Threshold-Authorized, Post-Event Selective Disclosure with Content Zero-Knowledge against the Storage Provider and Post-Quantum Confidentiality

John Sabatico · Semparo · isabatico@gmail.com

Revision 6, 2026-08-01

Published for open review · Revision 6 · 2026-08-01 · Semparo. **Patent pending.** The mechanisms are the subject of U.S. patent applications filed by Semparo. This paper is published for scrutiny, not peer-reviewed. The disposition of a prior independent review is in Appendix A. Send analysis to `contacts@semparo.com` (subject: paper review).

Working draft for external review, ***revision 6****. Revision 2 incorporated an independent peer review (see Appendix A for the disposition). Revision 3 (2026-07-05) corrects the primitive descriptions to match the implementation (AEAD-with-AAD key wraps rather than AES-KW, hybrid Ed25519 + ML-DSA-65 signatures, the as-built key hierarchy) and clarifies the anti-rollback implementation status (§3.5). Revision 4 (2026-07-09) adds an at-rest PII refinement to §6 (Appendix C): operational contact/identity PII is now application-layer encrypted at rest, closing the passive storage-exfiltration path without any new zero-knowledge claim. Revision 5 (2026-07-17) corrects several descriptions to match the construction (Appendix D): the session-channel standards label (an HPKE-style hybrid KEM-AEAD, not RFC 9180 HPKE), how E obtains the authentic owner key (an RWK-sealed commitment opened after threshold reconstruction, not a setup-time anchor), the owner veto (an authenticated control action enforced by the release state machine, not a signature verified in E), the per-Consumer wrap lifetime (eager in the reconstruction burst with* RWK *zeroized, not lazy), and the scope of the signed audience (entitlement edges and Consumer identifiers, not mutable delivery contacts). Revision 6 (2026-08-01) is a presentation revision: the paper is retypeset, the figures are corrected (Figure 1 now uses the paper's K_V notation; Figure 2 adds the signature layer it had omitted), and the prose is lightened. No technical content changed in revision 6. Abstract roles are used throughout; the construction is presented as a generic cryptographic scheme mediated by an attested TEE E, the production trust boundary the system is built on. The one enforcement stated as a pre-production hardening rather than as built is the TEE-sealed anti-rollback version register (§3.5), unchanged from revision 3.*

> **Scope of the "zero-knowledge" claim.** We mean *content* zero-knowledge: the Storage Provider learns no item *plaintext* or content key. Routing *metadata* (which Consumer is addressed by which item, and Consumer contacts) is deliberately visible to the provider (§2). We do not claim to hide the relation's existence.

Plain-language summary (read this first)

This paper describes a way to lock up digital information so that:

- Only specific people you choose can ever see it, and each piece of information can be addressed to a *different* set of people.
- Nothing is revealed until a particular event happens (we keep the event abstract here) and a small group of trusted helpers agrees to unlock it. No single party, not even the company that stores the data, can open it early or alone.
- The company storing the data can never read it. It only ever holds scrambled bytes. Even if its servers are completely compromised, the contents stay secret.
- It stays secret even against future quantum computers. This matters because the data may sit locked away for years before it is ever opened, and an attacker can record the scrambled bytes today and try to crack them later.

The cast. An **Owner** seals the information. Three **Agents** are the trusted helpers. Any two of the three can authorize an unlock, so losing one does not lock things up forever, and no single Agent can act alone. The chosen recipients are the **Consumers**. A tamper-proof **secure chip (a "TEE")** does the actual unlocking inside a sealed environment that even the company's own operators cannot look inside.

How it works, in three steps:

1. **Sealing.** The Owner scrambles each item with its own key, then locks those keys so only the Owner (now) or the group of Agents (later) can recover them. The master unlock key is split into three pieces, one per Agent, and each Agent's piece is *itself* split in two: half kept by the company, half kept offline by the Agent. So neither the company nor any Agent can act alone.
2. **Unlocking.** When the event occurs and enough Agents agree, the secure chip reassembles the master key, but only after a waiting period during which the Owner can cancel. That window is what stops a false alarm, or a dishonest group, from forcing an unlock. The chip then prepares each recipient's personal keys.
3. **Receiving.** Each recipient proves who they are with one-time codes that are checked *inside* the secure chip, and the chip hands them their keys over a quantum-safe channel. They decrypt and read only what was meant for them, on their own device, never on the company's servers.

The weak points. Recipients carry no secret in advance, so their *first* sign-in leans on codes the company delivers. That is strong against an honest company and weaker against a fully malicious one, and we explain below how to harden it. The whole design also trusts that the secure chip is genuine, which each recipient's device must verify.

That is the whole idea. The rest of the paper makes every step precise and argues why it is secure.

Abstract

We describe a scheme by which a single *Owner* seals a set of data items, each with an individual *audience* of designated *Consumers*, such that: (i) no item is disclosed until an externally-attested *Trigger Event* occurs *and* a threshold t of n *Agents* authorizes it; (ii) an untrusted *Storage Provider* that holds all ciphertext learns nothing about item *content*; (iii) after authorization, each Consumer

recovers exactly the items addressed to them, and nothing else; and (iv) all data held at rest is post-quantum confidential. Disclosure is mediated by an attested *Trusted Execution Environment* (TEE) that prepares per-Consumer keys in a single reconstruction burst and delivers each Consumer's keys lazily, on first contact. We give the construction, a threat model with four security goals, a security argument for each, a post-quantum analysis layer by layer, and a statement of the scheme's limitations. We solicit review of the threshold/two-factor reconstruction argument, the signed-audience anti-rollback design, the hybrid-KEM session channel, and the honest-but-weaker Consumer authentication tier.

