleanup, and CEK comparison are excluded. Instrumented calls separately time RSA import (including provider validation), explicit validation, decryption, ML-KEM decapsulation, derivation, unwrapping, signature verification, and remaining requester work.

Randomized blocks interleave the six primary cells and, during the first 300 measured blocks, the six diagnostic cells. Recorded process-specific seeds control order and public fixtures only; cryptographic randomness remains independent. No timed observation traverses Node, blockchain, IPFS, a network, record decryption, or durable storage, and there

is no concurrent load. A single exploratory run of 3000 releases under x86-64/Rosetta also bypassed Node; its results are reported separately in the Supplementary Materials and are not pooled with the native measurements.

RSA-2048 and ML-KEM-768 are operational choices, not matched security levels. Table 6 makes the distinction explicit. The inherited publication did not specify its RSA modulus length; the present RSA comparator belongs to the new authenticated service, not to a timing reproduction of that earlier system.

Table 6. Security-level context for the selected mechanisms.

Mechanism	Stated Security Context	Role in This Experiment
RSA-2048	Approximately 112-bit classical strength; not post-quantum	Legacy CEK transport or hybrid contribution
ML-KEM-768	NIST security category 3	Key encapsulation
ML-DSA-65	NIST security category 3	Authority signature in all modes

RSA strength follows NIST SP 800-57 Part 1 Rev. 5 [34]; category assignments follow FIPS 203 and 204 [11,12]. Classical bit strength and post-quantum categories are not interchangeable. The comparison is operational, not security-strength-matched.

5.5. Statistical Analysis and Size Accounting

Durations are recorded in integer nanoseconds. For each process/cell, analysis retains the median, quartiles, IQR, p95, mean, minimum, and maximum, using Hyndman–Fan type 7 quantiles. Main point estimates are medians of five process-level medians; displayed ranges are minima and maxima of those five medians, not confidence intervals. Supplementary Tables retain process-level distribution summaries.

Loading effects are calculated as differences and ratios of paired cell medians within each process and then summarized across the five pairs. Their displayed spread is the observed minimum–maximum of those five paired values. A percentile-bootstrap analysis uses 20,000 process-pair resamples. Exact enumeration of all 3125 ordered resamples confirms that, in these data, the 2.5th and 97.5th percentiles equal the observed extremes. The reported ranges therefore describe the observed spread rather than confidence intervals with calibrated 95% coverage. The sampling unit is the restarted process, not an individual release, and no cross-host or provider variability is estimated. The requester percentage is calculated for each observation before the two levels of median aggregation; it is not obtained by dividing separately summarized medians.

All measured rows are retained without trimming or winsorization. Warm-ups remain in the evidence but are excluded from summaries. Instrumentation sensitivity compares diagnostic and primary calls over their common first 300 blocks; these are separate calls, not paired measurements of exact instrumentation overhead. A separate Python-standard-library implementation, without importing the analysis module, reproduced summaries and paired effects from the recorded rows and enumerated the bootstrap resamples. This is a computational cross-check, not an independent host experiment. Key-generation/export measurements, reported in Supplementary Section S2.1, are descriptive and exploratory. Their schedule-position sensitivity remains unexplained and was not evaluated in an independent key-generation experiment. These measurements do not inform comparative performance conclusions.

Wire accounting separates the canonical transmitted artifact from public keys stored in the registry, endpoint private keys, and authority material. Payload, Base64url expansion, and remaining JSON bytes sum to wire size. Storage and transmission quantities are not added into a single deployment cost.

5.6. Reproducibility Materials

The Mendeley Data archive at https://doi.org/10.17632/88gk36mgkk.2 provides source code, the protocol specification, synthetic fixtures, sanitized observations, and analysis tools for the native paired benchmark, symbolic models, integrated test suite, and history wrapper. Exploratory x86-64/Rosetta and key-generation observations are identified separately within the archive. Code and records for the distinct-identifier races and CID diagnostics, together with per-method and statistical audit ledgers, are supplied in the Supplementary Verification Materials (`Verification_Materials.zip`). These files are distributed separately from the Mendeley archive.

Outputs exclude private keys, CEKs, shared secrets, replay tokens, plaintext, credentials, host identifiers, usernames, and absolute paths. No patient data were used. Dependencies, commands, expected outputs, and checksums distinguish recorded results from subsequent reproductions.

6. Results

The evaluation combines successful synthetic integration, bounded rejection and failure tests, a scoped historical-inventory gate, and a five-process service benchmark. Results from the integrated suite and the grant-contention/CID module are reported separately using their respective accounting units. Symbolic results are reported in Section 7.

6.1. Integration and Rejection Outcomes

The integrated Python suite passed all 242 methods and 602 subtests, with no failures, errors, or skips. Of these methods, 50 meet the real-provider-operation definition in Section 5.2. All 512 generated examples passed their property assertions. These totals describe bounded executed tests, not exhaustive attack coverage.

Five separate Node/Hardhat methods passed. Two methods contain four real release round trips: two ML-KEM-only, one legacy, and one hybrid. Two other methods reject an invalid proof or mismatched event/CID before invoking the bridge; the ciphertext-tampering method uses a bridge double to isolate record-integrity validation. Thus, five methods are not five full cryptographic workflows. The successful round trips recovered the expected CEKs and plaintext under the trusted-authorization assumption in Section 3.3.

Malformed, non-canonical, mixed-mode, and unsupported-profile objects failed their admission gates. Signed-field modification failed signature verification before requester private-key operations. Validly signed substitutions of the expected context failed context comparison; a legacy artifact under migrated capability state failed downgrade checking. Incorrect recipient keys or hybrid contributions produced uniform cryptographic rejection without fallback. Exact-expiry and replay cases failed before private-key processing.

The evidence audit reconciled all 53 claim mappings and 55 negative wire fixtures with their owning executions. The 116-row migration matrix passed structural validation only. Table 7 distinguishes executed evidence from assumptions and remaining gaps.

Table 7. Security properties and supporting evidence.

