SoK: Cryptographic Erasure on Public Ledgers: Application-Layer Architectures, Key-Lifecycle Adversaries, and GDPR Art. 17 Sufficiency*

Yitebeier Aikebaier
Nexum Ledger Ltd
London, United Kingdom
jsjjd3690@gmail.com

Abstract

Public permissionless ledgers are append-only by design, yet are increasingly asked to host data subject to statutory erasure obligations (GDPR Art. 17, EDPB Guidelines 02/2025). The prevailing cryptographic response, redacting the chain itself, needs either a different chaining hash or the cooperation of every node, neither of which a controller on Bitcoin or Ethereum mainnet commands. This SoK systematizes the orthogonal application-layer design space, in which the ledger is left untouched and erasure is achieved by destroying the key material required to read the on-chain bytes. We organise the literature along two axes, data locus (on-chain ciphertext, anchor-only, or hybrid), key custody (single-custodian or threshold), and access gate (none, time, or witness), into a twelve-cell grid with canonical constructions from 1996 to 2026. We extend standard IND-CCA security with a key-lifecycle adversary model parametrised by HSM side-channel leakage, coercion fraction over the custody committee, and algorithmic-break horizon, give the security experiment as pseudo-code with adaptive corruption, cumulative leakage across key rotations, a decryption oracle, and a stronger post-destruction adversary class, and prove that the resulting Destruction-IND-CCA notion suffices for a semantic-security formalisation of the EDPB "render unrecoverable" criterion. We state what is provable per custody model: a concrete bound for single-custodian AEAD, a bound for Shamir threshold custody that grows with the number of rotations, an impossibility result for delay-only designs, and a reduction showing witness-gated erasure needs extractable witness encryption.

Evaluating seven reference architectures against eleven engineering, cryptographic, and regulatory criteria, including a production RegTech engine measured at 2.6×10^3 events/s single-threaded (10^4 projected with connection pooling), we find that the literature has concentrated on chain-rewriting while the application-layer branch lacks a common threat-model vocabulary. We close this gap and identify five open problems: post-quantum destruction security, the leakage threshold of a custody scheme, decentralised lifecycle audit (with a multi-custody extension covering encrypted mempools), composition with ZKP selective disclosure, and erasure under multi-regime retention conflict.

Keywords

cryptographic erasure, GDPR Article 17, key destruction, Destruction-IND, public ledgers, secure deletion

1 Introduction

Article 17 of the GDPR gives a data subject the right to have their personal data erased. This paper asks what that right can mean once the data has been written to a public ledger from which nobody can delete. The property that makes such ledgers useful, that nothing is ever removed, is the property that makes erasure impossible in the ordinary sense. It is a privacy question before it is a cryptographic one: the threats are to the person the data is about (recovery of the datum, linkage of what remains on the chain to them, and refusal to erase), and the yardstick is the one data-protection authorities apply. It is also current rather than hypothetical: the reference system evaluated in §6 is used by regulated firms to anchor records that name people, and those firms receive erasure requests.

Bitcoin and Ethereum are append-only by design: their security rests on history not being rewritable without breaking consensus. Over the last decade they have come to carry richer data, much of it *personal data* under Regulation (EU) 2016/679 [109]. Article 17 grants erasure; Recital 66 and the EDPB's *Guidelines 02/2025 on processing of personal data through blockchain technologies* [45] accept making data cryptographically unrecoverable as a form of erasure, provided the bar to recovery is documented and high.

Neither Bitcoin nor Ethereum stores personal data at the protocol layer; a transaction is amounts, scripts, and keys. The data this paper concerns is written by *applications* into the space the protocol leaves open (`OP_RETURN`, calldata, witness envelopes, contract storage), and it is there in quantity, identifying and sensitive material included [91]; the EDPB's blockchain guidelines [45] address exactly this practice. Three deployment patterns recur: identity anchors, where a decentralised-identifier document or a credential commitment is written on-chain [89, 108, 115]; audit trails, where a regulated firm anchors event records that name clients and documents [74, 112]; and asset metadata, where provenance or ownership records carry names and addresses. Why go on-chain at all, given the EDPB's own advice to keep personal data off it? Because the ledger supplies the half of the problem the controller cannot supply itself: a tamper-evident, third-party-verifiable record that an event happened, and later that it was erased, without trusting the controller's database. The tension between that audit value and the erasure right is the subject of this paper, and §5.5 says what the ledger may keep.

*Preprint, version 2 (November 2026). Supersedes version 1 (IACR Cryptology ePrint Archive, Report 2026/1109, 30 May 2026). Revised after review at PoPETs 2027, Issue 2.

A substantial line of cryptographic work addresses this tension by re-engineering the ledger itself: *redactable blockchains* replace the chaining hash with a *chameleon hash* [12, 41] whose trapdoor lets authorised parties edit a block, a branch mature enough to have its own systematic review [86]. It is *structurally unavailable* on Bitcoin and Ethereum mainnet, where the chaining hash is fixed by consensus. Redaction that keeps the chaining hash and instead lets each node delete transaction data while proving in zero knowledge that the coin or contract state is unchanged has been designed for Bitcoin itself [27, 78] and for permissioned smart-contract chains [13]; it erases what a cooperating node stores, and a controller on a public chain commands no node but its own. For a practitioner who has already anchored data and must answer an Article 17 request, a chameleon hash is a different chain and node-side redaction is other people's nodes; neither is a tool.

What the practitioner does have is the other family: leave the ledger alone and destroy the material needed to read it. The family is old and wide, from crypto-erasure of storage [54, 105] through threshold sharing [81, 110], time-lock encryption [106], and witness encryption [56] to a grey literature of enterprise products. Its members are rarely described in one vocabulary, and almost none says what a break would have to look like for its claim of erasure to fail.

This paper systematizes that family. Taking the ledger as unchangeable, it asks under what conditions destroying key material erases the plaintext in a sense that survives the EDPB's test, organised by where the bytes live and who holds the key (Table 2), by a security experiment in which destruction, compromise, rotation, and a stronger future adversary are explicit events (§4), and by a theorem relating that experiment to the regulatory test (§5).

1.1 Scope

We deliberately fence the problem narrowly. *In scope:* constructions that leave the underlying chain untouched and achieve erasure through cryptographic key destruction; architectures that place encrypted payloads directly on-chain and rely on destruction of decryption keys; architectures that anchor only a commitment and store ciphertext off-chain; time-lock and witness-encryption gated release as erasure mechanisms; the adversary model and compliance analysis that cover these.

Explicitly out of scope. Chameleon-hash and trapdoor-based redactable blockchains [12, 41]: the ICA3PP 2025 SLR [86] already systematizes this branch, and we do not replicate its coverage. Consensus-level edits and hard-fork-based redaction, mutable-transaction designs that vote on edits [101], and node-local erasure that changes what a node stores rather than what the chain says [52], including its proof-carrying forms for Bitcoin [27, 78]. Permissioned-chain policy overrides, where the chain operator can unilaterally rewrite history, and smart-contract-aware redaction on such chains [13]. Off-chain-only deletion, where the ledger plays no role. Secure deletion at the storage-media layer, which Reardon et al. [105] cover. Sectoral record-keeping mandates that may independently compel retention of a readable record, such as the SRA Accounts Rules for UK legal practices or anti-money-laundering retention under the UK MLR 2017: this paper formalises erasure under Art. 17, not retention under sectoral law. The interaction between the two, where personal data is embedded in a mandatorily-retained record, is surfaced as OP5 (§7.5).

The paper's unit of analysis is an *architecture* (a combination of on-ledger data layout, off-ledger storage, and key-lifecycle protocol), not an individual cryptographic primitive.

1.2 Contributions

(1) **Taxonomy.** We classify application-layer erasure by data locus (on-chain ciphertext / anchor-only / hybrid), key custody (single / threshold), and access gate (none / time / witness), a 3×4 grid with a representative construction per cell.
(2) **Key-lifecycle adversary model.** We give a phased security experiment, Destruction-IND-CCA, with adaptive custodian corruption, cumulative leakage across key rotations, a decryption oracle while keys exist, and a stronger post-destruction adversary class; it is parametrised by (i) side-channel leakage, (ii) corruption budget, (iii) post-destruction adversary class. We state what is provable per custody model: concrete bounds for single and Shamir-threshold custody, an impossibility result for delay-only designs, and a reduction showing witness-gated erasure requires extractable witness encryption.
(3) **GDPR Art. 17 sufficiency framework.** We formalise the EDPB 02/2025 "render unrecoverable" test as semantic security of the post-destruction view, with the datum sampled from an arbitrary distribution and outside knowledge given to the attacker, prove that Destruction-IND-CCA suffices for it, and connect the on-chain residue to the singling-out, linkability and inference tests of the EDPB's 2026 anonymisation guidance.
(4) **Evaluation.** We assess 7 representative architectures against 11 criteria, with measured values from a production Rust engine (2.6×10^3 events/s single-threaded, 10^4 projected with pooling) and stated assumptions for each.
(5) **Open problems.** Five gaps the framework exposes: post-quantum destruction security; the leakage threshold of a custody scheme; decentralised lifecycle audit; composition with ZKP selective disclosure; and erasure under retention conflict.

2 Background & Related Systematizations

2.1 Cryptographic erasure: origins and assumptions

The idea that *deletion of key material is equivalent to deletion of plaintext* predates blockchain by two decades. Boneh and Lipton's revocable backup system [25] made files on backup tapes unreadable by destroying their keys, and Garfinkel and Malan's 2006 analysis of free-space sanitisation [54] generalised the framing that underpins this paper: rather than overwriting every copy of a ciphertext (which is impossible once the ciphertext has been broadcast or replicated) one overwrites only the key. NIST SP 800-88 Rev. 1 [95] later canonised the term *cryptographic erase* and listed it as an acceptable sanitisation method for encrypted storage. Early systems already map onto the custody axis of §3: Perlman's Ephemerizer [98] destroys keys on a schedule and

FADE [111] on policy (a single custodian, K1), while Vanish [58] scattered Shamir shares across a DHT and let churn erase them (K2), and was broken within a year by cheap Sybil harvesting of the shares before they expired [117].

Reardon, Basin, and Capkun's SoK on secure deletion [104, 105] remains the most complete systematization at the storage-media layer, with [31] establishing the same primitive in enterprise compliance. Its adversary model assumes a single-device deletion primitive and reasons about residual data on that device; it does not anticipate data deliberately replicated to a public ledger, and so lacks the vocabulary needed once the ciphertext has been broadcast to 10,000 nodes. Later work on assured deletion in the cloud [103], publicly verifiable deletion of a key [70], deniable encryption with deletion [124], memory erasure [32, 60], and formal analysis of deployed systems [7, 9] refines the toolkit but keeps the single-device assumption.

2.2 GDPR Article 17, Recital 66, and the EDPB 02/2025 guidance

Article 17 of Regulation (EU) 2016/679 grants the data subject a right to erasure "without undue delay." Recital 66 clarifies that the controller "should take reasonable steps" to inform third parties processing the data of the erasure request, but stops short of specifying a cryptographic criterion for what constitutes erased.