1. Roles and notation

Symbol	Role
O	**Owner.** Seals items, defines audiences, holds the long-term secrets.
$A_1, \ldots, A_n$	**Agents.** Threshold authorizers; here $n = 3$, threshold $t = 2$.
$C_1, \ldots, C_m$	**Consumers.** Designated recipients of specific items.
$\mathcal{T}$	**Trigger Event.** An external condition, attested out of band, that *enables* (never alone *causes*) disclosure.
S	**Storage Provider.** Untrusted host of all ciphertext and metadata.
E	**TEE.** An attested trusted execution environment with an attestation-gated key-management service (KMS).
δ	A high-entropy *setup secret* delivered to O out of band (e.g. printed).

Primitives: `Argon2id` (memory-hard KDF); `HKDF` (RFC 5869); `AEAD` = AES-256-GCM; `KW` = an *AEAD key wrap*, $\mathrm{KW}(k, m; ad) = \mathrm{AEAD.Enc}(k, m; ad)$ with a fresh random nonce and the owning identifiers bound as associated data (*not* RFC 5649 AES-KW; see §8 Q6); `SSS` = Shamir secret sharing over $\mathrm{GF}(2^8)$; `Sig` = *hybrid* Ed25519 (RFC 8032) + ML-DSA-65 (FIPS 204), where a signature carries both components and verification requires both. The session channel is an *HPKE-style hybrid KEM-AEAD*: a hybrid KEM (X25519 + ML-KEM-768, FIPS 203) whose two shared secrets are combined and fed through `HKDF` to an AEAD key, with the addressed identifiers/nonce bound as associated data. It is *not* RFC 9180 HPKE (no HPKE key schedule, suite identifiers, or labeled-KDF framing); we borrow only HPKE's seal-to-ephemeral-public-key shape (§3.6). `HOTP` (RFC 4226, SHA-256). $x \xleftarrow{\$} \{0,1\}^{256}$ denotes a fresh uniform 256-bit value. $\mathrm{AEAD.Enc}(k, m; ad)$ binds associated data ad.

2. Threat model and security goals

Adversaries. We consider (a) a *passive* S (honest-but-curious; sees all stored bytes and traffic); (b) an *active/malicious* S (may tamper with stored data and forge messages, but cannot break the TEE attestation); (c) up to $t - 1$ *colluding Agents*; (d) a *harvest-now-decrypt-later* adversary who records ciphertext today and possesses a cryptographically-relevant quantum computer (CRQC) in the future; (e) a *network* adversary; and (f) an adversary that *forges the Trigger Event* (a colluding S, up to $t - 1$ Agents, and the Event Notary of §3.5). The TEE's hardware root of trust and the soundness of remote attestation are assumed (discussed as a limitation in §6). We also assume an *Owner-liveness/veto channel* during the waiting window (§3.5): if the Trigger assertion is false, the

Owner is available to object. This is the semantic meaning of the event and is the non-cryptographic anchor of (G1).

Security goals.

- **(G1) No false disclosure.** For any PPT adversary controlling S and at most $t-1$ Agents, *before* an attested $\mathcal{T}$ together with $\geq t$ Agent authorizations, the adversary's advantage in distinguishing any m_i from random is negligible.
- **(G2) Storage-provider content zero-knowledge.** The entire view of S (passive or active) reveals nothing about any item's plaintext or any content key; S sees only ciphertext, wrapped keys, and routing metadata.
- **(G3) Metadata integrity.** A malicious S cannot cause an item to be disclosed to a Consumer the Owner did not authorize (no insertion, alteration, or rollback of the audience).
- **(G4) Post-quantum confidentiality at rest.** No long-lived secret is protected solely by a primitive a CRQC can break; recorded ciphertext stays confidential against (d).

Routing *metadata* (the audience graph and Consumer contacts) is deliberately in scope for S: a conscious "secure at the right limit" choice. (G2) constrains *content*, not the existence of the who-receives-what relation. §6 states the residual privacy cost.

3. Construction

3.1 Owner key hierarchy

From the passphrase P and setup secret δ:

$$K_{\text{pw}} = \text{Argon2id}(P,\ \mathit{salt}); \qquad \text{MK} = \text{HKDF}(K_{\text{pw}} \,\|\, \delta;\ \texttt{"MK"})$$

The master key MK wraps a randomly generated *vault key* $K_V \xleftarrow{\$} \{0,1\}^{256}$, the Owner's content-root key. (K_V is wrapped rather than derived so that it survives a credential reset: a new MK simply re-wraps the same K_V.) Two further secrets hang off K_V:

$$\text{RWK} \xleftarrow{\$} \{0,1\}^{256} \quad (\text{held by } O,\ \text{sealed under } K_V)$$
$$(sk_O, pk_O) = \text{Sig.KeyGen}(\text{HKDF-Expand}(K_V;\ \texttt{"sign"}))$$

K_V lets O read items while active. RWK (the *Release Wrapping Key*) is the secret the Agents will threshold-hold. It is deliberately an independent uniform key: deriving it from MK or K_V would entangle the release path with the Owner's credential lifecycle for no gain, and the Agents must be able to reconstruct it at disclosure time without any Owner secret. The hybrid signing keypair is derived from K_V by HKDF-Expand under a dedicated info label, domain-separated from every AEAD use of K_V. pk_O is bound to the release via an *RWK-sealed commitment*, not provisioned into E at setup. At designation time O seals pk_O under a key derived from RWK (HKDF-Expand of RWK under a dedicated label, with the account identifier as associated data) and stores the sealed commitment on S. E recovers the authentic pk_O only *after* it has reconstructed RWK from a threshold of Agents (§3.5), and then verifies σ under it. Opening the commitment requires RWK, which no party below threshold holds, so a malicious S cannot substitute an owner key it controls: forging the audience signature would require first reconstructing RWK, which is the same

threshold gate as disclosure itself. This ties owner-key authenticity to the threshold instead of to E's provisioning, a stronger property than a setup-time provisioned anchor. A separate printed recovery code ρ wraps K_V to allow credential reset without weakening the above (recovery path elided here).