Property	Executed Positive/Negative Evidence	Structural Evidence	Not Established
Key recovery	Four real adapter round trips; selected wrong-key/wrapper rejections	Claim-owner mapping	Upstream proof soundness or clinical validity
Signature and context	Valid releases; real signed-context substitutions and unsigned payload/precedence checks	Stage mapping	Exhaustive inputs or code-to-model equivalence
Wire admission	Positive vectors; 49 wire-owned negative fixtures and bounded generated inputs	Six additional fixtures defer to service checks	Exhaustive parser/JCS fuzzing

Table 7. *Cont.*

Property	Executed Positive/Negative Evidence	Structural Evidence	Not Established
Policy and time	Valid modes; expiry/future-time and downgrade rejection; executed forward transition, rotation, stale-CAS and backward-transition checks	116-row migration matrix	Global freshness or 116 executed scenarios
Single-use state	Duplicate races; 27 coordinated distinct-ID races; sequential, loss and termination checks	Paired-state transitions	Exhaustive schedules, power loss, or distributed recovery
Hybrid recovery	Both contributions recover one CEK; selected failed components reject without fallback	Fixed schema and binding map	Generic combiner theorem or forward secrecy
Exposure history	Clear/exposed admissions; unknown-history/fault rejection; CID-alias diagnostic and correct-history backfill	Scoped operator inventory	Automatic key-wide alias history, complete hospital history or full re-encryption
Symbolic properties	Conditional agreement/secrecy for an external adversary and one honest recipient; expected compromise/control traces	Manual implementation mapping	Malicious registered insiders, computational reduction or model composition

Positive/negative entries describe selected executed behavior. Some inherited controls use doubles, as distinguished in the evidence mapping. Counts overlap; structural mappings are not additional tests. Symbolic reachability and attack traces are not runtime test cases.

6.2. Replay and Failure Behavior

All 18 two-thread duplicate-delivery races (three modes, two stores, three repetitions) and three two-process SQLite duplicate-delivery races in the integrated suite returned one CEK and one replay rejection per pair. Observers recorded one mode-specific recovery-engine invocation and a consumed pair. Six sequential cases with different release identifiers for the same grant rejected the second release.

Distinct signed artifacts for one grant were also tested under coordinated contention at reservation. All 27 coordinated attempts passed: nine memory-thread, nine two-connection SQLite-thread, and nine two-process SQLite cases. These cases covered all three modes with three repetitions. Each returned exactly one matching CEK and rejected the other artifact at replay reservation, with one recovery-engine invocation overall. The losing release index was absent, its retry remained rejected, and SQLite reopening preserved the consumed grant/winner pair. The three-method module passed with 29 nested subtests, comprising these 27 contention cases and two CID-diagnostic chains. The 63 selected regression methods passed with 192 nested subtests and no failures, errors, or skips. Regression counts overlap the integrated suite and are not added to its totals.

In six injected acknowledgement-loss cases, the store had already committed consumption. The requester returned no CEK, cleared its owned candidate buffer, and left the pair consumed; retry performed no private-key operation. Three workers terminated after reservation left a committed reservation that became rejected at the controlled 60 s boundary, without reopening. Three workers terminated after consumption returned no CEK and left consumed state. These six process terminations test local committed-state behavior, not power-loss durability, arbitrary rollback, or distributed exactly-once execution.

6.3. Historical-Inventory Admission

All 31 recorded history scenarios passed. A complete clear scoped inventory admitted the record as not legacy-exposed; known RSA history remained legacy-exposed. Missing or incomplete history blocked admission unless an explicit conservative backfill acknowledgement marked it exposed. Same-CEK dual wrapping retained a non-PQ-safe classification. These labels refer to the accepted inventory and recorded release history, not independent evidence that an institution supplied a complete history.

Fault cases prevented publication when marker writes, readback, or acknowledgement failed. Reconciliation invalidates previous admission before updating stores, and restart requires fresh reconciliation rather than trusting stored admission rows alone. A substituted

record commitment was rejected against the actual trusted event binding. These results establish a small fail-closed workflow under trusted inventory inputs. They do not establish atomicity across both databases, mandatory use by every caller, distributed reconciliation, or corpus-scale migration.

6.4. CID-Alias Diagnostic

For each exposure-store backend, two multibase representations decoded to identical CID bytes and the same encrypted-content digest, but produced different textual commitments. A real RSA release marked the first commitment only. Unknown history for the alias blocked issuance; an asserted-complete, RSA-clear history limited to that alias admitted a real ML-KEM release of the same CEK as `not-legacy-exposed`. Supplying the RSA history for the alias instead produced `legacy-exposed`. These observations expose a gap between record-key identity and inventory scope: completeness for one textual commitment is insufficient for a key-wide non-exposure claim.

The signature and expected-context boundaries held. A modified unsigned commitment failed authority authentication, and a valid artifact for the other commitment failed expected-context comparison, both before requester private-key operations. A fresh-key/content version was a separate positive control and did not clear earlier markers. In the evaluated scenario, misclassification occurred when a valid accepted context was combined with a trusted clear-history assertion that omitted the same-key alias history. The 22 dependent scenario records across memory and SQLite do not establish an attack by an external adversary lacking those trusted inputs. The diagnostic does not implement a canonical identity map or a full record-migration workflow.

6.5. Service Latency and Loading Effects

All 30,000 primary and 9000 diagnostic measured releases succeeded, with no excluded rows. Table 8 reports the six primary cells. Per-call loading gave median process-level total latencies of 48.501, 1.507, and 48.824 ms for legacy, ML-KEM-only, and hybrid, respectively. Validated preloading reduced the RSA-bearing totals to 2.085 and 2.404 ms; the ML-KEM-only control was 1.495 ms.

Table 8. Issuer and requester service latency (ms).

Mode	Loading	Issuer	Requester	Total	Total Range
Legacy	Per-call	0.859	47.646	48.501	48.150–49.550
Legacy	Preloaded	0.881	1.195	2.085	2.077–2.107
ML-KEM	Per-call	0.953	0.545	1.507	1.489–1.520
ML-KEM	Preloaded	0.950	0.541	1.495	1.487–1.526
Hybrid	Per-call	1.052	47.754	48.824	48.511–49.920
Hybrid	Preloaded	1.071	1.309	2.404	2.382–2.425

Point estimates are medians of five process-level medians, with 1000 primary observations per cell/process. Ranges are the minimum and maximum process medians, not confidence intervals. Components are summarized separately and need not sum to the total.

Each joined pair in Figure 3 represents two medians from one process. The same five processes contribute to every mode. The paired loading effects in Table 9 show that the large RSA-bearing gap is strongly affected by key loading and reuse. Among preloaded cells, the median paired legacy/ML-KEM and hybrid/ML-KEM total-latency ratios were 1.389 (observed paired range 1.381–1.406) and 1.601 (1.589–1.609), respectively.

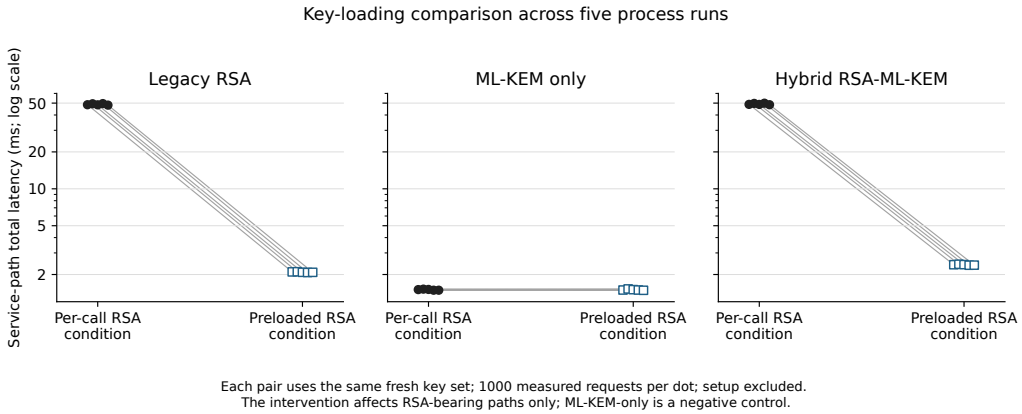

Figure 3. Service latency by key-loading condition (logarithmic scale). Points are process-level medians; each line joins the two loading conditions from the same process. All five process medians are plotted; some points overlap.