The EDPB's *Guidelines 02/2025 on processing of personal data through blockchain technologies* [45] are the first regulatory instrument to address the tension directly. Two passages matter here. The Guidelines accept that overwriting is "technically impossible" on a public permissionless ledger, and they accept *rendering the data unrecoverable through cryptographic means* as a substitute, provided the controller documents the threat model under which it is unrecoverable. Two earlier instruments anticipated them: the CNIL's 2018 analysis [39] ranked the forms in which personal data may touch a chain (a commitment, a keyed hash, a ciphertext) and treated deletion of the opening or of the key as a step towards erasure, and the European Parliament's STOA study [49] discussed destruction of the private key as an Article 17 substitute. Finck [48] and Finck and Pallas [50] supplied the vocabulary the Guidelines inherit, and we use it below; §5.5 makes the CNIL ranking precise. The Guidelines do not say what the documentation should contain or how it should be judged; a controller who writes "we destroyed the key" has no yardstick, and §5 supplies one.

2.3 Existing systematizations and their coverage gap

Prior systematizations approach the problem from four directions, none of which reaches the design space of this paper.

Reardon, Basin, and Capkun [105] systematize secure deletion at the storage-media layer; data replicated by design to third parties is outside their scope. The chain-rewriting branch (chameleon hashes, trapdoor edits, permissioned redaction) has the ICA3PP 2025 systematic review [86] and the overview of Calandra et al. [33], which together with [84, 96, 119, 120, 123] cover more than forty-five papers, and the proof-carrying redaction line for Bitcoin and for permissioned contract chains [13, 27, 78]; none concerns constructions that leave the chain alone. Hillmann et al. [73] propose the pattern of our L1×K1 cell as a single construction, on-chain ciphertext whose off-chain key is deleted, without an adversary model or a compliance criterion. The surveys of blockchain mutability and GDPR compliance [19, 71, 100, 122], the MDPI *Future Internet* survey [10], and the *Scientific Reports* analysis [62] all list key deletion among the options, at survey level. Esu [44] argues, as a lawyer, that key destruction is admissible erasure under EU and UK storage-limitation rules, without a cryptographic adversary; Ghaly et al. [59] build a tokenisation scheme for the right to be forgotten on permissioned chains, without a unifying model.

The closest formal predecessor is the *deletion-compliance* notion of Garg, Goldwasser, and Vasudevan [57], with the privacy-decoupled variant of Godin and Lamontagne [61]. It compares a collector's state and outgoing messages after deletion with an execution in which the data was never shared, and treats any surviving artefact as a violation; it assumes a collector that can scrub its own storage. We return to it in §5.3, since the ledger setting is the one it assumes away.

Table 1 summarises the gap: no prior work combines (i) application-layer constructions on non-redactable chains, (ii) a key-lifecycle adversary, and (iii) a formal Art. 17 criterion.

	App-layer construct.	Key-lifecycle adversary	Art. 17 criterion	Non-red. chain
Reardon 2013 [105]	partial	no	no	n/a
MDPI 2023 [10]	partial	no	no	no
ICA3PP 2025 [86]	no	no	no	no
Sci. Rep. 2022 [62]	partial	no	no	partial
Esu 2025 [44]	no	no	partial	partial
GGV 2020 [57]	no	no	partial	n/a
Hillmann 2020 [73]	yes	no	no	yes
Politou 2021 [100]	partial	no	no	partial
Sağlam 2023 [19]	partial	no	partial	yes
This SoK	**yes**	**yes**	**yes**	**yes**

Table 1: Coverage matrix. "Partial" indicates the dimension is addressed incidentally, not as a primary contribution. Esu 2025 covers Art. 17 compliance at the legal-analysis (not cryptographic) level.

3 Design Space and Taxonomy

We organise the application-layer design space along two axes, the second of which has two components. *Data locus* (Axis 1) says where the bytes that must become unreadable physically reside. *Key custody* (Axis 2a) says who holds the capability to read them and how it is destroyed; an optional *access gate* (Axis 2b) says what additional condition, time or a witness, must hold before that capability can be used. Gates are not custody models: a time-lock or a witness-encryption layer can sit on top of a single custodian or of a committee, and destroys nothing by itself (§4.7). We keep the labels K1–K4 for continuity with the literature, with the understanding that K1 and K2 are custody models and K3 and K4 are gates. Table 2 locates every cell and gives its canonical construction.

3.1 Axis 1: Data locus

Let m be the personal datum and c its ciphertext. The three loci are distinguished by what the ledger stores.

L1: on-chain ciphertext. The ledger stores c itself, typically as Ethereum calldata or contract storage, where a transaction carries kilobytes at predictable cost; on Bitcoin the `OP_RETURN` output is capped at 80 bytes and Taproot envelopes carry more at a fee premium, so L1 is realistic there only for short ciphertexts or via splitting. Tamper evidence is maximal; erasure is maximally hard: c remains, by consensus, for as long as the chain exists, and the entire burden of erasure falls on the key.

L2: anchor-only. The ledger stores only a commitment $\mathrm{Com}(c)$, a hash, Merkle root, polynomial commitment or accumulator, and *no other byte derived from m*. The ciphertext lives in an off-chain encrypted store (IPFS [21], S3, a self-hosted CAS) whose availability [69] the commitment does not guarantee; the deployed pattern on Bitcoin is OpenTimestamps [113], the anchor of the reference implementation (§6). Erasure has two parts, destroy the key and delete the off-chain ciphertext where that is in the controller's power, and the commitment must be *hiding* for the residue to pass the linkability test of §5.5.

L3: hybrid. The ledger stores a commitment to an off-chain body *and* an on-chain header that itself contains fields derived from m (possibly encrypted), kept on-chain so that they can be queried or selectively disclosed. The pattern is ubiquitous in decentralised identity [89, 115]: a credential commitment and an encrypted header on-chain, the credential body off-chain. What distinguishes L3 from L2 is exactly the presence of bytes beyond the commitment that derive from personal data: the header follows the L1 logic (only key destruction can erase it) while the body follows L2, and the two must be erased separately, which makes field-level erasure ("forget the income field but not the passport hash") a distinct design goal.

Design tension. L1 maximises tamper evidence and raises the bar for erasure to "permanent ciphertext"; L2 eases erasure but rests on off-chain availability a regulator may find hard to audit; L3 inherits both obligations plus composition risk between header and body.

3.2 Axis 2a: Key custody

K1: single custodian. One key, held in an HSM or an enclave with attestable zeroisation. "Destroy" is the zeroisation primitive plus its attestation. Standard in enterprise RegTech and the custody model of the reference implementation. Theorem 1 gives its security.

K2: (t, n)-threshold custody. The capability is split across n custodians by Shamir sharing [110], a DKG such as FROST [81], or a proactive scheme such as CHURP [90]; any t can decrypt; destruction is a quorum refusing to reconstruct and then erasing shares, with periodic refresh. CALYPSO [80] is the deployed form, a threshold access-control committee whose every decryption grant is logged on the ledger; Benhamouda et al. [23] host the committee on the chain itself; YOSO [36] and asynchronous variants [125] relax liveness; threshold-homomorphic constructions over class groups [29], deniable sharing [35], and the deployed Thetacrypt service [16] extend the primitive set. Theorem 2 gives its security and its leakage ceiling.

	Custody (2a)		Gate (2b), over K1 or K2	
	K1 single	**K2** (t, n)	**K3** time	**K4** witness
L1 on-chain ct	HSM-destroyed AES [73]‡	shares destroyed [23, 36, 90]	timed release [24, 76, 106]	WE on L1 [40, 56]
L2 anchor-only	**Ref. impl.** (§6.5), [15, 21, 126]	CALYPSO [67, 80]	IPFS + tlock [24, 82]	–
L3 hybrid	encr. header [89, 115]	encr. header + (t, n) [115]	†	–

Table 2: The design grid with one canonical construction per cell. Shaded: the reference implementation. "—" = no peer-reviewed instantiation; † position-paper only; ‡ on Bitcoin and Ethereum L1 the deployed pattern is anchor-only rather than on-chain ciphertext at scale, so the cell is sparse although the constructions above populate it. A gate column entry inherits the custody model beneath it.

3.3 Axis 2b: Access gates

K3: time gate. The capability becomes usable only after a wall-clock delay or a sequential computation: time-lock puzzles [106], VDF-gated release [24], or a beacon such as drand [82]; lattice-based puzzles [3] give a post-quantum instance. A time gate is a *release* policy; Theorem 3 shows it is not an erasure policy for any horizon beyond the delay. Liu et al. [85] build time-lock encryption from a blockchain and witness encryption, the construction McFly later made practical; encryption to the future [34] addresses a message to a committee that does not yet exist, and timed secret sharing [76] composes a time gate with K2 custody; scalable and homomorphic puzzles [1, 2] extend the gate along throughput and composability.

K4: witness gate. The capability is usable only if a witness for an NP statement is known [56, 66]; erasure means making the witness permanently unavailable. General witness encryption rests on indistinguishability obfuscation, but restricted-language instances are practical: Derler–Slamanig for algebraic languages [40], signature-based witness encryption [14], and McFly [43], which gates on a future signature and so composes K4 with K3. Proposition 4 shows that a witness gate relocates the question to the custody of the witness and that erasure needs *extractable* witness encryption.

3.4 Populating the grid

Of the twelve cells, nine are populated by peer-reviewed constructions or deployed systems, one (L3×K3) has position-paper coverage only, and two (L2×K4, L3×K4) are empty: expressible, but with no deployed or peer-reviewed instance. The empty cells are gaps that §7 promotes to open problems. Every cell inherits its security from its custody model and, for the gate columns, from the composition results of §4.7: a K3 cell is secure only for horizons shorter than its delay, and a K4 cell is as secure as the custody of its witness.

3.5 The adversary axis

The grid classifies *designs*, not *security claims*: two HSM systems in the same cell may claim very different things about what break

would void their erasure. Section 4 adds a transverse axis, the threat model $(\varepsilon, k, \mathcal{C})$ of Def. 1, over which §5 is stated.

4 Key-Lifecycle Adversary Model

This section makes the adversary axis precise: parties (§4.2), scheme syntax with key refresh (§4.3), threat parameters (§4.4), the experiment as pseudo-code (§4.5), and what is provable per custody model (§4.7; proofs in Appendix C).

4.1 Why IND-CCA is not enough

IND-CCA is the right starting point and the wrong stopping point, for two reasons that are properties of the *experiment*, not of any cipher chosen to instantiate it.

(i) The experiment has no key lifecycle. In IND-CCA the key exists from KeyGen to the final guess. There is no event "the key is destroyed", and so the property we care about cannot even be stated: an adversary who compromises the custodian *after* destruction must learn nothing, while one who compromises it *before* learns everything. Destruction, coercion of custodians, key rotation, and the imperfect physics of zeroisation are events in time, and IND-CCA has no time.

(ii) The experiment fixes one adversary class for its whole duration. The IND-CCA quantifier chooses the adversary's computational class once, at the start. A ciphertext on a public ledger outlives that choice: it is written under today's adversary and read, decades later, by tomorrow's. The statement we need quantifies over a *pair* of adversaries, the one who attacks while keys exist and the one who attacks after they are gone, and allows the second to be stronger than the first.

The quantum horizon is one instance of (ii); the coerced or hacked custodian is one instance of (i), and the Sybil harvest that defeated Vanish [117] is its historical form. No choice of primitive fixes either; the phased experiment built next does, by giving the key's lifecycle its own oracles. The post-destruction phase is the erasure counterpart of post-compromise security [38], with destruction as the healing step after which the adversary is allowed everything.