3.2 Per-item sealing (Owner, while unlocked)

For each item m_i with identifier id_i:

$$\mathrm{CEK}_i \xleftarrow{\$} \{0,1\}^{256}, \qquad c_i = \mathrm{AEAD.Enc}(\mathrm{CEK}_i,\ m_i;\ ad = id_i)$$
$$w_i^O = \mathrm{KW}(K_V,\ \mathrm{CEK}_i;\ ad = id_i), \qquad w_i^R = \mathrm{KW}(\mathrm{RWK},\ \mathrm{CEK}_i;\ ad = id_i)$$

O uploads $(id_i,\ c_i,\ w_i^O,\ w_i^R)$. Content is encrypted exactly once under CEK_i; the two wraps are constant-size ($\approx$ 60-byte) AEAD key-wraps, each binding id_i as associated data so a wrap cannot be replayed against a different item. w_i^O lets O re-read; w_i^R lets E recover CEK_i after the event. RWK is recoverable by O at any time (it is held sealed under K_V as an opaque blob on S), so new items can be sealed for release without RWK ever appearing to S in the clear.

3.3 Audience and authenticated metadata

Let the audience graph be $G = \{(id_i,\ \mathrm{aud}(i))\}$ where $\mathrm{aud}(i) \subseteq \{C_1, \ldots, C_m\}$, alongside a Consumer registry $\mathcal{R}$ (Consumer id + delivery contacts). O signs a canonical serialization with a *monotonic version* v (anti-rollback):

$$\sigma = \mathrm{Sig.Sign}(sk_O,\ \mathrm{encode}(G \,\|\, \{\text{Consumer ids}\} \,\|\, v))$$

S stores $(G, \mathcal{R}, v, \sigma)$. What is signed is the audience *entitlement*: the item-to-Consumer-id edges and the Consumer *identifiers*, together with v. The mutable *delivery contacts* in $\mathcal{R}$ (a Consumer's current email or phone) are deliberately outside σ, because they change over the vault's lifetime and are routing data S already sees (§2). A malicious S therefore cannot add, drop, or re-point an entitlement edge without breaking σ (the property (G3) needs), though it *can* alter the contact address to which a legitimately-entitled Consumer's codes are sent. That is the first-contact weakness §6.1 already discloses (an S that controls a Consumer's delivery channel can complete that Consumer's protocol). Binding contacts under σ would not close that gap (the Consumer still holds no pre-shared secret) and is left to the §6.1 durable-credential hardening. The default audiences are policy, not crypto (some item classes default to $\emptyset$, others to "all Consumers"); only the *signed* graph is authoritative.

3.4 Threshold split of the RWK (Agent designation)

O Shamir-splits RWK with threshold $t = 2$ over $n = 3$: $(s_1, s_2, s_3) = \mathrm{SSS.Split}_{2,3}(\mathrm{RWK})$. Each share is then *two-factored* into a *server-half* and an *offline fragment*:

$$s_k = h_k \oplus f_k, \qquad h_k \to S, \quad f_k \to A_k \text{ (out of band, offline)}$$

Reconstructing share s_k requires both h_k and f_k. S holds $\{h_k\}$; each Agent holds only its f_k. Each f_k is a $\geq$ 128-bit uniform value, so the $\{h_k\}$ alone are useless.

3.5 Trigger Event, Owner veto window, and reconstruction (inside E)

Attesting $\mathcal{T}$. The event is asserted by a signed statement τ from a designated *Event Notary* N (a party or quorum whose public key pk_N is bound at setup) and/or from the Agent quorum itself; E verifies τ under pk_N. A signed assertion is not treated as sufficient, because it could in principle be forged by a colluding S, $t-1$ Agents, and N (threat (f)). *Instantiation.* In this system N is realized as the Agent quorum itself: the trigger is a lapse of the Owner's liveness signal followed by identity-verified confirmations from the Agents, and presenting $\geq t$ fragments (below) *is* the quorum's cryptographic assertion. A separate standalone notary key pk_N is a supported generalization of the same interface.

Owner veto window. Reconstruction is therefore gated behind a mandatory waiting period W after a valid τ. By the semantics of $\mathcal{T}$, the Owner is available to object *iff the assertion is false*. During W the Owner aborts through an *authenticated control action*: an owner-session (or delegated-staff) request, or a single-use unguessable abort token delivered out of band to the Owner (stored only as its hash). The release state machine honors it by transitioning the case to a terminal *aborted* state and resetting. The abort is accepted at any point up to the instant reconstruction begins, under a row lock, so a veto and a reconstruction cannot interleave; disclosure proceeds only if W elapses with no abort. Hence (G1) is enforced by threshold secrecy *and* a liveness/veto window: a false $\mathcal{T}$ is caught by the live Owner, and a genuine $\mathcal{T}$ goes un-aborted. Purely-cryptographic no-false-disclosure is impossible when the triggering condition is *external* to the cryptosystem, so we state the hybrid guarantee as such. *Note on the veto's form.* The veto is an authenticated control action enforced by the release state machine, not a signature $\mathrm{Sig.Sign}(sk_O, \texttt{"abort"} \| nonce)$ verified inside E: E recovers pk_O only after reconstructing RWK (§3.1), so it cannot check an Owner signature during the pre-reconstruction window in which the veto must act. The (G1) guarantee, that the Owner stops a false trigger during W and an abort wins over any not-yet-begun reconstruction, rests on the state machine's row-locked ordering of abort-versus-reconstruct. A cryptographically-signed veto is a possible hardening that would remove the orchestration layer from the abort's trust path as well.