Table 9. Paired effects of RSA key-loading condition.

Mode	Difference (ms)	Paired Range	Ratio	Paired Range
Legacy	46.400	46.066–47.473	23.219	23.089–23.856
ML-KEM	0.002	−0.006–0.012	1.002	0.996–1.008
Hybrid	46.419	46.125–47.538	20.338	20.241–20.961

Difference: per-call minus preloaded; ratio: per-call divided by preloaded. Point estimates are medians and ranges are minima–maxima of five within-process contrasts. Archived percentile-bootstrap endpoints equal these extremes; the ranges do not assert calibrated 95% coverage or cross-platform uncertainty. Values are rounded once from the raw-row calculation. ML-KEM-only performs no RSA loading.

Requester work accounted for 98.24% of total latency in per-call legacy and 97.85% in per-call hybrid, falling to 57.88% and 55.23% under preloading. These percentages use per-observation shares and process-level medians, as specified in Section 5.5.

The diagnostic calls separate the RSA-bearing paths in Table 10. Most per-call requester time occurred inside provider key import and validation. RSA-OAEP decryption was also faster with the reused object. Other hybrid operations changed too: ML-KEM decapsulation was 74.6 versus 30.2 µs, HKDF 19.0 versus 9.5 µs, and AES-GCM unwrapping 15.2 versus 7.1 µs in per-call versus preloaded diagnostic cells. The experiment therefore does not isolate a cost attributable solely to DER parsing or RSA import; other provider and process effects were not separated. Component medians are not additive, and diagnostic calls do not replace the primary measurements.

Across the matched schedule window, diagnostic/primary total-median ratios ranged from 0.979 to 1.026; requester ratios ranged from 0.989 to 1.016. Late/early primary window ratios ranged from 0.940 to 1.020. These checks describe sensitivity in this campaign without correcting measurements or removing observations. Full component and process-level summaries and the separate exploratory release results appear in the Supplementary Materials. The exploratory key-generation/export measurements are reported in Supplementary Section S2.1.

Table 10. Requester RSA diagnostic intervals (ms).

Interval	Legacy Per-Call	Legacy Preload	Hybrid Per-Call	Hybrid Preload
RSA import/provider validation	45.395	—	45.212	—
RSA explicit validation	0.046	—	0.045	—
Validated-object lookup	—	0.001	—	0.001
RSA-OAEP decryption	1.876	0.731	1.876	0.731

Medians of five process-level medians from 300 separate diagnostic calls per cell/process. A dash denotes an operation absent from that path. Import includes provider-internal validation; explicit validation includes canonical-byte and profile checks. Intervals include their service adapter calls and are not a bare-primitive benchmark. Medians are not additive.

6.6. Operational Artifact and Key Sizes

For the recorded benchmark contexts and serialized keys, canonical artifacts occupy 5670 B for legacy, 7031 B for ML-KEM-only, and 7488 B for hybrid. Table 11 separates transmitted bytes from requester key storage: the "Public key" and "Private key" columns refer to registry and endpoint storage, respectively, and neither is included in the transmitted release size ("Wire"). Metadata lengths and RSA serialization may change these totals. Authority public and secret keys occupy 1952 and 4032 bytes in all modes. Transport framing, on-chain metadata, and deployment storage overhead are not measured.

Table 11. Release-artifact and requester-key sizes (bytes). "Public key" and "Private key" denote registry and endpoint storage, respectively; neither is included in the transmitted release size ("Wire").

Mode	Wire	Payload	Encoding	JSON	Public Key	Private Key
Legacy	5670	3565	1189	916	294	1217
ML-KEM	7031	4489	1497	1045	1184	2400
Hybrid	7488	4745	1583	1160	1478	3617

Wire = payload + Base64url expansion (Encoding) + JSON metadata. Public key bytes are stored in the registry and private key bytes at the endpoint; neither storage column contributes to Wire. Hybrid key sizes combine the two requester keys. Authority keys are separate and identical across modes.

6.7. Application-Rule Comparison

Table 12 compares what the evaluated application layer supplies with what remains necessary when adopting HPKE with a PQ KEM, X-Wing, or a LAMPS composite. This is an architectural comparison based on the cited specifications, not a result from executing those alternatives. The signed envelope, independent expected context, and paired state form the reusable part; algorithm identifiers, ciphertext binding, key admission, and wire rules require a new profile.

Table 12. Qualitative transfer of application rules to alternative constructions.

Rule Evaluated Here	What Remains Applicable	What a Replacement Must Define
Accepted-event and expected-context binding	The application still identifies the approved recipient, record and grant.	Map these values to authenticated envelope/HPKE context inputs and compare independently expected values.
Paired release/grant single use	The two-index policy can remain outside the cryptographic primitive.	Preserve grant identity and atomic reserve/consume semantics under the new profile.
Registry mode and no fallback	Current trusted capability state remains the policy authority.	New algorithm/key identifiers, admissible states, key validation and migration transitions.

Table 12. *Cont.*

Rule Evaluated Here	What Remains Applicable	What a Replacement Must Define
Payload authentication	An external signature can bind the complete release artifact.	A PQ-KEM HPKE profile needs suitable sender authentication; ciphertext/key binding must follow the chosen construction.
Historical exposure	Changing a KEM does not remove a prior wrapper of the same CEK.	Trusted key/version identity, alias-aware history, and explicit inventory enforcement.

Sources: RFC 9180 and HPKE-PQ -05, X-Wing, and LAMPS -19 [14,21–23]. The application mapping is our analysis. It does not transfer proofs or measurements or claim a tested alternative implementation. X-Wing changes the traditional component to X25519; LAMPS includes RSA-OAEP options but uses its own combiner and profile.

7. Security Analysis

Symbolic models examine selected protocol properties under explicit assumptions; implementation tests examine concrete validation and state behavior. Neither substitutes for a computational reduction or a proof that the source code refines the model.

7.1. Symbolic Model and Scope

ProVerif 2.05, which supports automated symbolic protocol analysis [35], was used with perfect cryptographic constructors and an active external adversary controlling public transport. The cryptographic models fix one honest authority and recipient key set while allowing for unbounded replicated grant contexts and receives. They do not establish agreement against a malicious registered recipient or a multi-recipient insider. Each context distinguishes a target record/mode from other-record and other-mode releases carrying decoy CEKs. The expected authority, registry identities, and accepted authorization event are trusted inputs.