4.2 Parties

Five roles appear in every architecture of §3; we name them once and map them to ledger vocabulary.

- The **controller** (GDPR Art. 4(7)) decides what is written, declares the threat model $\mathcal{T}$ of Def. 1, and issues the erasure instruction. On a ledger it is the application operator that submits transactions.
- The **custodians** $P_1, \ldots, P_n$ hold the decryption capability and execute Refresh and Destroy. A custodian is an HSM ($n = t = 1$), a committee member, or a beacon; it is never the ledger.
- The **data subject** is the person m is about. The data subject never holds keys; the guarantees of §5 are stated from their point of view.
- The **ledger** is an append-only broadcast channel, including the mempool through which transactions pass before inclusion. Validators, miners, and sequencers see bytes and nothing else; every party, including the adversary, can read it forever. Encrypted mempools add a second ciphertext of the same bytes under different keys (OP3).
- The **auditor** is any third party who later wants to verify that destruction happened (criterion C9), typically a full node or a regulator.

The adversary is one entity that is at once a ledger reader (always), a corrupter of custodians (bounded by k), an observer of side channels (bounded by ε), a client of the decryption service (the CCA oracle), and, after destruction, a cryptanalyst of a stronger class $\mathcal{C}$. Three threats to the data subject follow: recovery of m after erasure (§5); linkage of the on-chain residue to the data subject (§5.5); and refusal to erase, a liveness failure outside the model and the reason criterion C9 exists.

4.3 Scheme syntax

A *custody scheme* is a tuple $\Pi = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Refresh}, \mathsf{Destroy})$ over n custodians with threshold $t \le n$. Shares carry an epoch index $e \in \mathbb{N}$.

- $\mathsf{Setup}(1^\lambda, n, t) \to (pp, \mathsf{sk}_1^{(0)}, \ldots, \mathsf{sk}_n^{(0)})$. $n = t = 1$ is single custody (K1); $1 < t \le n$ is threshold custody (K2).
- $\mathsf{Enc}(pp, m) \to c$ uses public parameters only.
- $\mathsf{Dec}(pp, \{\mathsf{sk}_i^{(e)}\}_{i \in S}, c) \to m$ or $\perp$, for any S with $|S| \ge t$ and all shares from the same epoch e.
- Refresh is an interactive protocol among the custodians that maps epoch-e shares to epoch-$(e{+}1)$ shares of the *same* decryption capability. Each custodian then runs Destroy on its epoch-e share. For $n = t = 1$, Refresh is a re-wrap of the key or the identity.
- $\mathsf{Destroy}(\mathsf{sk}_i^{(e)}) \to \rho_i^{(e)}$ is local and outputs the custodian's *residual state*: whatever the device still holds after zeroisation. Ideally $\rho_i^{(e)} = \perp$.

Correctness: decryption with t same-epoch shares returns m. The access gates of Axis 2b (time, witness) are not part of this syntax; they sit on top of a custody scheme and are treated in §4.7.

4.4 Threat parameters

Definition 1 (Destruction threat model)**.** A destruction threat model is a triple $\mathcal{T} = (\varepsilon, k, \mathcal{C})$:

(1) $\varepsilon \in [0, 1]$ is the *per-share leakage budget* ($\varepsilon = 1$: the whole share may leak, the honest budget while a copy of it survives). For every custodian i and epoch e, the total number of bits the adversary obtains about $\mathsf{sk}_i^{(e)}$, from side channels during the epoch, from the destruction event, and from the residual state $\rho_i^{(e)}$, is at most $\ell = \lfloor \varepsilon \cdot |\mathsf{sk}_i^{(e)}| \rfloor$. Leakage is *local*: each leakage function is an arbitrary function of one share alone [22], with no access to the scheme's oracles (in the ideal-cipher model, none to E). Tables in this paper report the budget as $\ell = \varepsilon\,|\mathsf{sk}_i^{(e)}|$ bits per share per epoch; for threshold custody that is the operative unit (Theorem 2). Leakage requested before the challenge ciphertext exists is adaptive; leakage after it is produced by a *side-channel profile* $(L_i)_{i \in [n]}$ that the adversary fixes before the challenge (Remark 2).

(2) $k < t$ is the *corruption budget*. The adversary corrupts custodians adaptively and holds at most k of them at any time, in

Experiment $\mathrm{Exp}^{\mathrm{dIND}}_{\Pi,\mathcal{A}}(\lambda;\, \varepsilon, k, \mathcal{C})$, with $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$:

$(pp, \mathsf{sk}^{(0)}_{1..n}) \leftarrow \mathrm{Setup}(1^\lambda, n, t)$; $e \leftarrow 0$; $C_0, R_0 \leftarrow \emptyset$; $\ell_i^{(e)} \leftarrow 0$;
$c^* \leftarrow \perp$; prof $\leftarrow \perp$; $b \leftarrow \{0,1\}$
$st \leftarrow \mathcal{A}_1^{\mathcal{O}}(pp)$ ▷ Phases 1–2, $\mathcal{A}_1 \in$ PPT
$b' \leftarrow \mathcal{A}_2^{\mathrm{Residue}}(st)$ ▷ Phase 3, $\mathcal{A}_2 \in \mathcal{C}$
return $[\, b' = b \,]$

Oracles $\mathcal{O} = \{$Corrupt, Release, Leak, Dec, Profile, Challenge, Refresh, Destroy$\}$; all are disabled once Destroy has been called:

Corrupt(i): require $|C_e \cup \{i\}| \le k$; $C_e \leftarrow C_e \cup \{i\}$; return $\mathsf{sk}_i^{(e)}$ (and, while $i \in C_e \setminus R_e$, P_i's view of every later Refresh).
Release(i): $R_e \leftarrow R_e \cup \{i\}$; i stays in C_e until the next Refresh ▷ P_i is rebooted; its next share is fresh
Leak(i, f): require $c^* = \perp$, $i \notin C_e$, $\ell_i^{(e)} + |f(\cdot)| \le \varepsilon |\mathsf{sk}_i^{(e)}|$; $\ell_i^{(e)}$ += $|f(\cdot)|$; return $f(\mathsf{sk}_i^{(e)})$.
Dec(c): require $c \neq c^*$; return $\mathrm{Dec}(pp, \{\mathsf{sk}_i^{(e)}\}_{i \in S}, c)$ for any S with $|S| = t$.
Profile($L_1, \ldots, L_n$): require $c^* = \perp$; prof $\leftarrow (L_i)_i$.
Challenge(m_0, m_1): require $c^* = \perp$, $|m_0| = |m_1|$, prof $\neq \perp$; $c^* \leftarrow \mathrm{Enc}(pp, m_b)$; return c^*.
Refresh(): run Refresh among $P_1..P_n$, giving $\mathcal{A}_1$ the views of $C_e \setminus R_e$; for all i: $\rho_i^{(e)} \leftarrow \mathrm{Destroy}(\mathsf{sk}_i^{(e)})$, and for $i \notin C_e$ charge $|\rho_i^{(e)}|$ to $\ell_i^{(e)}$; return $\big(L_i(\mathsf{sk}_i^{(e)})\big)_{i \in C_e}$ if prof $\neq \perp$, charging $|L_i(\cdot)|$ likewise; $C_{e+1} \leftarrow C_e \setminus R_e$; $R_{e+1} \leftarrow \emptyset$; $e \leftarrow e{+}1$.
Destroy(): for all $i \notin C_e$: $\rho_i^{(e)} \leftarrow \mathrm{Destroy}(\mathsf{sk}_i^{(e)})$; return $\big(L_i(\mathsf{sk}_i^{(e)})\big)_{i \notin C_e}$, charged as above; disable all other oracles.
Residue(i): return $(\rho_i^{(e')})_{e' \le e}$ for *any* $i \in [n]$ (no corruption budget after destruction).

Figure 1: The Destruction-IND-CCA experiment. A party corrupted at any point of epoch e stays in C_e until the next Refresh **even if released (R_e), so $|C_e| \le k$ bounds the shares of any one epoch the adversary ever holds; a released party rejoins as honest with a fresh share, as in proactive secret sharing. $\ell_i^{(e)}$ counts the bits leaked about share $\mathsf{sk}_i^{(e)}$ from** Leak**, from the profile at destruction, and from the residual state; an adversary is *admissible* if $\ell_i^{(e)} \le \varepsilon\,|\mathsf{sk}_i^{(e)}|$ for every i, e at the end of the run. Corrupted custodians' shares are not charged: the adversary holds them outright.**

the mobile-adversary sense of proactive secret sharing [72]: a corruption persists across a Refresh (the corrupted party sees its new share) until the adversary releases the party; a released party still counts against the budget of the epoch in which it was corrupted, since its share of that epoch is known, and is rebooted at the next Refresh with a share unknown to the adversary. After destruction the budget is lifted: every custodian may be corrupted.

(3) $\mathcal{C}$ is the *post-destruction adversary class*. $\mathcal{C}_\infty$ is classical probabilistic polynomial time; $\mathcal{C}_\mathrm{Q}$ is quantum polynomial time with quantum access to the primitives (Grover, Shor). A controller's horizon "T years" (Table 3) is a forecast that *selects* $\mathcal{C}$; it is not itself a parameter of the experiment.

We keep $(\varepsilon, k/n, T)$ as shorthand elsewhere in the paper, with $T \mapsto \mathcal{C}_T$.

4.5 The Destruction-IND-CCA experiment

Definition 2 (Destruction-IND-CCA)**.** Let Π be a custody scheme and $\mathcal{T} = (\varepsilon, k, \mathcal{C})$ a threat model. Π is *Destruction-IND-CCA secure under* $\mathcal{T}$ if for every admissible $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with $\mathcal{A}_1$ PPT and $\mathcal{A}_2 \in \mathcal{C}$ that respects the oracle preconditions and the leakage budget of Figure 1,

$$\mathbf{Adv}^{\mathrm{dIND}}_{\Pi,\mathcal{A}}(\lambda; \mathcal{T}) \;:=\; \Big|\Pr\big[\mathrm{Exp}^{\mathrm{dIND}}_{\Pi,\mathcal{A}}(\lambda; \mathcal{T}) = 1\big] - \tfrac{1}{2}\Big| \;\le\; \mathrm{negl}(\lambda).$$

The experiment has three phases. In the *lifetime* phase the adversary corrupts, leaks, decrypts, and rotates keys adaptively and in any order, and at some point obtains the challenge: the ciphertext is written to the ledger and lives for years. In the *destruction* phase the honest custodians zeroise and the adversary observes the event through its profile. In the *post-destruction* phase a possibly stronger $\mathcal{A}_2$ holds the ledger, every residual state, and all earlier knowledge. The decryption oracle exists while keys do and disappears with them.

Remark 1 (Recovering IND-CCA2)**.** Take $n = t = 1$, $k = 0$, $\varepsilon = 0$, $\mathcal{C} = \mathcal{C}_\infty$, no Refresh call, and Destroy outputting $\perp$. Then Leak, Profile, and Residue are vacuous, Dec is available before and after the challenge on all $c \neq c^*$, and $\mathcal{A}_2$ receives nothing beyond $\mathcal{A}_1$'s state. This is exactly the IND-CCA2 experiment [20]. Every other setting of $\mathcal{T}$ strictly strengthens the adversary.