Reconstruction. On a valid τ, $\geq t$ Agents presenting their f_k, and an elapsed un-vetoed W, E:

1. forms $s_k = h_k \oplus f_k$ for each of the $\geq t$ participating Agents and reconstructs $\mathrm{RWK} = \mathrm{SSS.Recover}(\{s_k\})$;
2. verifies σ under the bound pk_O and that v is fresh, rejecting any G with a stale or rolled-back v (defeating an S that restores an old signed G to re-grant a revoked Consumer). Freshness is enforced structurally: the store refuses any state write whose v does not strictly increase, and an append-only, tamper-evident audit chain anchors the latest accepted v, so a G rolled back to an older signed state is detected and refused at disclosure time. This makes rollback tamper-*evident* but leaves the anchor outside the TEE. The hardened target, a stated *pre-production requirement, not yet deployed*, pins v in a *TEE-sealed monotonic version register* $v^\star$ held in sealed enclave state that S cannot revert (e.g. a hardware monotonic counter or a sealed monotonic object bound to the enclave identity), with E advancing $v^\star$ on each accepted update, removing the provider from the freshness path entirely (see also §8 Q2);
3. unwraps RWK only through the attestation-gated KMS, whose unwrap policy requires both the measured-image attestation and an authorization token witnessing (valid $\tau \wedge \geq t$ Agents $\wedge$ elapsed W), so attestation alone never suffices, then zeroizes the plaintext RWK after the wrapping burst.

3.6 Per-Consumer wrap (eager, at authorization) + lazy session delivery

Per-Consumer key material is prepared in the single reconstruction burst (§3.5): having reconstructed RWK, E unwraps each authorized CEK_i, and for every Consumer C_j in the audience mints a fresh PCK_j and re-wraps that Consumer's CEK_i's under it, then zeroizes RWK before the burst returns. Only the *session delivery* of PCK_j to C_j is deferred to C_j's first contact. (This is a change from the earlier "lazy per-Consumer wrap" framing: the wraps are eager, but RWK is used once and destroyed instead of kept reachable across the disclosure window; see §6.2. The per-Consumer PCK indirection is what makes this possible: the durable wraps commit to a PCK, and only that PCK is later sealed to the Consumer's first-contact session key, so no future session key is needed at prep time.) For each C_j:

$$\text{PCK}_j \xleftarrow{\$} \{0,1\}^{256}; \qquad \forall i : C_j \in \text{aud}(i) :$$
$$\text{CEK}_i = \text{KW}^{-1}(\text{RWK},\, w_i^R;\ ad = id_i), \qquad wp_{i,j} = \text{KW}(\text{PCK}_j,\, \text{CEK}_i;\ ad = (id_i, C_j))$$

E stores $\{wp_{i,j}\}$ and the KMS-wrapped PCK_j. Entitlement is checked against the verified G (§3.3): E will not wrap CEK_i for C_j unless $(id_i, C_j) \in G$.

Session. C_j authenticates with a *dual one-time code* (HOTP secret + counter held *inside* E; the two codes are delivered over two independent channels by S and validated inside E with a constant-time, both-codes-required comparison, so a compromised host cannot forge a "valid" verdict). On success C_j's client sends a fresh hybrid ephemeral public key $(epk^x, epk^{\text{mlkem}})$; E seals PCK_j to it with the HPKE-style hybrid KEM-AEAD of §1: combiner $ss = \text{HKDF}(ss_{x25519} \,\|\, ss_{\text{mlkem}})$ to an AES-256-GCM key, AAD = a per-session nonce (again, this is *not* RFC 9180 HPKE; see §1). The client opens PCK_j in memory, unwraps the relevant $wp_{i,j} \to \text{CEK}_i$, and AEAD-decrypts c_i *client-side*. Everything is discarded at session end.

Figure 1 shows the three stages end to end.

4. Security argument (sketch)

(G1) No false disclosure. All item content is AEAD-sealed under per-item CEK_i, available only via K_V (O-only) or RWK. RWK is $\text{SSS}_{2,3}$-shared, so any $\le t-1 = 1$ shares are information-theoretically independent of RWK. Each share further requires $h_k \oplus f_k$; an adversary holding all $\{h_k\}$ (full S compromise) but $< t$ fragments learns nothing. So below threshold no CEK_i, and no m_i, is recoverable. *Above* threshold, recovery is further gated by E on a valid τ and an elapsed un-vetoed window W (§3.5): the KMS unwrap policy releases RWK to E only on attestation $\wedge\, \tau \wedge \ge t$ Agents $\wedge\, W$. A *forged* $\mathcal{T}$ (threat (f)) is defeated by the Owner's veto during W rather than by cryptography; the trigger condition lives outside the cryptosystem, so the guarantee takes this hybrid form. ■ (sketch)

(G2) Content zero-knowledge vs. S. S's view is

$$\{c_i,\ w_i^O,\ w_i^R,\ G,\ \mathcal{R},\ v,\ \sigma,\ \{h_k\},\ wp_{i,j},\ \text{KMS-wrapped PCK}_j\}.$$