The campaign contains 34 models and 167 query results: 100 true and 67 false. There were no unresolved queries. All false results have reconstructed traces corresponding to the specified reachability, compromise, missing-check, or stale-policy cases. The model families comprise 24 mode/key-disclosure combinations, three post-delivery compromise cases, three removed-check controls, one stale-policy case, and three replay-store variants. The Supplementary Materials record their assumptions and query accounting.

Under the modeling assumptions stated at the beginning of this subsection, and in uncompromised configurations, the cryptographic models establish non-injective agreement on the signed authority/key identities, recipient/key identities, deployment/event/grant/record context, mode, release identifier, payload, and CEK. They also establish signature verification before private-key use. This agreement authenticates the trusted assertion, not the upstream authorization decision. Cryptographic models deliberately allow for replay; single-use behavior is analyzed separately. The symbolic results do not establish a generic hybrid-combiner theorem, computational IND-CCA security, forward secrecy, or security against malicious registered recipients.

7.2. Confidentiality and Key Compromise

Legacy protects the CEK directly with RSA-OAEP; ML-KEM-only derives an authenticated wrapping key from the encapsulated secret. The ZK-EHR dual-contribution hybrid requires both the ML-KEM secret and an independent RSA-protected contribution. Table 13 summarizes secrecy of an honestly issued target CEK under the modeled recipient-key disclosures.

Table 13. Symbolic confidentiality under recipient-key disclosure.

Disclosed Recipient Secret	Legacy	ML-KEM-Only	Hybrid
None	Secret	Secret	Secret
RSA only	Exposed	Secret	Secret
ML-KEM only	Secret	Exposed	Secret
Both	Exposed	Exposed	Exposed

Results assume ideal cryptography and no independent legacy wrapper of the target CEK. Hybrid secrecy under one disclosed key is a symbolic result for the modeled derivation, not a generic robust-combiner or computational IND-CCA theorem.

In the symbolic models, disclosure of the authority signing key alone preserves secrecy of a target CEK honestly issued before the signing-key disclosure, provided that the required recipient secrets remain protected, but breaks origin agreement in every mode: the attacker can sign a release carrying its own chosen CEK. This does not represent complete authority-host compromise, which may expose the original CEK directly. Post-delivery disclosure of the recipient secrets required for each mode enables recovery from retained releases. Static recipient keys therefore provide no forward secrecy.

The signature covers every artifact field except the signature itself, including both hybrid ciphertexts. The derivation does not independently bind the ciphertexts or component public-key bytes as the compared composite constructions do. Payload binding depends on trusted registry key resolution and successful ML-DSA verification before RSA decryption or ML-KEM decapsulation. The symbolic signing-key-compromise analysis does not establish a computational binding guarantee for this construction. The hybrid result also does not model arbitrary adversarially broken component algorithms or quantum-oracle attacks.

7.3. Expected Context and Concrete Validation

A valid signature alone does not identify the release a requester should accept. The requester independently supplies expected authority, recipient, deployment, event, grant, and record values. Context comparison rejects an authentic artifact for another request. Removed-check models produce the expected traces when signature, expected-context, or migration-policy gates are omitted, supporting their distinct roles.

Concrete parsing and canonicality checks precede issuer-key resolution and signature verification. The executed substitutions and unsigned ciphertext modifications were rejected before requester private-key processing. ML-KEM implicit rejection is handled by the subsequent AES-GCM check [11]; it is not a release-authenticity verdict. RSA-OAEP and wrapper failures become `CRYPTOGRAPHIC_REJECTION`, while unexpected faults become `INTERNAL_FAILURE`. The symbolic destructor abstraction is not a proof of equivalence to these provider and parser behaviors.

7.4. Replay, Expiry, and Downgrade

A separate replay model uses two arrivals, two release identifiers, and two grant identifiers with arbitrarily scheduled workers and an ideal atomic serialized store. Its two injective correspondences and consume-before-delivery property hold with both indexes; removing the release or grant index yields the intended reuse traces. A complementary invariant argument is conditional on monotonic state and atomic reservation: successful reservation requires both indexes to be absent, and terminal pairs are never reopened. This supports at-most-once return within one trusted store, not guaranteed delivery.

The cryptographic and replay models have no machine-checked composition theorem. Neither model proves SQLite crash consistency. The concrete races, acknowledgement loss, and process-termination tests in Section 6.2 provide bounded implementation evidence

instead. They also illustrate the availability cost of consuming before return: a crash can permanently deny a key that was never delivered.

The receiver rejects artifacts at signed expiry, lifetimes outside 1–300 s, and issuance more than 30 s ahead of its clock. A coherent but stale registry snapshot can nevertheless accept an old legacy artifact when external current policy requires a post-quantum mode; the stale-policy model exhibits this trace. Artifact lifetime is relative to signed issuance, subject to clock assumptions and the future-time allowance. It is not a global 300 s revocation-propagation bound and cannot prevent further issuance under persistently stale policy. Numeric clock behavior is tested concretely, not proved by the model's abstract eligibility condition. Downgrade rejection requires a current trusted policy and never triggers fallback.

7.5. Exposure History and Remaining Trust

New direct-RSA issuance records a monotonic exposure marker before wrapping or publication. The optional history gate rejects unresolved scoped inventories or conservatively marks them exposed after explicit acknowledgement. Its tests establish fail-closed handling of the supplied assertions, not their completeness or truth. The core requester neither reads the exposure store nor authenticates an exposure-status field; the selected mode label alone says nothing about prior wrappers. The CID diagnostic shows why key-wide history requires an additional identity/alias assumption. The wrapper is not a mandatory cryptographic API boundary and does not provide distributed inventory consistency.

A later post-quantum wrapper cannot remove earlier RSA ciphertexts or same-CEK dual-wrap exposure. A fresh CEK, ciphertext, commitment, and version may establish a new prospective boundary, but that complete migration workflow was not evaluated and does not erase retained older material. The analysis also does not prove erasure, side-channel resistance, source-code refinement, arbitrary storage rollback tolerance, or the inherited proof and blockchain layers.

8. Discussion

The results distinguish replacement of a cryptographic mechanism from preservation of the conditions under which a record key may be released. The contribution is an application-level relationship among authenticated context, migration policy, single-use state, and exposure history, rather than the use of ML-KEM alone.

8.1. Answers to the Research Questions

RQ1—integration. Four real adapter round trips recovered the expected keys and synthetic records across the three modes. The release mechanism can therefore change without replacing the retained record cipher or authorization interface in this prototype. The accepted event remains a trusted input; successful recovery does not validate the inherited access decision.