Remark 2 (Why the post-challenge leakage is a fixed profile)**.** If the adversary could choose a leakage function after seeing c^*, it would choose $f(\mathsf{sk}) =$ "the first $\varepsilon|\mathsf{sk}|$ bits of $\mathrm{Dec}_{\mathsf{sk}}(c^*)$" and win outright for any $\varepsilon > 0$. That is not a side channel of a zeroisation but a decryption that reads the ledger. The physical channels of a destruction event (power, timing, remanence) are functions of the device and the key bits it wipes and exist before the ciphertext does, so Profile makes the adversary commit to them before the challenge, the standard restriction of the bounded-leakage literature [4, 93]; pre-challenge leakage stays fully adaptive.

Remark 3 (Why $k < t$, and the counting bound)**.** If $k \ge t$ the adversary reconstructs the key in epoch 0 and wins without any destruction event; Def. 2 measures what remains when it cannot. A softer boundary follows from counting: k full shares plus ε of each of the $n{-}k$ honest ones determine a linear secret once $k{+}(n{-}k)\,\varepsilon \ge t$, so we require $k + (n-k)\,\varepsilon < t$. It is necessary, not sufficient: Theorem 2(c) shows the leakage resilience of linear schemes fails at $\log_2 t$ bits per share, long before this bound.

Remark 4 (Phases and the horizon)**.** The split into $\mathcal{A}_1 \in$ PPT and $\mathcal{A}_2 \in \mathcal{C}$ is what the horizon T of Table 3 means operationally: keys were destroyed under today's adversary, and the question is whether an adversary of class $\mathcal{C}$, arriving later with the ledger and the residue, can distinguish. Declaring $T = \infty$ asserts $\mathcal{C} = \mathcal{C}_\infty$; declaring $T = 15$ asserts $\mathcal{C} = \mathcal{C}_\mathrm{Q}$ within fifteen years (OP1).

4.6 Parameter ranges for deployed systems

Table 3 gives the leakage budgets a controller can justify declaring, in bits per share per epoch, for reference settings, together with k/n and the horizon T that selects $\mathcal{C}$ as in Remark 4.

4.7 What is provable per custody model

The definition is only as useful as the schemes that satisfy it. We state one result per column of the grid; proofs are in Appendix C. Throughout, AE is a nonce-based authenticated encryption scheme [107] with a κ-bit key whose block cipher is modelled as an ideal cipher E, and $\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(q_D)$ is the AE advantage of

Setting	ℓ (bits / share / epoch)	k/n	T (yrs)
No zeroisation (key file deleted, medium not wiped)	$\lvert sk\rvert$ ($\varepsilon = 1$)	n/a	∞
Software zeroisation, commodity server	$\leq$ 8 (declared allowance for remanence)	n/a	∞
Certified HSM (FIPS 140-3 L4), attestable zeroisation	0 (attested)	n/a	∞
Reference impl. (current: software KMS; dumps rewritten every 10 min; master key unrotated)	$\lvert sk\rvert$ for $\leq$ 10 min after destruction, then ≤ 8	n/a	~ 15
Reference impl. (target: FIPS-L4 HSM; backups re-keyed on destruction)	0	n/a	~ 15

Table 3: Leakage budgets a controller can justify declaring, as $\ell = \lfloor \varepsilon \lvert sk\rvert \rfloor$ bits per share per epoch. Declarations, not measurements: the software figure is an allowance for remanence after an overwrite; "attested" takes the device's attestation at face value. For the reference implementation the dominant residual state is the backup, not the device: a destroyed, wrapped key survives in the daily dumps, under an unrotated master key, until a job that runs every ten minutes (deployed 11 October 2026; before that, for the 14-day retention) rewrites them, so for that window $\varepsilon = 1$. ∞ denotes $\mathcal{C}_\infty$; $T \approx 15$ denotes $\mathcal{C}_Q$.

the mode of operation when its block cipher is replaced by a uniformly random permutation, against one encryption query and q_D decryption queries.

Theorem 1 (K1: single custodian). *Let Π_1 be the scheme with* Setup *sampling $K \leftarrow \{0,1\}^\kappa$,* $\mathsf{Enc} = \mathsf{AE.Enc}_K$, $\mathsf{Dec} = \mathsf{AE.Dec}_K$, Refresh *advancing the epoch index and leaving K in place (so the per-epoch destruction inside Figure 1's* Refresh *erases nothing), and* Destroy *zeroising K. For every $\varepsilon \in [0,1]$ and every adversary against $(\varepsilon, 0, \mathcal{C}_\infty)$ that triggers E epochs with $(E+1)\varepsilon < 1$ and makes q_E ideal-cipher queries in total and q_D decryption queries,*

$$\mathbf{Adv}^{\mathrm{dIND}}_{\Pi_1,\mathcal{A}} \;\leq\; \mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(q_D) \;+\; q_E \cdot 2^{-(1-(E+1)\varepsilon)\kappa}.$$

Against $(\varepsilon, 0, \mathcal{C}_Q)$ the second term becomes $O\big(q_E \cdot 2^{-(1-(E+1)\varepsilon)\kappa/2}\big)$, by the one-way-to-hiding lemma and its semi-classical search bound [11] applied to a key of average min-entropy $(1-(E+1)\varepsilon)\kappa$ given the leakage.

Each destruction event costs at most $\ell = \lfloor \varepsilon\kappa \rfloor$ bits of key and nothing else: for AES-256 and a single epoch, attested zeroisation ($\ell = 0$) loses nothing, the software allowance of Table 3 ($\ell \leq 8$) leaves 248 bits classically and 124 against $\mathcal{C}_Q$, and the reference implementation's backup window ($\ell = \kappa$) leaves nothing. A key refreshed in place spends a further ℓ bits per epoch, the $(E+1)$ factor, as Theorem 2(b) charges a committee.

Theorem 2 (K2: Shamir threshold custody). *Let Π_2 share $K \in \mathbb{F}_p$ ($p > n$ prime; write $\kappa = \log_2 p$) with a (t,n) Shamir scheme [110], encrypt with AE_K, refresh by adding a fresh sharing of 0 over private channels [72], and destroy by zeroising shares. Write $\ell = \lfloor \varepsilon \lvert sk\rvert \rfloor$, $\lvert sk\rvert = \lceil \kappa \rceil$, for the per-share, per-epoch budget in bits. Call an adversary epoch-static if its leakage is non-adaptive (all of it through* Profile*) and the set of parties it corrupts in each epoch is fixed before it sees any share of that epoch; it may still choose each epoch's set from everything learned in earlier epochs.*

(a) If $\ell \leq 1$ and $(t-k)/(n-k) \geq \gamma$ for a constant $\gamma > c \approx 0.78$ and prime $p \geq p_0(\gamma)$, then for every epoch-static adversary against $(\varepsilon, k, \mathcal{C}_\infty)$ that triggers E epochs,

$$\mathbf{Adv}^{\mathrm{dIND}}_{\Pi_2,\mathcal{A}} \;\leq \mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(q_D) \;+\; q_E/p \;+\; (E+1)\cdot 2^{-(3\log_2 \pi - 2)(\gamma - c)(n-k)},$$

where the last term is the insider local-leakage bound of Maji et al. [88, Thm. 2]; Klein and Komargodski [79] lower the admissible ratio to 0.69 in the eavesdropping case.

(b) For every ℓ with $(E+1)\,n\,\ell < \kappa$, and every admissible adversary against $(\varepsilon, k, \mathcal{C}_\infty)$ that triggers E epochs, adaptive or not,

$$\mathbf{Adv}^{\mathrm{dIND}}_{\Pi_2,\mathcal{A}} \;\leq\; \mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(q_D) \;+\; q_E \cdot 2^{-(\kappa - (E+1)\,n\,\ell)},$$

by counting alone, without any leakage-resilience property of the sharing.

(c) If $\ell \geq \log_2 t + O(1)$, the local leakage resilience of the sharing fails for every n, t, and prime $p > 4t$ (for smaller p the budget already covers a whole share): a non-adaptive local leakage profile distinguishes, with certainty, a sharing of 0 from a sharing of $(p-1)/2$ [28, Thm. 4]. The hybrid argument behind (a) is therefore unavailable at that budget, and only (b) remains.

Two lessons follow. The $(E+1)\cdot 2^{-\Omega(n-k)}$ term is the price of rotation: each epoch's destruction of old shares is a fresh local-leakage event, and the bound grows with the number of rotations, which is exactly the accumulation a static-committee model hides. The *unit* of leakage also changes between custody models. A fraction ε of the key is the right budget for a single custodian (Theorem 1 degrades gracefully in ε), but for a committee the relevant quantity is bits per share, and the leakage-resilience route closes at a budget logarithmic in t regardless of how large the field is: the scale-and-round attack of Boudgoust and Simkin [28] maps each share into a ring of size $O(t)$ while preserving the separation of two secrets, and needs only $\log_2 t + O(1)$ bits from each. What it recovers is the scaled rounding of the secret, a constant number of bits of K, so (c) is the loss of a proof technique, not a decryption of c^*. Past that budget only the counting bound (b) is available, and it is weak: for 256-bit shares, a committee of seven under the software allowance $\ell \leq 8$ of Table 3 keeps $256 - 56(E+1)$ bits, so three rotations leave thirty-two and a fourth leaves nothing. Under attested zeroisation ($\ell = 0$) the leakage in case (a) is empty and its argument reduces to the perfect privacy of Shamir sharing; threshold custody implemented in software must either rotate rarely or justify a budget below one bit. Case (a) is epoch-static because the resilience theorems it rests on fix the corrupted set and the leakage functions before the shares exist; adaptive leakage and within-epoch adaptive corruption are open (OP2).

Theorem 3 (K3: delay is not destruction). *Let Π_3 be any scheme in which, from public information alone, the decryption key is computable by an algorithm of τ sequential steps (a time-lock puzzle [106], a VDF-gated release [24], or a beacon with a public schedule [82]). Then for every class $\mathcal{C}$ that contains a τ-step sequential algorithm there is an adversary with $\mathbf{Adv}^{\mathrm{dIND}}_{\Pi_3,\mathcal{A}} = \frac{1}{2} - \mathrm{negl}(\lambda)$, irrespective of ε and k.*

The theorem is immediate, and that is its point: a time-lock is Destruction-IND-CCA secure only for classes that cannot afford τ sequential steps, that is, for horizons shorter than the delay. It guarantees confidentiality *until* τ, which is a release policy, not an erasure policy. A time-lock can still play a legitimate role in erasure as a *retention-then-erase* trigger: the controller time-locks the destruction instruction rather than the key, so that custodians destroy at the end of a statutory retention period without the controller's further cooperation; the security of that pattern is that of the underlying K1 or K2 custody, and the time-lock adds only the commitment. The one time gate whose recipient can prove it has given up the ability to decrypt is Unruh's revocable timed-release encryption [114], and it needs quantum states to do so (OP1).

Call a custody scheme $\mathcal{T}$*-key-hiding* if no admissible adversary, given the oracles of Figure 1 other than Challenge and Dec, outputs the custodied secret with non-negligible probability. The proofs of Theorems 1 and 2(a) show that Π_1 and Π_2 are key-hiding under the same hypotheses (Appendix C).