Every key appears only AEAD-wrapped under a key S lacks, or Shamir-hidden; content appears only AEAD-sealed. Plaintext keys exist only inside E, released only under attestation-gated KMS. Thus even a malicious S reduces to breaking the AEAD or the TEE attestation. ■ (sketch)

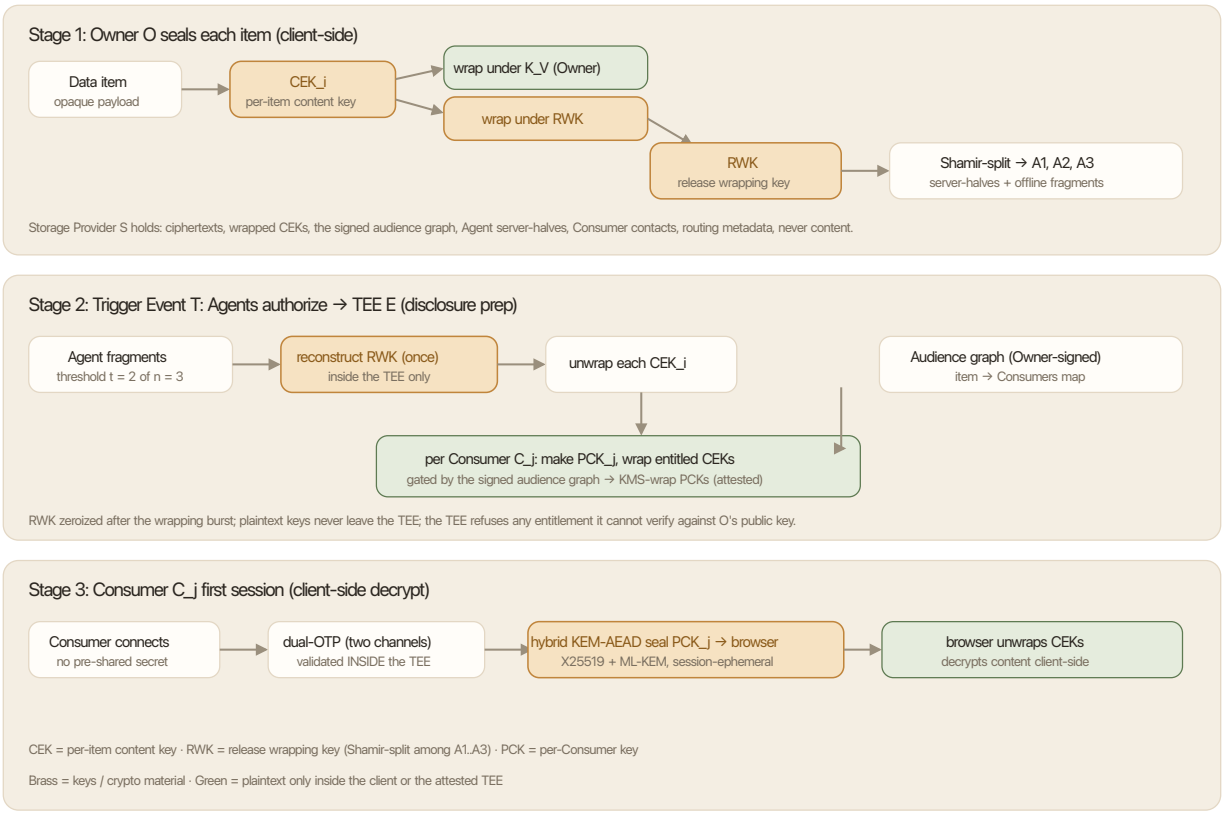

Figure 1: The three-stage key lifecycle: sealing, threshold-authorized reconstruction, first-contact delivery.

(G3) Metadata integrity. E acts only on a G whose signature verifies under the bound pk_O at the pinned latest version v. A malicious S that inserts a Consumer, re-points an item, or replays an older G produces a bad signature or a stale v and is rejected. (Without §3.3, S could silently add a "shadow" Consumer after $\mathcal{T}$, which is why owner-signing is mandatory.) ■ (sketch)

(G4) See §5.

5. Post-quantum analysis

A scheme whose ciphertext may persist for years must resist harvest-now-decrypt-later. We classify every layer (Figure 2):

Layer	Mechanism	PQ status
Item content	AES-256-GCM	symmetric, PQ
CEK wrap (to K_V, RWK, PCK)	AES-256-GCM (AAD-bound)	symmetric, PQ
RWK/PCK at rest	attested-KMS (AES)	symmetric, PQ
Agent threshold split	Shamir over GF(2^8)	information-theoretic, PQ
Audience/state signature	hybrid Ed25519 + ML-DSA-65	hybrid, PQ

Layer	Mechanism	PQ status
Per-Consumer asymmetric delivery	hybrid X25519 + ML-KEM-768	hybrid, PQ
Session seal (PCK → client)	HPKE-style hybrid KEM-AEAD (X25519 + ML-KEM-768)	hybrid, PQ

Symmetric layers retain $\geq$ 128-bit security under Grover. The two asymmetric channels are hybrid: the shared secret combines a classical (X25519) and a lattice (ML-KEM) encapsulation, so confidentiality holds if *either* survives. The audience/state signature is likewise hybrid (Ed25519 + ML-DSA-65; both components must verify), so the integrity of the durable audience graph does not rest on a classical assumption alone either. No layer protecting long-lived data relies on a classical assumption alone, including the live session seal (a recorded sealed-PCK blob transits S and is therefore harvestable, so we hybridize it rather than rely on its ephemerality).