RQ2—rejection behavior. The executed suite, 512 generated examples, and bounded race/failure scenarios rejected the selected invalid or repeated operations at their intended checks. These evaluations include 27 reservation-coordinated races with distinct identifiers for one grant. Symbolic agreement holds for the modeled external adversary and honest recipient. Together, these support the ordering of signature, expected-context, policy, and replay checks within the stated scope, without implying exhaustive attack coverage or production fault tolerance.

RQ3—operational cost. Across five native process runs, per-call RSA loading produced a large latency gap, but validated preloading reduced legacy and hybrid median service latencies from 48.501 to 2.085 ms and from 48.824 to 2.404 ms, respectively. ML-KEM-only remained near 1.5 ms in both conditions. The comparison concerns unmatched security

levels and excludes network and durable-state costs. ML-KEM-only and hybrid artifacts were larger than legacy (7031 and 7488 versus 5670 B).

RQ4—coexistence and history. Current trusted capability policy chooses one mode without fallback, while the scoped local inventory gate distinguishes clear, exposed, and unresolved histories relative to supplied trusted inventory assertions. The 31 tested inventory scenarios demonstrate fail-closed handling of the supplied trusted assertions within the local gate. These results do not establish that institutional history is complete or that every caller uses the gate. The CID diagnostic shows that distinct CID spellings do not share exposure state automatically. Key-wide exposure status across CID aliases requires an authoritative alias/history mapping, which the gate does not establish or enforce. End-to-end migration to a new record version with a fresh encryption key was not evaluated.

8.2. Relationship to Earlier Work

The published ZK-EHR study addresses proof-based policy enforcement and encrypted record retrieval [6]; the present extension studies subsequent key delivery. Its grant-level restriction prevents repeated release for one accepted event, but a newly accepted event may identify another grant. It complements rather than repairs upstream authorization and proof-level replay controls.

Table 12 identifies the application rules that remain necessary when the KEM changes: grant binding, paired single-use state, current policy, and historical identity. As discussed in Section 2.2, HPKE with a PQ KEM needs a suitable authentication arrangement rather than assuming RFC 9180 Auth is supported [14,23]. X-Wing and LAMPS make different composition and binding choices [21,22]. The contribution is the evaluated policy and state layer rather than a requirement to use this custom hybrid. Adopting another construction would require a new profile and tests at the cryptographic interface; this study's timings and symbolic results do not transfer to it.

Healthcare-oriented authentication, signcryption, and device-security work addresses adjacent boundaries [9,10,26,33]. The present analysis isolates off-chain release after an accepted event. Since these studies differ in primitives, workloads, trust models, and experimental platforms, the local latency measurements do not support a numerical ranking against them.

8.3. Interpreting Performance and Migration

The paired experiment shows that the per-call latency gap in this service is strongly affected by provider key loading and object reuse; it is not a portable comparison of the cryptographic primitives. Most per-call RSA-bearing requester time lies inside key loading and validation, although other diagnostic operations also change. A vendor should therefore decide where validated private keys are held and how often they are admitted before using these timings to choose a release mechanism. The results do not identify parsing as the sole cause, nor do they isolate an architecture effect from the Rosetta run.

Preloading brings the local services much closer together, while leaving different security assumptions and wire costs. A deployment comparison must account for key custody, caching, key lifetime, recipients, message frequency, network delay, and durable state. The five restarted processes provide within-host replication, not independent platforms.

Coexistence is equally distinct from fallback. Legacy clients must still support the new authority signature. The hybrid requires both contributions for one wrapper; independently wrapping the same CEK under RSA and ML-KEM instead permits recovery through either route. The inventory gate preserves this distinction, but a truthful complete inventory remains an administrative prerequisite. An exposed historical version cannot become

unexposed through registry rotation or a later post-quantum artifact. A fresh key and ciphertext may protect a new version prospectively; this full workflow remains untested.

Long retention makes static-key compromise consequential: a retained artifact can be decrypted later if the recipient secrets it requires become available. Key rotation and destruction of retired private-key copies can limit part of the exposure to subsequent compromise, but only when backups and other copies are also controlled; neither measure repairs a leaked CEK or prevents later recovery from an RSA wrapper that has already been harvested. Limiting artifact retention likewise cannot remove a copy already held by an adversary. Fresh CEKs and ciphertext versions protect future record versions under new assumptions. Per-request ephemeral recipient keys are a possible design option, but they would require a different profile, authenticated key distribution, and erasure assumptions, and they are not evaluated here.

8.4. Implications for EHR System Design

For a health-technology vendor extending an RSA-based system, the results identify concrete decisions: who controls release-authority keys and registry updates; which accepted event defines a grant; how validated requester keys are retained; and whether unresolved record histories are blocked or conservatively classified as exposed. These decisions affect deployment architecture even when a standardized key-establishment mechanism is selected.

Clinical policy requires additional authoritative context. Consent, purpose of use, and custodian approval would need explicit representations and binding if they constrain release. Emergency access should be a separately authorized, time-bounded, reason-coded, audited path, not a silent retry under a weaker mode. These are integration requirements, not clinically evaluated features.

The failure tests illustrate the associated availability cost. A committed consumption followed by lost acknowledgement or process termination can deny a legitimate request without delivering its CEK. A deployment recovery procedure should let the requester raise a failed-delivery incident, let the custodian inspect the terminal grant state, and obtain a newly authorized accepted event before another release. It must not reopen the old pair or silently choose a weaker mode. A reason-coded emergency grant would likewise use a new identity, the current policy and exposure accounting, and a separate audit trail. Timeouts, repeated denials, or unresolved inventory should trigger operator escalation. These are proposed operating requirements; recovery latency, emergency access, and clinical safety have not been evaluated. Corpus-wide backfill and re-encryption also incur unmeasured storage, transfer, version-management, and key-custody costs.

9. Limitations and Future Work

9.1. Authorization and Application Scope

The evaluation uses synthetic records, an ephemeral local chain, and in-memory or isolated SQLite state. It does not evaluate clinical workflows, institutional identity, public-chain finality, or production storage. The accepted event is trusted. Requester and record/grant identity are not public inputs of the inherited Groth16 statement, and this work does not repair its equality-constraint limitation or add proof-level nullifiers. Successful key recovery therefore does not validate complete access control.

9.2. Measurement Generalizability

The five restarted processes share one Apple M4 Pro, operating system, provider configuration, and experiment period. Background activity, core placement, thread quality of service, frequency and thermal state were not controlled or continuously recorded. Fresh

process/key sets and randomized interleaving improve within-host replication but do not establish cross-host or multi-day variability. The reported paired ranges describe five observed contrasts; no calibrated population coverage is claimed.

The preload condition changes both key admission and object reuse; the diagnostic intervals do not isolate provider first-use effects or bare primitive latency. RSA-2048 and ML-KEM-768 have unmatched security levels. Network, blockchain, record transfer/decryption, concurrent load, and durable-store performance remain outside the benchmark. The exploratory Rosetta run is not a controlled architecture comparison. Key-generation/export measurements retain unexplained schedule-position sensitivity and remain descriptive, exploratory supplementary evidence; they do not inform comparative performance conclusions.