Proposition 4 (K4, conditional: witness-gated custody reduces to custody of the witness)**.** *Let* Π_4 *encrypt* m *under a witness-encryption scheme* WE *for an* NP *language* L *with unique witnesses and a statement* $x \in L$ *whose witness* w *is the secret of a custody scheme* Π_{cust} *(K1 or K2) with* x *in its public parameters. If* WE *is extractable [64] and* Π_{cust} *is* $\mathcal{T}$*-key-hiding for* $\mathcal{T} = (\varepsilon, k, \mathcal{C}_\infty)$*, then* Π_4 *is Destruction-IND secure under* $\mathcal{T}$*, in the experiment of Figure 1 without the* Dec *oracle, against adversaries whose custody queries all precede* Challenge*, with* Destroy *called immediately after it. Plain (non-extractable) witness-encryption security gives no guarantee: it is only stated for* $x \notin L$*, and here* $x \in L$ *by construction.*

Four limits are deliberate. The statement is CPA-style, because no witness-encryption scheme we know of is stated with a decryption oracle; classical, because extractability is defined for classical distinguishers; restricted to custody queries made before the challenge, because the distinguisher must answer them without w; and it assumes extractability *with the custody view as auxiliary input*, a strong assumption (implausible for arbitrary auxiliary input [55]; only the fixed, efficiently samplable distribution of custody views is needed here). Of the K4 constructions we surveyed [14, 40, 56], none is stated with the extractability that erasure needs, because none was designed for it. Witness encryption does not destroy anything; it relocates the question to who holds w.

Together, Theorems 1–3 and Proposition 4 justify the shape of Axis 2 in §3: destruction happens in the custody layer (K1, K2), and the gates (time, witness) either weaken it to a delay or reduce to it.

5 GDPR Art. 17 Sufficiency

5.1 From regulatory text to cryptographic criterion

Article 17 of GDPR grants a right to erasure. Recital 66 clarifies the controller's obligation to "take reasonable steps" to propagate the erasure request. Neither the Article nor the Recital gives a cryptographic criterion. The EDPB Guidelines 02/2025 [45] are the first instrument to do so: they admit that a ciphertext may persist on a ledger, require the controller to document the threat model under which the data is considered unrecoverable, and ask that the data be "rendered unrecoverable" under it. We formalise the third requirement as a game between three parties, the controller, the data subject, and an attacker, and the whole criterion as the conjunction of three clauses.

The recovery experiment. Fix a custody scheme Π and a threat model $\mathcal{T} = (\varepsilon, k, \mathcal{C})$ declared by the controller. The data subject's datum m is drawn from a message ensemble $\{M_\lambda\}$, which may be any efficiently samplable distribution over $\{0,1\}^{L(\lambda)}$ (any distribution at all, if the reduction is allowed to be non-uniform); it is not chosen by the attacker and it need not have high entropy. The attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ receives *auxiliary information* $h(m)$, modelling everything it may know about m from outside the ledger, and then plays the experiment of Figure 1 with one change: Challenge takes no arguments and returns $c \leftarrow \mathsf{Enc}(pp, m)$, the ciphertext that is written to the ledger. After Destroy, $\mathcal{A}_2 \in \mathcal{C}$ holds the ledger, every residual state, and $\mathcal{A}_1$'s state, and outputs a value y. The attacker's *target* is a function $f(m)$: $f = \mathrm{id}$ is recovery of the datum, f a predicate is inference of an attribute. We write $\mathsf{Exp}^{\text{rec}}_{\Pi,\mathcal{A}}(\lambda; \mathcal{T}, M, h, f)$ for this experiment and say it *succeeds* when $y = f(m)$.

Definition 3 (EDPB "render unrecoverable" test, formalised)**.** A controller deploying Π satisfies the EDPB 02/2025 erasure criterion for a datum m under $\mathcal{T}$ from time t_0 if:

(1) **Ciphertext persistence admissibility.** The ciphertext c and the on-chain residue remain in every party's view after t_0. (This is not an obligation but a statement of the setting: the experiment gives c to the attacker by construction.)
(2) **Documented threat model.** $\mathcal{T}$ was declared by the controller before t_0.
(3) **No residual recovery channel.** For every efficiently samplable message ensemble $\{M_\lambda\}$, every polynomial-time computable h and f, and every attacker $\mathcal{A}$ admissible under $\mathcal{T}$, there exists a simulator $\mathcal{S}$ in the computational class of $\mathcal{A}_2$ (PPT when $\mathcal{C} = \mathcal{C}_\infty$) such that

$$\Pr\left[\mathsf{Exp}^{\text{rec}}_{\Pi,\mathcal{A}}(\lambda; \mathcal{T}, M, h, f) \text{ succeeds}\right] - \Pr\left[\mathcal{S}(1^\lambda, 1^{|m|}, h(m)) = f(m)\right] \;\le\; \mathrm{negl}(\lambda),$$

where t_0 is the time at which every honest custodian has completed Destroy.

Clause (3) is semantic security [65], in the auxiliary-input form of Goldreich [63, Def. 5.2.1], taken relative to the post-destruction view rather than to a single ciphertext. It says: whatever anyone within $\mathcal{T}$ can learn about m after t_0 while holding the ledger and the destruction residue, they could have learned without them. Three features matter for the regulatory reading: the datum is *sampled*, not chosen, so a one-bit attribute is as admissible as a passport scan and the clause asks only that the ledger not help the guess; outside knowledge is explicit as $h(m)$ and available to both sides, so it can never be the source of a "recovery"; and the target is any f, so the clause covers inference of attributes, which the EDPB's 2026 anonymisation guidance requires (§5.5).

5.2 The sufficiency theorem

Proposition 5 (Destruction-IND-CCA suffices for the formalised criterion)**.** *Let* Π *be a custody scheme and* $\mathcal{T}$ *a threat model declared by the controller before* t_0*. If* Π *is Destruction-IND-CCA secure under* $\mathcal{T}$ *(Def. 2), then the controller satisfies Def. 3 for every datum* m *under* $\mathcal{T}$ *from* t_0*. The converse is false.*

Proof sketch. Clauses (1) and (2) hold by the setting and the hypothesis. For clause (3), given an attacker $\mathcal{A}$, define the simulator $\mathcal{S}$ that runs its own instance of Π: it executes Setup, answers every oracle of Figure 1 honestly using the shares it generated, hands $\mathcal{A}$ the auxiliary input $h(m)$ it was given, and in place of the challenge returns $\mathsf{Enc}(pp, 0^{|m|})$; it outputs whatever $\mathcal{A}_2$ outputs. $\mathcal{S}$ never sees m beyond $h(m)$ and $|m|$. The real experiment and the simulation differ only in whether the challenge encrypts m or $0^{|m|}$; an attacker whose success probability differs non-negligibly between the two yields a Destruction-IND-CCA distinguisher that samples $m \leftarrow M_\lambda$, submits $(m, 0^{|m|})$, forwards all oracle queries, and guesses 0 when $\mathcal{A}_2$ outputs $f(m)$. This is the classical implication from indistinguishability to semantic security [63, Thm. 5.2.5]; the full argument, including the handling of the oracles and of $\mathcal{A}_2 \in \mathcal{C}$, is in Appendix B. For the converse, physical destruction of an air-gapped off-chain ciphertext store satisfies Def. 3 with no cryptographic property at all. □

The implication is the direction a controller needs: prove security, then document compliance. We do not claim, and the title does not say, that the two notions are equivalent; a regulatory criterion that admitted only cryptographic erasure would be wrong, and Def. 3 is deliberately satisfiable by other means.

5.3 What the theorem does and does not say

A controller who documents $\mathcal{T}$ and deploys a scheme with a proof of Destruction-IND-CCA under $\mathcal{T}$ has, under our reading of the EDPB criterion, a defensible claim of Art. 17 erasure. Three limits apply.

(1) It does not give the controller a free choice of $\mathcal{T}$. A controller who documents a leakage budget that covers the whole key satisfies the criterion vacuously, and a court will not find that defensible. The theorem is silent on which values are *regulator-defensible*; Appendix A tabulates per-industry defaults as a conjecture.
(2) It is a property of a scheme, not of a deployment. Implementation bugs, misconfigured HSMs, and social-engineered coercion are outside the model, as is refusal to erase.
(3) It does not discharge sectoral retention mandates. Art. 17 erasure and, say, SRA Accounts Rules or UK MLR 2017 retention are distinct regimes, and for the sectoral regime a destroyed-key ciphertext is no longer a readable record at all. This is a conflict of law, formalised as OP5 (§7.5).

Relation to deletion-compliance. Deletion-compliance [57, 61] (§2.3) requires the collector's post-deletion state to be indistinguishable from one in which the data was never shared, so any surviving artefact is a violation. Def. 3 targets the regime it assumes away: the ciphertext provably persists on a ledger nobody can scrub. What is a violation there is the premise here, and the guarantee moves from the collector's state to the threat model of Def. 1, for which deletion-compliance has no counterpart.

5.4 Consequences for controllers

Proposition 5 gives controllers a four-step procedure: pick a cell of Table 2; declare $\mathcal{T} = (\varepsilon, k/n, T)$ against Table 3; adopt a construction with a Destruction-IND-CCA proof under $\mathcal{T}$ (Theorem 1 for single custody, Theorem 2 for threshold custody, Proposition 4 for witness gates; Theorem 3 rules out delay-only designs); file the declaration with the erasure log under GDPR Art. 30. No deployed system known to us documents the third step with a proof; industry claims of "crypto-shredding for GDPR" are assertions. The reference implementation of §6 is the L2×K1 cell and falls under Theorem 1.

5.5 Beyond recovery: the on-chain residue under the 2026 anonymisation tests

Def. 3 answers one question: can anyone within $\mathcal{T}$ learn anything about m from what remains. The EDPB's 2026 draft guidelines on anonymisation [46] ask three. They adopt a *relative* notion of identifiability, assessed from the standpoint of each party who may obtain the data, and test it by whether that party can *single out* a person, *link* records about the same person, or *infer* information about them. Two observations connect this to the present framework.

First, the relative standpoint is exactly what the declared $\mathcal{T}$ provides: the party against whom the three tests are run is the attacker admissible under $\mathcal{T}$, and clause (3) of Def. 3 is the inference test for that party, with f ranging over all efficiently computable attributes.

Second, singling out and linkability are not about m; they are about the *residue* R that stays on the ledger after t_0: the ciphertext or commitment, its length, the transaction's timestamp and graph position, and the destruction certificate itself. Cryptographic erasure says nothing about R unless R is designed for it, and here the data-locus axis bites. For L1, an AE ciphertext is indistinguishable from random to any party within $\mathcal{T}$ and so cannot be linked by content; its length leaks $|m|$ and must be padded to a class. For L2, the anchor is linkable by anyone who holds m if it is a *plain hash of the plaintext*: such a party can recompute the hash and prove that this record concerned this datum, which is linkage in the guidelines' sense and survives key destruction entirely. An L2 anchor must therefore be a *hiding* commitment, a salted hash $H(r \,\|\, m)$ or the hash of a ciphertext under the destroyed key, with the opening r destroyed together with the key. The CNIL reached the same ranking in 2018 [39], placing a commitment above a keyed hash above a ciphertext and noting that deleting the opening or the key moves the controller closer to erasure; the rule below says what that ranking requires. We record it as a design rule.