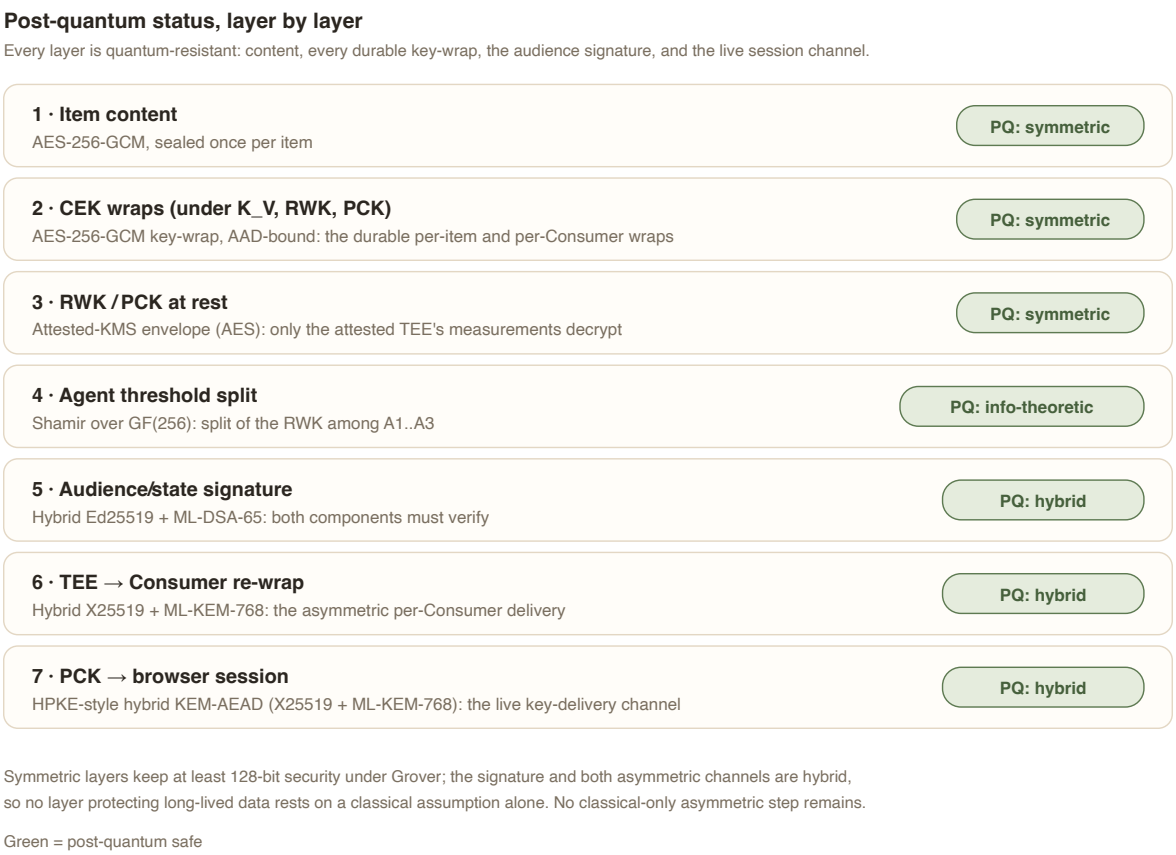

Figure 2: Post-quantum status by layer: every layer is symmetric, information-theoretic, or hybrid.

6. Limitations

1. **Consumer tier is weaker than Owner tier.** Consumers hold no pre-shared secret; S delivers their one-time codes, so an adversary controlling both delivery channels, or a malicious S itself, can complete a Consumer's protocol and obtain PCK_j. No cryptography fixes

a "nothing pre-shared plus provider-delivered code" design. Mitigation (optional, opt-in): enroll a durable client credential (e.g. WebAuthn) at first successful contact so *return* sessions are strong. Consequently initial Consumer authentication assumes an honest-but-curious S; resisting a *malicious* S on the *first* session requires either a code channel independent of S, or a durable Consumer credential enrolled before any item is sealed. We state this rather than claim content zero-knowledge for Consumers.

2. **The TEE is a trust anchor.** If E's memory is exfiltrated while RWK or a PCK is resident in plaintext, confidentiality fails. Mitigations: attestation-gated KMS (a stolen DB plus stolen fragments cannot reconstruct outside the measured image), a memory-safe implementation, a bounded disclosure window, and zeroize-after-burst. In the construction (§3.6), RWK is reconstructed, used to produce all per-Consumer wraps, and zeroized within the single reconstruction burst; it is *not* kept reachable for the disclosure window's duration. The per-Consumer PCK indirection (durable PCK-wraps, lazy session seal of the PCK) is what removes the need to retain RWK or to seal eagerly to a *future* session key. What does persist for the window is each per-Consumer PCK (KMS-wrapped at rest), a much narrower exposure than a live RWK.
3. **Attestation verification is the decisive human factor.** The Consumer's client must verify E's attestation to the hardware root and *pinned* measurements; a user who accepts a forged attestation can be impersonated. Pre-published measurements and client-side pinning are required.
4. **Availability.** If E is unreachable, no disclosure occurs (S cannot substitute, since it lacks the KMS key). Mitigated by redundant TEEs sharing identical attested measurements and KMS policy.
5. **Metadata.** S learns the audience graph and contacts (§2). Owner-signing makes it tamper-*evident* but not hidden; the relation's existence is exposed by design. **Refinement (contact PII at rest):** operational PII (Owner and Consumer email, phone, legal name, and date of birth) is now *encrypted at rest* at the application layer (randomized AES-256-GCM, AAD-bound per row; equality lookups via a keyed-HMAC blind index), under a data key and index key that are KMS-wrapped and never stored in the database. A *passive* compromise of the storage layer (a DB dump, snapshot, backup, SQL-injection read, or leaked DB credential) therefore yields ciphertext, not contacts. (A marketing/waitlist email list, which holds no vault data and is not part of any Owner or Consumer record, was initially exempt from this refinement; as of 2026-07-18 it is encrypted under the same envelope and blind-index pattern, so the statement above now covers every stored contact address.) This does *not* strengthen the metadata goal against an *active* S: the operating service holds the wrapping key and must decrypt contacts to route notices, so a fully-compromised operator (app-host RCE plus KMS access) can still read them. It is defense-in-depth against at-rest exfiltration, one tier weaker than the content zero-knowledge of (G2), which is unaffected. This is the standard blind-index / envelope pattern, and it makes no zero-knowledge claim over PII.