9.3. Security and Operational Scope

The symbolic analysis assumes ideal primitives, an external adversary, one honest authority/recipient pair, trusted authorization and registry inputs, and selected key disclosures rather than complete host compromise. Its results are limited to those assumptions and do not establish a generic hybrid-combiner theorem, computational IND-CCA security, forward secrecy, or security against malicious registered recipients. The separate cryptographic and two-arrival replay models lack a proved composition or source-code refinement relation. The models do not include quantum-oracle attacks, so no security claim against such attacks follows.

Implementation tests cover finite generated inputs, two-worker races including distinct release identifiers for one grant, acknowledgement loss, and selected process-termination points. The coordinated schedules do not establish power-loss durability, rollback resistance, distributed exactly-once execution, or deployment-scale concurrency. Best-effort clearing is not certified erasure, and inherited Node dependency advisories remain a deployment concern.

The historical gate is a small optional wrapper relying on trusted scoped inventory assertions and process-local admission. Literal-CID commitments do not merge equivalent locators or all uses of a CEK. The application must provide an authoritative identity/history mapping for a key-wide claim; this mapping is not enforced here. The gate also does not enforce complete institutional inventory, mandatory use by every caller, atomic updates across both stores, or distributed reconciliation. Full record re-encryption and fresh-version migration, corpus-scale backfill, authenticated profile-update distribution, emergency access, and clinical governance remain unevaluated.

9.4. Further Research

Priorities are multi-platform replicated measurements; isolated investigation of provider key-loading and first-use behavior; combined replay and failure testing under durable distributed state; and a formal link between cryptographic agreement, replay state, and implementation. A computational analysis or a standards-based replacement is needed for stronger hybrid assurance. Deployment studies should assess inventory provenance, revocation propagation, recovery after denied delivery, and the cost of creating new encrypted record versions without implying repair of previously exposed data.

10. Conclusions

This study evaluates authenticated post-authorization key release across RSA, ML-KEM-only, and a dual-contribution hybrid at an existing EHR interface. Its policy layer binds the accepted event, the expected requester and record, the selected mode, and paired single-use state, while issuer-side exposure records support an optional history gate. Symbolic agreement holds for an external adversary and one honest recipient. Implementation tests rejected the selected invalid and repeated operations, while 27 coordinated races

between distinct releases for one grant each returned exactly one matching key. The CID diagnostics show that key-wide exposure accounting needs record-key identity beyond literal commitments. Together, these results characterize authenticated release and grant-level single use within the stated trust assumptions; they do not validate the inherited authorization decision or a complete clinical deployment.

The five-process native benchmark shows why key handling matters to interpretation. Per-call total service medians of 48.501, 1.507, and 48.824 ms became 2.085, 1.495, and 2.404 ms with validated preloading for legacy, ML-KEM-only, and hybrid, respectively. These operational measurements exclude network and durable-state costs and compare unmatched security levels. The corresponding recorded artifacts occupy 5670, 7031, and 7488 B.

Post-quantum migration must preserve release policy and account for historical exposure, not merely replace an algorithm. Static recipient keys do not provide forward secrecy, and a later post-quantum release cannot restore secrecy to a CEK already exposed through RSA or another route. Prospective protection of a new record version remains a separate migration operation.

Supplementary Materials: The following supporting information can be downloaded at https://www.mdpi.com/article/10.3390/electronics15214984/s1: The Supplementary Materials report detailed measurements, symbolic queries, and implementation tests. Verification Materials (`Verification_Materials.zip`) provide concurrency and CID-diagnostic code, sanitized records, and audit ledgers.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Code and results are available at https://doi.org/10.17632/88gk36mgkk.2 and in the Supplementary Verification Materials.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

AAD	Additional authenticated data	IQR	Interquartile range
AEAD	Authenticated encryption with associated data	JCS	JSON Canonicalization Scheme
AES	Advanced Encryption Standard	JSON	JavaScript Object Notation
AES-GCM	AES Galois/Counter Mode	KDF	Key-derivation function
ASCII	American Standard Code for Information Interchange	KEM	Key-encapsulation mechanism
CEK	Content-encryption key	MGF1	Mask generation function 1
CSRC	Computer Security Resource Center	ML-DSA	Module-Lattice-Based Digital Signature Algorithm
CID	Content identifier	ML-KEM	Module-Lattice-Based Key-Encapsulation Mechanism
CISA	Cybersecurity and Infrastructure Security Agency	NIST	National Institute of Standards and Technology
IACR	International Association for Cryptologic Research	NSA	National Security Agency
DER	Distinguished Encoding Rules	OAEP	Optimal Asymmetric Encryption Padding
DICOM	Digital Imaging and Communications in Medicine	PKCS	Public-Key Cryptography Standards
IETF	Internet Engineering Task Force	RFC	Request for Comments
EHR	Electronic health record	RPC	Remote procedure call

FIPS	Federal Information Processing Standards	RQ	Research question
HKDF	HMAC-based Extract-and-Expand Key Derivation Function	RSA	Rivest–Shamir–Adleman
HMAC	Keyed-hash message authentication code	SHA	Secure Hash Algorithm
HPKE	Hybrid Public Key Encryption	SP	Special Publication
ID	Identifier	UTC	Coordinated Universal Time
IND-CCA	Indistinguishability under adaptive chosen-ciphertext attack	UTF-8	Unicode Transformation Format, 8-bit
IPFS	InterPlanetary File System	UUIDv4	Universally unique identifier, version 4
		ZK-EHR	Zero-Knowledge Electronic Health Record

Appendix A. Artifact Fields and Validation Order

This appendix defines the fields needed to interpret the authenticated key-release object specified in Section 4. CEK denotes the 32-byte content-encryption key, and AEAD denotes authenticated encryption with associated data. The research archive at https://doi.org/10.17632/88gk36mgkk.2 supplies the complete machine-readable schema, lexical constraints, stage-specific error codes, and conformance vectors.

Appendix A.1. Required Fields

Every branch requires the following common fields:

- Profile: `schema_version`, `profile_id`, `release_mode`, and `canonicalization_id`.
- Release: `release_id`, `issued_at`, and `expires_at`.
- Parties: `issuer_id`, `issuer_signing_key_id`, and `requester_address`.
- Event: `chain_id`, `contract_address`, `transaction_hash`, and `log_index`.
- Binding and signature: `access_grant_id`, `record_commitment`, `signature_id`, and `authority_signature`.

Constants are schema version 2, profile `ZKEHR-KR-v2-MLKEM768-MLDSA65`, canonicalization `JCS-RFC8785`, and signature `ML-DSA-65`. Table A1 gives the mode-specific additions. No field is optional; unknown and mixed-branch fields are rejected.