> *Rule (hiding anchors).* An anchor-only (L2) architecture whose on-chain commitment is a deterministic function of the plaintext does not satisfy the linkability test for any party who may later obtain the plaintext, regardless of its custody model. Anchors must be hiding commitments whose openings are destroyed with the key.

Timestamps and transaction-graph position are outside cryptographic erasure altogether and must be assessed by the controller under the relative standard like any other metadata. The destruction certificate must not embed an identifier of the data subject. Criterion C10 in §6 scores each architecture on the inference test via Def. 3 and on the residue tests via the rule above.

6 Evaluation of Representative Architectures

We now evaluate seven reference architectures drawn from the grid of §3 against eleven criteria spanning engineering, cryptographic, and regulatory dimensions. Table 4 says what each architecture is (a published construction, a deployed system, or a deployment pattern) and what it assumes; the list below describes each. The reference implementation (**A3**, the L2×K1 anchor-only system) is measured end to end and the cryptographic cost of A4 and A5 with public implementations (Appendix D); the rest is estimated from published numbers (Appendix D.1). Table 5 is the summary; the prose below explains the scoring.

6.1 Systems evaluated

A0: plaintext on-chain. Personal data written to the ledger unencrypted, a negative control: with no key there is nothing to destroy.

A1: hash-only anchor. A SHA-256 hash of the record on the ledger, the record off-chain in plaintext, "erasure" its deletion: the commonest enterprise pattern. It sits in row L2 with no custody model, and its plain hash is linkable by anyone who later holds the record (§5.5).

A2: encrypted anchor, HSM key. The L2×K1 pattern as usually described: ciphertext on IPFS, a Merkle root on the chain, one key in an HSM, destruction by the HSM's zeroisation primitive with a vendor attestation.

A3: encrypted anchor, software-protected key (reference implementation). The same cell, instantiated by the production system of §6.5. Document records and AI interactions are each encrypted under their own AES-256-GCM data-encryption key, wrapped under a software-protected master key with explicit zeroisation; ledger events have been encrypted under a key held per data subject since August 2026, and before that their payloads were hash-chained in the clear. Events are hash-chained and the chain's Merkle roots are submitted to OpenTimestamps calendar servers; each block's answer is stored as a standard `.ots` file and upgraded in the background to a Bitcoin block-header proof once the calendar's transaction confirms. Erasure zeroises the wrapped key and its nonce in place and writes a destruction certificate; the ciphertext stays, and so, for up to ten minutes, does the wrapped key in the retained database dumps (Table 3; limitations below). The system is run by its developer as audit-trail infrastructure for SRA-regulated UK legal practices, integrated with a commercial practice-management platform.

A4: threshold custody on-chain. L1×K2: a committee hosted on the chain itself holds Shamir shares of the key, in the manner of Benhamouda et al. [23] and YOSO [36]; destruction is a quorum refusing to reconstruct and then erasing its shares. It is a published construction, not a deployed system.

A5: time-locked release. L1×K3: ciphertext on the chain whose key is released by a time-lock puzzle [106] or by a beacon such as drand's tlock [24, 82]. Puzzles are constructions; drand is deployed.

A6: witness-gated release. L1×K4 in its deployable corner: Derler–Slamanig witness encryption for algebraic languages [40], where plaintext is readable only with a witness and "destruction" is destroying the one witness. By Proposition 4 its erasure security is that of the custody of the witness.

6.2 Criteria

Eleven criteria, grouped into three blocks:

Engineering (lower = better where applicable): (**C1**) steady-state throughput, events/s; (**C2**) destruction-acknowledgement latency, p50; (**C3**) on-chain cost of the destruction operation, gas units; (**C4**) L1 deployability on Bitcoin/Ethereum mainnet without a hard fork.

Cryptographic (higher = better): (**C5**) forward secrecy against pre-destroy key leak; (**C6**) quorum tolerance, expressed as the maximum k/n of custodians that can be adversarial before erasure fails; (**C7**) post-quantum robustness under Grover on the key space and Shor on the ciphertext's anchor signatures; (**C8**) the leakage budget ℓ, in bits per share per epoch, that the destruction procedure can justify declaring (Table 3).

Regulatory & audit (higher = better): (**C9**) third-party destruction auditability without the custodian's cooperation, the goal of proofs of secure erasure [75, 97] and certified deletion [17, 30]; (**C10**) legal defensibility under the EDPB 02/2025 criterion (§5.1); (**C11**) composability with ZKP-based selective disclosure in the same session.

6.3 Summary table

Table 5 scores the seven architectures. The A3 row is measured: `criterion-rs` on the production Rust core (Apple M2) and on the production cloud VPS (PostgreSQL 16), with hardware, commit hashes, and raw outputs in Appendix D. The cryptographic costs of A4 and A5 are measured on the same machine with public implementations (Appendix D); the remaining cells are order-of-magnitude estimates from published numbers (CHURP and YOSO for A4's protocol, Derler–Slamanig for A6, Ethereum gas costs for A1 and A2), not benchmarks of ours. The qualitative entries are our judgement against Def. 3 and the residue rule (C10), Def. 1 (C5–C8), and deployability (C4, C11); Appendix D.1 gives the source behind every estimated cell.

6.4 Walk-through

Three observations organise Table 5.

Observation 1: only A4 fully satisfies all three regulatory criteria; A2 and A3 do so subject to L2 availability and trust in a third party. A0 fails trivially. A1 relies entirely on the off-chain controller honouring the delete, and its plain hash anchor fails the linkability rule of §5.5. A5 fails C10 by Theorem 3. A6 has weak C9, since verifying that a witness has been destroyed is proving a non-existence statement. A2 and A3 inherit the L2 availability failure mode, and their C9 rests on a vendor's or operator's certificate where A4 has on-chain quorum-refusal proofs. Bitcoin anchoring leaves A3's C9 unchanged: it proves a block existed, not that a key was destroyed.

Sys.	Type	Cell	Hardness assumption	Parties / honest	Destroy
A1	pattern (deployed)	L2	collision resistance of H	1 / —	local delete
A2	pattern	L2×K1	AE (ideal cipher), HSM zeroisation	1 / —	local, attested
A3	deployed system	L2×K1	AE (ideal cipher), software zeroisation	1 / —	local, software
A4	papers [23, 36]	L1×K2	DL / pairing for DKG; Shamir privacy	$n, t \le n$ / t honest	interactive (quorum)
A5	papers + drand (deployed)	L1×K3	sequential hardness (RSW, VDF) or beacon honesty	1 or beacon n / majority	none (release)
A6	paper [40]	L1×K4	pairing (Groth–Sahai), extractability (Prop. 4)	witness custody	as its custody

Table 4: What each evaluated architecture other than the plaintext baseline A0 is and assumes. "Type" says whether the row is a published construction, a deployed system, or a deployment pattern; "Parties / honest" gives the minimum number of custodians and the honesty requirement; "Destroy" says whether destruction is a local operation or an interactive protocol.

	Engineering				Cryptographic				Reg.		
System	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11
A0 plaintext	$\gg 10^5$	n/a	0	yes	n/a	n/a	✗	✗	n/a	✗	part.
A1 hash-only anchor	$\sim 10^5$	$<$1s	21K	yes	n/a	n/a	part.	n/a	weak	✗	yes
A2 L2×K1 HSM	$\sim 10^4$	$<$1s	21K	yes	part.	1/1	part.	0 (att.)	vendor	yes	yes
A3 (ref.) L2×K1	$\sim 2.6{\cdot}10^3$ (m.)[⋆]	1.23 ms (m.)	free (cal.)	yes	part.	1/1	p.$T{\approx}15$y	$\lvert sk\rvert \le 10$ min, then sw.	operator	after 10 min	yes
A4 L1×K2 (t, n)	$\sim 5{\cdot}10^2$ (m.)[b]	5–30s[c]	$\sim 10^5$	yes	part.	$<t/n$	part.	0 (att.)	on-chain	yes	yes
A5 L1×K3 tlock	$\sim 5{\cdot}10^2$ (m.)	τ	n/a	yes	part.	n/a	part.	n/a	beacon	✗	yes
A6 L1×K4 WE alg.	$\sim 10^2$	as cust.	21K[a]	part.	part.	n/a	✗ (pair.)	as cust.	weak	part.	weak

Table 5: Seven reference architectures against eleven criteria. Shaded row is the production reference implementation; "(m.)" indicates measured via `criterion-rs` benchmarks (§6.5, Appendix D). [⋆] A3 C1 is single-threaded; 10^4 events/s is a projection for a 5-connection pool, not a measurement. A4 and A5 C1 are per-core encryption rates measured with public implementations (Appendix D); [b] a threshold decryption costs 0.35–0.54 ms unverified, 12–20 ms with t verified shares; [c] network-bound, one decryption-share round under 25 ms. Units: C1 events/s; C2 p50 latency; C3 gas to record destruction on Ethereum ([a]one attestation transaction); "as cust." = inherited from the custody of the witness (Prop. 4); τ = the configured delay, after which A5 is readable (Thm. 3). "part." = partial; "vendor" = HSM vendor attestation, "operator" = the operator's own destruction certificate; neither is third-party verifiable; "pair." = pairing-based, vulnerable to Shor on the underlying pairing-friendly group. C8 is the leakage budget ℓ of Table 3: "att." = attested zeroisation taken at face value, "sw." = the software allowance; A3's entry is its backup window of up to ten minutes (§6.5).

Its C10 is "after 10 min": while a dump holds the key the declared budget is the whole key, which §5.3 rules out.

Observation 2: throughput gap is three orders of magnitude. A3 measures 2.6×10^3 events/s single-threaded on a small cloud VPS (10^4 projected with a connection pool); A4 and A5 are held to about $5 \cdot 10^2$ by the pairings in each operation, and A4 to below 10^2 once decryption shares are verified; A6 to about 10^2 by pairings. For audit trails that record every transaction, only the single-custodian designs currently combine production throughput with a destruction argument (Theorem 1), and they pay for it in C9.

Observation 3: post-quantum exposure is uniform at $T \approx 15$. Every row with a key to destroy reads "partial" under C7 (Grover on AES-256-GCM, Shor on secp256k1 anchors), and A6's pairing-based gate fails outright; none solves OP1.

6.5 Reference implementation availability

The A3 reference implementation is a production engine deployed by an SRA-regulated UK legal-sector RegTech provider. SHA-256 Merkle roots over hash-chained event blocks go to a public OpenTimestamps calendar [113]; a background job upgrades each `.ots` answer to a Bitcoin block-header proof once the calendar's transaction confirms. All 24 blocks sealed from 14 May to 10 August 2026 are confirmed (Bitcoin blocks 949,407 to 961,868), the files are byte-identical to the reference client's `ots upgrade` output, and anyone can check one with `ots verify` against a Bitcoin node.

Benchmarks. Per-event cryptographic overhead (DEK generation, AES-256-GCM encryption of 256 B, DEK wrap) measured with `criterion-rs` on the Rust core is $7.02 \pm 0.01\,\mu$s; on the production VPS an event append costs $390 \pm 10\,\mu$s ($\sim$ 2,560 events/s single-threaded, cryptography 1.8% of the path) and a cryptographic erasure 1.23 ± 0.03 ms ($\sim$ 810 destructions/s). With a 5-connection pool the projected throughput is $10\text{–}12 \times 10^3$ events/s. The measurements are of the April 2026 code; the encryption figures are for the document-record path.