7. Efficiency

Content storage is $O(\text{items})$: each item is sealed once, independent of audience size. The only audience-dependent cost is key-wrap material, $O(\sum_i |\text{aud}(i)|)$ in constant-size ($\approx$ 60-byte) units. An item shared with k Consumers costs one ciphertext plus k wrapped CEKs, never k copies of the content.

8. Questions for the reviewer

1. **(G1) argument.** Is the composition of $\mathrm{SSS}_{2,3}$ with the per-share $h_k \oplus f_k$ two-factoring sound? And on separation: the signing key is derived from K_V by HKDF-Expand under a dedicated info label while K_V also keys AEAD wraps. Is expand-with-distinct-info clean separation, or should each purpose get an independent extraction?
2. **Anti-rollback.** Is signed-G with a monotonic v pinned in E enough against a replay or rollback by S, or do we need freshness such as a TEE-side version register or monotonic counter?
3. **Hybrid KEM.** Is $ss = \mathrm{KDF}(ss_{x25519} \,\|\, ss_{\mathrm{mlkem}})$ an adequate combiner (binding, robustness), and is hybridizing the *session* seal worth it given it is ephemeral?
4. **RWK lifetime.** The construction zeroizes RWK within the single reconstruction burst and keeps only per-Consumer PCKs for the window (§3.6, §6.2), rather than holding RWK reachable throughout. Is that per-PCK residual exposure acceptable given (2) in §6, and is the PCK indirection the right way to avoid retaining RWK without sealing eagerly to a future session key?
5. **OTP tier.** Given Consumers pre-share nothing, is the in-E HOTP plus provider-delivered codes the best achievable, and is the WebAuthn-on-first-contact hardening the right path?
6. **Key-wrap choice.** The CEK wraps use AES-256-GCM with the owning identifiers bound as explicit AAD and a fresh random nonce per wrap. Is this defensible against AES-KW (RFC 5649) or an MRAE mode, from a misuse-resistance standpoint?

References (informative)

RFC 5869 (HKDF) · RFC 5649 / 3394 (AES-KW, discussed as an alternative in §8 Q6) · RFC 9180 (HPKE) · FIPS 203 (ML-KEM) · FIPS 204 (ML-DSA) · RFC 7748 (X25519) · RFC 8032 (Ed25519) · RFC 4226 (HOTP) · Shamir, *How to Share a Secret* (CACM 1979) · NIST SP 800-38D (GCM) · Argon2 (RFC 9106).

Appendix A: Disposition of the independent peer review (rev 1 → rev 2)

An independent review (PC-member style) returned "major revision required", confirming the core cryptography (the $\mathrm{SSS}_{2,3} + h_k \oplus f_k$ composition, K_{read}/RWK HKDF domain separation, the hybrid-KEM combiner under a dual-PRF assumption, the worth of hybridizing the session seal, the lazy-RWK tradeoff, and AES-KW given the AEAD's $ad = id_i$ binding) and validating (G2) and (G4). The required revisions, all now incorporated:

1. **Trigger-Event attestation was undefined (the main gap).** Added §3.5: an *Event Notary* N signs τ; a signed assertion by itself is not sufficient, and disclosure is gated behind a mandatory *Owner-veto window* W, so a *forged* event (colluding $S + t - 1$ Agents + N) is defeated by the live Owner's veto. Threat (f) and the liveness assumption were added to §2; (G1) reframed as the threshold-secrecy-plus-veto-window guarantee.
2. **Anti-rollback needed a tamper-proof anchor.** §3.5 now specifies a *TEE-sealed monotonic version register* $v^\star$ (not a provider-stored v alone) as the hardening target, so S cannot replay an old signed G to re-grant a revoked Consumer. The current implementation enforces strict store-level version monotonicity plus a tamper-evident audit-chain freshness anchor; the

sealed register is scheduled pre-production hardening (§3.5).

3. **KMS policy made explicit.** The attestation-gated unwrap now requires attestation $\wedge \tau \wedge \geq t$ Agents $\wedge$ elapsed W; attestation alone never releases RWK.
4. **"Zero-knowledge" scoped.** Re-titled to *content* zero-knowledge; added a scope note that routing metadata is intentionally provider-visible.
5. **Initial Consumer session.** §6 now states the honest-but-curious-S assumption for first-session authentication and the conditions to harden it.

The reviewer's remaining points (Consumer tier below Owner tier; bounded TEE/RWK exposure window; attestation-UX as the human factor) were already disclosed in §6 and are retained as stated limitations.