Table A1. Mode-specific artifact fields and binary lengths.

Wire Mode	Additional Required Fields	Binary Layout and Forbidden Alternative
`legacy-rsa`	`requester_rsa_key_id`, `rsa_oaep_id`, `rsa_wrapped_record_key`	One 256-byte RSA ciphertext over the CEK; all ML-KEM, KDF, AEAD, and contribution fields are forbidden.
`mlkem-only`	`requester_kem_key_id`, `kem_id`, `kdf_id`, `kdf_info`, `wrap_aead_id`, `kem_ciphertext`, `hkdf_salt`, `aead_nonce`, `wrapped_record_key`	1088-byte KEM ciphertext, 32-byte salt, 12-byte nonce, and one 48-byte `ciphertext[32]` ‖ `tag[16]` CEK wrapper; every RSA field is forbidden.
`hybrid-rsa-mlkem`	All `mlkem-only` fields plus `requester_rsa_key_id`, `rsa_oaep_id`, and `rsa_wrapped_contribution`	One 256-byte RSA ciphertext over r_R and one 48-byte AEAD CEK wrapper; `rsa_wrapped_record_key` and any second CEK wrapper are forbidden.

Table abbreviations: key-derivation function (KDF); key-encapsulation mechanism (KEM).

The context C_m contains the common fields except `authority_signature`, plus the applicable requester-key and algorithm identifiers. Legacy adds `rsa_oaep_id` and `requester_rsa_key_id`, giving 19 context fields. ML-KEM-only adds `kem_id`, `kdf_id`, `kdf_info`, `wrap_aead_id`, and `requester_kem_key_id`, giving 22; hybrid uses both addi-

tions, giving 24. Payloads are excluded from C_m but included in the signature, as defined in Section 4.3. The hybrid length prefix uint16be defined in Section 4.6 is unsigned 16-bit big-endian.

Appendix A.2. Encoding Constraints

An artifact is at most 32,768 bytes and must decode as valid UTF-8 without a byte-order mark. It contains exactly one JSON object, with no duplicate property names after escape expansion, non-finite numbers, or trailing values. Received bytes must equal their JCS serialization. Addresses and hashes use lowercase hexadecimal, identifiers use restricted ASCII, integers are JSON-safe, and timestamps represent real Gregorian UTC seconds without fractions or offsets.

Binary fields use unpadded Base64url with exact decoded lengths and byte-identical re-encoding. The decoded-byte/encoded-character pairs are 1088/1451 for the KEM ciphertext, 32/43 for the salt, 12/16 for the nonce, 48/64 for the CEK wrapper, 256/342 for either RSA ciphertext, and 3309/4412 for the signature. Canonicality does not establish cryptographic validity.

Appendix A.3. Validation Sequence

The first failing stage determines the outcome. Table A2 follows the implementation's stage numbering; the archive preserves the complete code-to-stage mapping. A single clock reading and a coherent registry snapshot are used throughout artifact validation.

Table A2. Requester validation sequence.

Stages	Required Checks or Operation
1–7	Size, strict parsing, profile and algorithm dispatch, closed schema, canonical bytes, and calendar semantics. Missing or wrongly typed dispatch fields remain schema violations.
7.25–8	Acquire clock and registry snapshots, then resolve the candidate issuer key and active window.
9	Verify the authority signature before expected-context, policy, replay, or private-key operations.
10–12	Compare independently expected context; enforce requester ownership, key status and capability mode; check lifetime, future skew, and expiry.
13–14	Atomically reserve release and grant indexes, then perform only the private-key operations required by the selected mode.
15–16	Atomically consume both reservations, then return the CEK. Uncertain consumption returns no key.

Failed RSA decryption or wrapper authentication returns `CRYPTOGRAPHIC_REJECTION`. Unexpected faults return `INTERNAL_FAILURE`. Failures never trigger fallback. The archive's detailed vectors support implementation conformance, not a formal security proof.

References

1. U.S. Department of Health and Human Services, Office for Civil Rights. *The Security Rule*; content reviewed 19 March 2026. Available online: https://www.hhs.gov/hipaa/for-professionals/security/index.html (accessed on 22 August 2026).
2. Chandak, A.; Chandak, P.; Soni, N. Blockchain Applications in Electronic Health Records: A Systematic Review of Qualitative and Quantitative Evidence. *BMC Med. Inform. Decis. Mak.* **2026**, *26*, 176. [CrossRef] [PubMed]
3. Cybersecurity and Infrastructure Security Agency; National Security Agency; National Institute of Standards and Technology. Quantum-Readiness: Migration to Post-Quantum Cryptography. *Joint Fact Sheet*, 17 August 2023. Available online: https://www.nccoe.nist.gov/sites/default/files/2023-08/quantum-readiness-fact-sheet.pdf (accessed on 19 October 2026).
4. Banerjee, A.; Reddy, K.T.; Schoinianakis, D.; Hollebeek, T.; Ounsworth, M. *Post-Quantum Cryptography for Engineers*; RFC 9958; Internet Engineering Task Force: Fremont, CA, USA, 2026 . [CrossRef]
5. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM J. Comput.* **1997**, *26*, 1484–1509. [CrossRef]
6. Albalwy, F. Zero-Knowledge-Based Policy Enforcement for Privacy-Preserving Cross-Institutional Health Data Sharing on Blockchain. *Systems* **2026**, *14*, 385. [CrossRef]
7. Kaliski, B.; Jonsson, J.; Rusch, A. *PKCS #1: RSA Cryptography Specifications Version 2.2*; Moriarty, K., Ed.; RFC 8017; Internet Engineering Task Force: Fremont, CA, USA, 2016. [CrossRef]