The cryptographic core, the `criterion-rs` harness, and raw benchmark outputs will be released with the peer-reviewed version, without the proprietary practice-management connectors, and are available from the author on request.

Limitations of the A3 reference. Four caveats qualify the shaded row of Table 5. A3 inherits the limits of single custody: its C9 depends on the operator's own certificate (after an HSM migration, on the HSM vendor), and A4 genuinely outperforms it there. The current deployment uses a software-protected master key and keeps backups, so its budget is the backup window of Table 3 rather than the attested zero of the FIPS-L4 target. Document records anchor a plain hash of the document, and ledger events written before the August 2026 migration hash-chained their payloads in the clear; both must move to hiding commitments under the rule of §5.5. Destroyed keys persist in the daily dumps, under an unrotated master key, until the ten-minute job rewrites them (for 14

days before October 2026): the residual state ρ of Figure 1, declared in Table 3. A3 is the authors' own system; its entries should be read with that in mind.

7 Open Problems

7.1 OP1: Post-quantum destruction

Every custody row of Table 5 reads "partial" under C7: AES-256-GCM loses half its effective key length to Grover, and the secp256k1 signatures that anchor to Bitcoin and Ethereum fall to Shor. A controller who declares $T = \infty$ today is claiming more than the deployed primitives deliver. The question: *is there a deployable scheme that is Destruction-IND-CCA secure against $\mathcal{C}_{\mathrm{Q}}$?* Candidate routes: (i) $\geq$ 384-bit symmetric keys; (ii) ML-KEM [94] hybrid encryption; (iii) post-quantum anchor signatures [8, 121], so that Shor does not enable retroactive forgery; (iv) LWE-based updatable encryption [6], with destruction as the absence of further updates; (v) lattice time-lock puzzles [3] for K3, which make the delay post-quantum but, by Theorem 3, still not an erasure [47]. No public-ledger deployment executes any of these.

7.2 OP2: The leakage threshold

Define the *leakage threshold* of a custody scheme with design parameter k as $\varepsilon^*(\Pi, \mathcal{C}) = \sup\{\varepsilon : \Pi \text{ is secure under } (\varepsilon, k, \mathcal{C})\}$. Theorem 1 determines it for single custody in the ideal-cipher model and a single epoch: $\varepsilon^* = 1 - \omega(\log \lambda)/\kappa$, divided by $E + 1$ when the key is refreshed in place. For Shamir custody Theorem 2 gives one bit per share (secure, non-adaptively, for threshold ratios above 0.69–0.78), shows that beyond $\log_2 t + O(1)$ bits the leakage-resilience argument is lost and only the counting bound of Theorem 2(b) remains, and leaves three questions open: whether Destruction-IND-CCA in fact fails between those budgets (the known attack recovers $O(1)$ bits of the key, not the key), the exact resilience threshold in between [87], and whether *adaptively* chosen one-bit local leakage is tolerated, which the Figure 1 lifetime phase permits but no cited result covers. Beyond Shamir: non-linear reconstruction escapes the attack of [28], and puncturable primitives [18, 83, 116] come with definitions of the same destruction form, which may carry over.

7.3 OP3: Decentralised key-lifecycle audit

Criterion C9 shows that only K2 architectures admit on-chain destruction audit without the custodian's cooperation; K1 relies on vendor attestation, K3 on beacon verifiability, and K4 on proving a non-existence statement. Three lines of work approach this from different ends: proofs of secure erasure [75, 97], where a bounded-memory device proves it has overwritten its state; publicly verifiable deletion [70] of a discrete-logarithm key; and certified deletion [17, 30], where the holder of a *ciphertext* proves, using quantum states, that it can no longer decrypt. None covers an HSM or a committee destroying a share on which an on-chain ciphertext depends. The open problem: *a destruction-audit protocol sound without a trusted third party*, for any custody model. Traceable threshold encryption [26] advances the dealer side; explainable deletion [102] fixes what the trail must say to a non-expert data subject.

Multi-custody extension. Encrypted mempools [5, 37, 51, 77] add a second ciphertext of the same data, the transaction intent, whose keys are held by a different party set than the application-layer keys. Def. 1 extends by giving each layer its own (ε, k); auditing the intent, application, and block-producer keys as one lifecycle is a sub-problem of OP3.

7.4 OP4: Composing with selective disclosure

Identity deployments pair encrypted credentials with zero-knowledge selective disclosure ("over 18" without the date of birth); C11 asks whether erasure composes with this. A proof about m stays verifiable after the key is destroyed and, as part of the residue of §5.5, preserves the predicate it proved, which may be what erasure was meant to remove. The open problem: define *selective-destruction semantics* under which the residue of a disclosure passes the inference test for the erased attributes.

7.5 OP5: Multi-regime retention conflict

The sufficiency theorem of §5 is silent on personal data embedded in a record that a *sectoral* mandate compels the controller to retain in readable form (SRA Accounts Rules, UK MLR 2017). Destroying the key satisfies Def. 3, but for the sectoral regime a destroyed-key ciphertext is no longer a record at all. This is a conflict of law, and it leaves two sub-problems. **(a)** Erase the personal fields of a record while leaving a readable, sectorally valid skeleton; puncturable encryption [68], which revokes one ciphertext while keeping the key, is the natural primitive. **(b)** Log each record's retention basis and Art. 17(3) exemption verifiably during the retention period and erase cryptographically when it ends; the erasure step is a Destruction-IND-CCA question, the log's legal effect one for lawyers.

7.6 Limitations of this SoK

First, the evaluation measures one system end to end, ours, and only the cryptographic cost of two others; the rest of Table 5 is estimates, not head-to-head measurements. Second, the formal results are stated under explicit restrictions: Theorem 1 is in the ideal-cipher model, Theorem 2(a) covers epoch-static adversaries and one-bit leakage, and the gate results are an impossibility and a conditional reduction. Third, our reading of the EDPB criterion (Def. 3) is endorsed by no authority, and the 2026 anonymisation guidance is a draft.

8 Conclusion

This paper systematizes the design space in which Article 17 lets key destruction count as erasure on a ledger one cannot rewrite: a taxonomy of locus, custody, and gate (§3), a phased adversary model with a result per custody model (§4), a sufficiency theorem for the EDPB criterion (§5), and a residue rule for the 2026 anonymisation tests (§5.5).

9 Ethical Considerations

The systematization and the formal model involve no human subjects and no personal data. The reference implementation evaluated in §6 runs in production, but the measurements reported here use synthetic compliance events and aggregated

throughput/latency only; no record-level personal data appears in any artifact accompanying this paper. The work introduces no new attack and surfaces no offensive-security or dual-use capability beyond what is already public in the cited literature.

10 Open Science

This is a systematization paper; its primary artefacts are the taxonomy, the adversary model, the proofs, and the evaluation, all contained in the paper and its appendices. An artifact (SHA-256 `204c5612cf14a69e`) contains the cryptographic core of the reference implementation (envelope encryption, key management, per-subject ledger encryption, Merkle anchoring, OpenTimestamps encoding), the `criterion-rs` benchmark harness with the raw outputs behind Appendix D, the 24 Bitcoin-confirmed `.ots` proofs of §6.5 with a verification script, and the schema needed to run the database-inclusive benchmarks; it is available from the author and will be published with the peer-reviewed version. The application layer and all personal data are excluded. Every estimated cell of Table 5 is traceable to a published number or a stated calculation in Appendix D.1.

11 AI Use

A generative-AI assistant was used for drafting assistance, LaTeX formatting, and literature-search support. All technical content, including the taxonomy, the formal model and its definitions, the Art. 17 sufficiency argument and its proofs, and the evaluation, is the authors' own intellectual contribution; the authors take full responsibility for the accuracy, originality, and integrity of the paper and its artefacts. All references were checked against primary sources.

A Per-industry defensible $\mathcal{T}$ defaults

Proposition 5 is silent on what values of the controller's threat model $\mathcal{T} = (\varepsilon^*, t^*/n^*, T^*)$ a regulator will find defensible. The EDPB 02/2025 Guidelines require the threat model to be documented but do not prescribe values. Table 6 offers our *conjecture* of per-industry defaults. These are not regulatory guidance; they are a conjecture intended to be refined by case law and subsequent regulatory instruments.

Industry	ℓ^* (bits)	t^*/n^*	T^* (yrs)
Payments / KYC	0 (attested)	3/5 or HSM	15 (PQ-aware)
Healthcare (patient record)	0 (attested)	5/7	25 (lifetime)
Public-sector audit logs	≤ 8 (software)	3/5	10 (PQ-migr.)
Enterprise RegTech audit	0 (attested)	HSM L4	15
Identity / verifiable cred.	0 (attested)	2/3	lifetime-of-iss.

Table 6: Conjectural per-industry defensible defaults for $\mathcal{T}$. Not legal advice; a framing hypothesis.

The conjecture reflects three regularities. First, T^* scales with the longevity of the data subject's interest in the record: healthcare and identity exceed a working lifetime; payments records tend to follow the PCI-DSS 7-year baseline. Second, the reconstruction threshold t^*/n^* is raised when the data subject has no ongoing relationship with the custodian, because the custodian's incentive to refuse reconstruction weakens over time. Third, ℓ^* is zero, that is, attested zeroisation is required, wherever plaintexts are high-value and concentrated (payments, healthcare, identity); a software allowance is tolerable only for audit logs whose personal content is incidental.

B Proof of Proposition 5

We prove clause (3) of Def. 3 from Destruction-IND-CCA security. Fix Π, $\mathcal{T} = (\varepsilon, k, \mathcal{C})$, a message ensemble $\{M_\lambda\}$ over $\{0,1\}^{\ell(\lambda)}$, polynomial-time computable h and f, and an attacker $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ admissible under $\mathcal{T}$ in $\mathsf{Exp}^{\mathrm{rec}}$.

The simulator. $\mathcal{S}(1^\lambda, 1^L, z)$ with $z = h(m)$ runs $(pp, \mathsf{sk}_{1..n}^{(0)}) \leftarrow \mathsf{Setup}(1^\lambda, n, t)$ and executes the experiment of Figure 1 as challenger for $\mathcal{A}$, with two changes: it gives $\mathcal{A}_1$ the input (pp, z), and on the (argument-less) Challenge call it returns $\tilde{c} \leftarrow \mathsf{Enc}(pp, 0^\ell)$. All oracles (Corrupt, Leak, Dec, Profile, Refresh, Destroy, Residue) are answered honestly from the shares $\mathcal{S}$ generated; this is possible because $\mathcal{S}$ is the challenger of its own instance and the oracles depend on the shares, the profile, and the ciphertexts, none of which involve m. When $\mathcal{A}_2$ halts with y, $\mathcal{S}$ outputs y. $\mathcal{S}$ runs $\mathcal{A}_1$ and $\mathcal{A}_2$ as subroutines, so it lies in the class of $\mathcal{A}_2$ (PPT, or quantum polynomial time when $\mathcal{C} = \mathcal{C}_{\mathrm{Q}}$), which is what Def. 3 requires. We denote

$$p_{\mathrm{real}} = \Pr\left[\mathsf{Exp}^{\mathrm{rec}}_{\Pi,\mathcal{A}} \text{ succeeds}\right],$$
$$p_{\mathrm{sim}} = \Pr\left[\mathcal{S}(1^\lambda, 1^L, h(m)) = f(m)\right],$$

both over $m \leftarrow M_\lambda$ and all coins.