Appendix B: Revision 3 note (2026-07-05)

Primitive descriptions corrected to match the implementation: the per-key wraps are AES-256-GCM with explicit associated data and a fresh random nonce per wrap, *not* RFC 5649 AES-KW (§1, §3.2, §3.6, §5, §8 Q6); signatures are hybrid Ed25519 + ML-DSA-65, verification requiring both components (§1, §5); and the key hierarchy is stated as built, where MK wraps a random vault key K_V, the RWK is an independent random key rather than a derived one, and the signing key is HKDF-Expand-derived from K_V under a dedicated info label (§3.1). The anti-rollback mechanism's implementation status is clarified in §3.5: store-level strict monotonicity plus a tamper-evident audit-chain anchor today, with the TEE-sealed monotonic register as scheduled pre-production hardening. References to K_{read} and AES-KW in the rev-1 review disposition (Appendix A) are retained as a historical record of what that review examined.

Appendix C: Revision 4 note (2026-07-09)

Added an at-rest refinement to §6 (limitation #5): operational PII (Owner and Consumer email, phone, legal name, and date of birth) is now encrypted at rest at the application layer (randomized AES-256-GCM, AAD-bound per row; equality lookups via a keyed-HMAC blind index), under a data key and index key that are KMS-wrapped and never stored in the database. This closes the *passive* at-rest exfiltration path (DB dump, snapshot, backup, injection read, or leaked credential now yield ciphertext, not contacts) without altering any formal goal: it makes *no* zero-knowledge claim over PII (the operating service holds the wrapping key and must decrypt to route notices, so an active or compromised operator can still read contacts), and the content goals (G2) and (G4) are untouched. This is the standard blind-index / envelope pattern (CipherSweet, AWS DB-Encryption-SDK). Implementation: `internal/piicrypto`, migrations 0060 (expand) then 0061 (plaintext drop).

Appendix D: Revision 5 note (2026-07-17)

An accuracy pass that corrects descriptions which had drifted from the implementation. The construction, the primitives, and the attested-TEE trust model are unchanged and were re-confirmed against the code (curves, hashes, KDF, AEAD mode, the 2-of-3 Shamir split over GF(2^8), the both-component hybrid signature, key sizes). Revision 5 corrects:

1. **Session channel (§1, §3.6, §5).** Re-labeled from "HPKE (RFC 9180)" to an *HPKE-style hybrid KEM-AEAD*. The combiner (HKDF of $ss_{x25519} \| ss_{\text{mlkem}}$ to an AES-256-GCM key) and

the session-nonce AAD are as previously described; the construction is not RFC 9180 HPKE (no HPKE key schedule, suite IDs, or labeled KDF).

2. **Owner-key authenticity (§3.1).** Replaced "pk_O bound into E's trust anchor at setup" with the built mechanism: an *RWK-sealed commitment* to pk_O that E opens only after threshold reconstruction. This is a stronger guarantee, since owner-key authenticity is tied to the threshold rather than to E's provisioning.
3. **Owner veto (§3.5).** Described as it is built, an *authenticated control action* enforced by the release state machine (an abort wins over any not-yet-begun reconstruction, under a row lock), not a signature $\mathrm{Sig.Sign}(sk_O, \texttt{"abort"} \| nonce)$ verified inside E (which E could not check pre-reconstruction anyway, per item 2). The (G1) guarantee is unchanged; the cryptographically-signed veto is named as a possible hardening.
4. **Signed audience (§3.3).** Clarified that the owner signature covers the entitlement edges and Consumer *identifiers* with the version, *not* the mutable delivery *contacts*, consistent with the §6.1 recipient-tier limitation.
5. **Per-Consumer wrap lifetime (§3.6, §6.2, §8 Q4, Abstract).** Corrected "lazy per-Consumer wrap" to *eager wrap in the single reconstruction burst with* RWK *zeroized before return*; only session delivery is lazy. This is favorable, since RWK is not held across the disclosure window, and it moots the earlier "lazy-RWK lifetime" tradeoff.
6. **At-rest PII scope (Appendix C).** Revision 5 initially noted that a plain marketing/waitlist email list (no vault data, not an Owner/Consumer record) was out of scope of the §6.5 at-rest encryption and stored in the clear. On 2026-07-18 that list was brought under the same at-rest envelope and blind-index pattern, removing the exemption; §6.5 reflects this.

On deployment status. The paper describes the production architecture: disclosure mediated by an attested TEE E with an attestation-gated KMS, the design the system is built on and launches with. It states its trust assumptions in §6 (the TEE as trust anchor; attestation verification as the decisive human factor). The single enforcement the paper marks as *not yet deployed*, unchanged since revision 3, is the TEE-sealed monotonic version register $v^\star$ for anti-rollback (§3.5, §8 Q2); attestation-gated key release was validated on production TEE hardware. Two internal source comments that described the enclave as *already* enforcing version monotonicity are being corrected to match §3.5 (that register is the pre-production hardening); this is a source-comment fix and does not affect the construction.

Appendix E: Revision 6 note (2026-08-01)

A presentation revision, prompted by external reader feedback on the PDF. The previous PDF was produced by printing a rendered web page, and that pipeline damaged the mathematics: several formulas shipped as raw source text, sub- and superscripts were mis-positioned throughout, and browser print artifacts appeared on every page. The PDF is now typeset with LaTeX. Two figure corrections ride along: Figure 1 previously labeled the Owner wrap key with the retired K_{read} name (renamed K_V in revision 3), and Figure 2 omitted the audience/state signature layer that the §5 table lists. The prose was lightened in the same pass (less typographic emphasis, no wording-pattern repetition). No definition, mechanism, claim, or security argument changed in this revision.