8. Groth, J. On the Size of Pairing-Based Non-Interactive Arguments. In *Advances in Cryptology—EUROCRYPT 2016*; Fischlin, M., Coron, J.S., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9666, pp. 305–326. [CrossRef] [PubMed]
9. Tan, H.; Gu, S.; Zhu, H.; Shen, J.; Bhuiyan, Z.A.; Wu, Q.M.J. An Efficient ASCON-Based Group Authentication and Key Agreement Scheme with Non-Linkability and Integrity Assurance for IIoTs. *IEEE Trans. Dependable Secur. Comput.* **2026**, *23*, 8032–8047. [CrossRef]
10. Sourav; Ali, R.; Obaidat, M.S. Lattice-based Certificateless Signcryption Scheme using Blockchain for Smart Healthcare Systems. *IEEE Trans. Emerg. Top. Comput.* **2026**, *early access*. [CrossRef]
11. National Institute of Standards and Technology. *Module-Lattice-Based Key-Encapsulation Mechanism Standard*; Federal Information Processing Standards Publication 203; U.S. Department of Commerce: Gaithersburg, MD, USA, 2024. [CrossRef]
12. National Institute of Standards and Technology. *Module-Lattice-Based Digital Signature Standard*; Federal Information Processing Standards Publication 204; U.S. Department of Commerce: Gaithersburg, MD, USA, 2024. [CrossRef]
13. Alagic, G.; Barker, E.; Chen, L.; Moody, D.; Robinson, A.; Silberg, H.; Waller, N. *Recommendations for Key-Encapsulation Mechanisms*; NIST Special Publication 800-227; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2025. [CrossRef]
14. Barnes, R.; Bhargavan, K.; Lipp, B.; Wood, C. *Hybrid Public Key Encryption*; RFC 9180; Internet Research Task Force: Fremont, CA, USA, 2022. [CrossRef]
15. Rundgren, A.; Jordan, B.; Erdtman, S. *JSON Canonicalization Scheme (JCS)*; RFC 8785; RFC Editor: Fremont, CA, USA, 2020. [CrossRef]
16. Barker, E.; Chen, L.; Cooper, D.; Moody, D.; Regenscheid, A.; Souppaya, M.; Newhouse, B.; Housley, R.; Turner, S.; Barker, W.; et al. *Considerations for Achieving Crypto Agility: Strategies and Practices*; NIST Cybersecurity White Paper 39 Update 1; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2025; includes updates as of 29 June 2026 . [CrossRef]
17. Moody, D.; Perlner, R.; Regenscheid, A.; Robinson, A.; Cooper, D. *Transition to Post-Quantum Cryptography Standards*; NIST Internal Report 8547 Initial Public Draft; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2024. [CrossRef]
18. Krawczyk, H.; Eronen, P. *HMAC-Based Extract-and-Expand Key Derivation Function (HKDF)*; RFC 5869; Internet Engineering Task Force: Fremont, CA, USA, 2010. [CrossRef]
19. Giacon, F.; Heuer, F.; Poettering, B. KEM Combiners. In *Public-Key Cryptography—PKC 2018*; Abdalla, M., Dahab, R., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 10769, pp. 190–218. [CrossRef]
20. Bindel, N.; Brendel, J.; Fischlin, M.; Goncalves, B.; Stebila, D. Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange. In *Post-Quantum Cryptography—PQCrypto 2019*; Ding, J., Steinwandt, R., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2019; Volume 11505, pp. 206–226. [CrossRef]
21. Barbosa, M.; Connolly, D.; Duarte, J.D.; Kaiser, A.; Schwabe, P.; Varner, K.; Westerbaan, B. X-Wing: The Hybrid KEM You've Been Looking For. *IACR Commun. Cryptol.* **2024**, *1*, 1–22. [CrossRef]
22. Ounsworth, M.; Gray, J.; Pala, M.; Klaußner, J.; Fluhrer, S. *Composite ML-KEM for Use in X.509 Public Key Infrastructure*; Internet-Draft draft-ietf-lamps-pq-composite-kem-19; Internet Engineering Task Force: Fremont, CA, USA, 2026; work in progress. Available online: https://www.ietf.org/archive/id/draft-ietf-lamps-pq-composite-kem-19.html (accessed on 19 October 2026).
23. Barnes, R.; Connolly, D. *Post-Quantum and Post-Quantum/Traditional Hybrid Algorithms for HPKE*; Internet-Draft draft-ietf-hpke-pq-05; Internet Engineering Task Force: Fremont, CA, USA, 2026; work in progress. Available online: https://www.ietf.org/archive/id/draft-ietf-hpke-pq-05.html (accessed on 19 October 2026).
24. Pathum, U.; De Silva, A. Post-Quantum Migration Strategy for RSA Encryption. *Cryptology ePrint Archive* 2026, Paper 2026/1172, preprint. Available online: https://eprint.iacr.org/2026/1172 (accessed on 19 October 2026).
25. Yadav, B.; Arakeri, M. Secure Near-Lossless Medical Image Compression and Encryption with Image-Specific Post-Quantum Key Management and Blockchain-Based Verification. *PLoS ONE* **2026**, *21*, e0348013. [CrossRef] [PubMed]
26. Latif, R.; Yakubu, B.M.; Jamail, N.S.M.; Talib, A.M.; Alomary, F.O. BBAS: A Blockchain-Based Authentication System for E-Health with Multi-Factor Authentication, Access Control, and Post-Quantum Security. *Sci. Rep.* **2026**, *16*, 9163. [CrossRef] [PubMed]
27. Segar, N.; Vijayan, V. CITADEL: A Post-Quantum Secure Blockchain Framework for Privacy-Preserving Electronic Health Records with Temporally-Partitioned Federated Learning. *Front. Artif. Intell.* **2026**, *9*, 1804943. [CrossRef] [PubMed]
28. Olushola, A.; Meenakshi, S.P. Design and Implementation of an Authenticated Post-Quantum Session Protocol Using ML-KEM (Kyber), ML-DSA (Dilithium), and AES-256-GCM. *Front. Phys.* **2026**, *13*, 1723966. [CrossRef]
29. Hao, L.; Wang, R.; Wang, X.; Yue, X.; Tariq, N.; Sajid, A. Post-Quantum-Inspired Scalable Blockchain Architecture for Internet Hospital Systems with Lightweight Privacy-Preserving Access Control. *PLoS ONE* **2025**, *20*, e0332887. [CrossRef] [PubMed]
30. Driscoll, F.; Parsons, M.; Hale, B. *Terminology for Post-Quantum Traditional Hybrid Schemes*; RFC 9794; Internet Engineering Task Force: Fremont, CA, USA, 2025. [CrossRef]
31. Stebila, D.; Fluhrer, S.; Gueron, S. *Hybrid Key Exchange in TLS 1.3*; RFC 9954; Internet Engineering Task Force: Fremont, CA, USA, 2026. [CrossRef]

32. Tjhai, C.J.; Tomlinson, M.; Bartlett, G.; Fluhrer, S.; Van Geest, D.; Garcia-Morchon, O.; Smyslov, V. *Multiple Key Exchanges in the Internet Key Exchange Protocol Version 2 (IKEv2)*; RFC 9370; Internet Engineering Task Force: Fremont, CA, USA, 2023. [CrossRef]
33. Khan, U.H.; Khan, R.; Chelloug, S.A.; Alturise, F.; Khan, S.; Alkhalaf, S. HybridTrust: On-Device Federated Learning with Crypto-Agile Security for Legacy and Quantum-Safe Medical Devices. *Sci. Rep.* **2026**, *16*, 19539. [CrossRef] [PubMed]
34. Barker, E. *Recommendation for Key Management: Part 1—General*; NIST Special Publication 800-57 Part 1 Revision 5; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2020. [CrossRef]
35. Blanchet, B. Modeling and Verifying Security Protocols with the Applied Pi Calculus and ProVerif. *Found. Trends Priv. Secur.* **2016**, *1*, 1–135. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.