The distinguisher. Suppose $p_{\mathrm{real}} - p_{\mathrm{sim}} \geq \eta(\lambda)$ for infinitely many λ and some non-negligible η. Build $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ against $\mathsf{Exp}^{\mathrm{dIND}}_{\Pi,\mathcal{B}}(\lambda; \mathcal{T})$ as follows. $\mathcal{B}_1$ receives pp, samples $m \leftarrow M_\lambda$ (this is where efficient samplability is used; for an arbitrary ensemble the same argument goes through with a non-uniform $\mathcal{B}$ that has a sample hard-wired, exactly as in [63, Thm. 5.2.5]), computes $z = h(m)$, and runs $\mathcal{A}_1(pp, z)$. Every oracle query of $\mathcal{A}_1$ is forwarded to $\mathcal{B}_1$'s own oracle of the same name and the answer returned; the preconditions of Figure 1 are identical in the two experiments, so $\mathcal{B}_1$ is admissible whenever $\mathcal{A}_1$ is. When $\mathcal{A}_1$ calls Challenge(), $\mathcal{B}_1$ calls Challenge$(m, 0^\ell)$ and returns the answer c^*. When $\mathcal{A}_1$ halts with state st, $\mathcal{B}_1$ halts with state (st, m). $\mathcal{B}_2$ runs $\mathcal{A}_2(st)$, forwarding Residue queries, and outputs $b' = 0$ if $\mathcal{A}_2$'s output equals $f(m)$, else $b' = 1$. $\mathcal{B}_2$ lies in $\mathcal{C}$ because $\mathcal{A}_2$ does and the wrapper is polynomial.

Analysis. Condition on the challenge bit. If $b = 0$, then $c^* = \mathsf{Enc}(pp, m)$ and $\mathcal{A}$'s view is distributed exactly as in $\mathsf{Exp}^{\mathrm{rec}}$, so $\Pr[b' = 0 \mid b = 0] = p_{\mathrm{real}}$. If $b = 1$, then $c^* = \mathsf{Enc}(pp, 0^\ell)$ and $\mathcal{A}$'s view is distributed exactly as under $\mathcal{S}$ (the shares were generated by the challenger rather than by $\mathcal{S}$, but with the same algorithm and independently of m, so the joint distribution of view and m is identical), hence $\Pr[b' = 0 \mid b = 1] = p_{\mathrm{sim}}$. Therefore

$$\begin{aligned}\Pr[b' = b] &= \tfrac{1}{2}\, p_{\mathrm{real}} + \tfrac{1}{2}\,(1 - p_{\mathrm{sim}}) \\ &= \tfrac{1}{2} + \tfrac{1}{2}\,(p_{\mathrm{real}} - p_{\mathrm{sim}}) \;\geq\; \tfrac{1}{2} + \tfrac{1}{2}\,\eta(\lambda),\end{aligned}$$

contradicting Destruction-IND-CCA security under $\mathcal{T}$. So $p_{\mathrm{real}} - p_{\mathrm{sim}} \leq \mathrm{negl}(\lambda)$, which is clause (3). □

What the proof uses. Only two facts: that $\mathcal{S}$ can run the whole experiment itself, which holds because every oracle is a function of objects the challenger generates, and that $\mathcal{B}$ can forward $\mathcal{A}$'s queries, which holds because the two experiments have the same oracle interface. Outside knowledge $h(m)$ never enters the reduction because both $\mathcal{A}$ and $\mathcal{S}$ receive it; this is why the criterion is stated relative to a simulator rather than as an absolute bound on $\Pr[y = m]$.

Why the converse fails. Def. 3 is satisfied by any process that leaves nothing in the attacker's view that depends on m, cryptographic or not; physical destruction of an off-chain store is one example, and a scheme that is semantically secure only for a particular ensemble $\{M_\lambda\}$ while failing Def. 2 for adversarially chosen messages is another. Destruction-IND-CCA is sufficient, not necessary.

C Proofs for §4.7

A lemma on leaked keys. For a random variable K and leakage Z with $|Z| \le \beta$ bits, the average min-entropy satisfies $\tilde{H}_\infty(K \mid Z) \ge H_\infty(K) - \beta$ [42, Lemma 2.2]. For K uniform on $\{0,1\}^\kappa$ and $\beta = \varepsilon\kappa$ this gives $\tilde{H}_\infty(K \mid Z) \ge (1-\varepsilon)\kappa$, and so for any algorithm making q queries to an oracle that is answered independently of K unless the query names K, the probability that some query names K is at most $q \cdot 2^{-(1-\varepsilon)\kappa}$.

Proof of Theorem 1. Let $\mathcal{A}$ be an adversary against Π_1 under $(\varepsilon, 0, \mathcal{C}_\infty)$ that triggers E epochs. With $n = 1$ and $k = 0$ there are no corruptions, and Refresh leaves K in place, so every epoch's budget is spent on the same key: the leakage $\mathcal{A}$ obtains about K is $Z = (Z_{\mathrm{pre}}, (L^{(e)}(K), \rho^{(e)})_{e \le E})$, with at most $\varepsilon\kappa$ bits per epoch by the budget of Figure 1 and $|Z| \le (E+1)\varepsilon\kappa$ in all, where Z_{pre} is the adaptive pre-challenge leakage. Consider the event Hit that some ideal-cipher query of $\mathcal{A}_1$ or $\mathcal{A}_2$ is made at key K. Conditioned on $\neg$Hit, the permutation E_K is independent of every oracle answer $\mathcal{A}$ has seen, so AE_K is instantiated with a permutation the adversary has never queried; the remaining experiment, in which $\mathcal{A}$ has a decryption oracle on $c \ne c^*$ and must distinguish $\mathsf{Enc}_K(m_0)$ from $\mathsf{Enc}_K(m_1)$, is the AE experiment for AE with a random permutation, and its advantage is at most $\mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(q_D)$ [107]. The adaptive pre-challenge leakage does not help here: it is a function of K and so of which permutation is in use, but conditioned on $\neg$Hit the adversary cannot evaluate that permutation anywhere, and the AE bound holds for every fixed K. Finally, $\Pr[\mathsf{Hit}] \le q_E \cdot 2^{-(1-(E+1)\varepsilon)\kappa}$ by the lemma with $\beta = (E+1)\varepsilon\kappa$, because queries at keys other than K are answered by permutations independent of K. The hypothesis $(E+1)\varepsilon < 1$ is necessary: an adversary that leaks a fresh $\varepsilon\kappa$-bit segment of K in each of $\lceil 1/\varepsilon \rceil$ epochs holds K within budget.

For $\mathcal{A}_2 \in \mathcal{C}_{\mathrm{Q}}$ with quantum ideal-cipher access, conditioning on Hit is replaced by the one-way-to-hiding lemma of Ambainis, Hamburg and Unruh [11]. Write $h = (1-(E+1)\varepsilon)\kappa$. Let Find be the event that the semi-classical oracle, which measures whether a query is at key K, fires. The lemma bounds the advantage of distinguishing E from E reprogrammed at K by $2\sqrt{(q_E+1)\Pr[\mathsf{Find}]}$, and its search theorem, which makes no independence assumption, gives $\Pr[\mathsf{Find}] \le 4q_E \cdot \Pr[\mathcal{B} = K]$ for the algorithm $\mathcal{B}$ that runs $\mathcal{A}$ with the oracle at keys other than K up to a uniformly chosen query and measures it. $\mathcal{B}$'s input is the leakage Z together with the cipher tapes at the other keys, which are independent of K; the pre-challenge leakage may be adaptive in those tapes, which is why we do not condition on the value of Z alone. By the chain rule for average min-entropy (the lemma on leaked keys, applied given the tapes), $\Pr[\mathcal{B} = K] \le 2^{-\tilde{H}_\infty(K|Z,\mathrm{tapes})} \le 2^{-(\kappa-|Z|)} \le 2^{-h}$, so the advantage is at most $2\sqrt{4q_E(q_E+1)2^{-h}} \le 4(q_E+1)\,2^{-h/2}$, which is the stated $O(q_E \cdot 2^{-h/2})$. The square root relative to the classical term is the usual price of the one-way-to-hiding technique; the query threshold it gives, $2^{h/2}$, is the one quoted after the theorem. □

Proof of Theorem 2. Fix an epoch-static adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ under $(\varepsilon, k, \mathcal{C}_\infty)$ that triggers epochs $0, \dots, E$, with corrupted sets $C_0, \dots, C_E$, $|C_e| \le k < t$, and a profile $(L_i)_{i \in [n]}$ of one-bit leakage functions. Evaluation points $\alpha_1, \dots, \alpha_n \in \mathbb{F}_p^*$ are distinct. Write $f^{(e)}$ for the degree-$(t-1)$ sharing polynomial of epoch e, so $\mathsf{sk}_i^{(e)} = f^{(e)}(\alpha_i)$ and $f^{(e)}(0) = K$.

Step 0: what the view contains. Four kinds of object depend on K: (i) the corrupted shares $\{\mathsf{sk}_j^{(e)}\}_{j \in C_e}$ for each e; (ii) the corrupted parties' views of each Refresh; (iii) the leakage $L_i(\mathsf{sk}_i^{(e)})$ and residual state $\rho_i^{(e)}$ of each honest share at the destruction of epoch e, together at most one bit per share by admissibility (a randomised Destroy is a convex combination of deterministic ones, and the bounds below are convex); (iv) the challenge ciphertext and the answers of Dec, all computed with the ideal cipher at key K. Leakage functions have no access to E (Def. 1).

Step 1: the corrupted parties learn nothing about K from (i) and (ii). Within one epoch, $|C_e| < t$ evaluations of a uniform degree-$(t-1)$ polynomial with fixed constant term are uniform in $\mathbb{F}_p^{|C_e|}$ and independent of K. Refresh adds $g^{(e)} = \sum_j g_j^{(e)}$, each party P_j contributing a uniform polynomial of degree $< t$ with $g_j^{(e)}(0) = 0$ over private channels; a corrupted party's view of the protocol is its own $g_j^{(e)}$ and the values $g_{j'}^{(e)}(\alpha_i)$, $i \in C_e \setminus R_e$, sent by honest j' to the corrupted parties that take part (released parties are rebooted and receive nothing), and for a uniform polynomial of degree $< t$ with zero constant term the values at $m \le t-1$ distinct non-zero points are uniform in $\mathbb{F}_p^m$. Hence the joint distribution of all corrupted shares and all refresh views, across all epochs, is independent of K [72], and it can be sampled by a simulator that knows neither K nor the honest shares. Moreover, given K and the corrupted information of epoch e, $f^{(e)}$ is uniform on a coset of $V_e = \{h : \deg h < t,\ h(0) = 0,\ h|_{C_e} = 0\}$, while $g^{(e)}$ is independent of $f^{(e)}$ and uniform on a coset of $V_e' = \{h : \deg h < t,\ h(0) = 0,\ h|_{C_e \setminus R_e} = 0\} \supseteq V_e$ (one honest contribution $g_{j'}^{(e)}$ already is, given the values it sent to $C_e \setminus R_e$). Hence $f^{(e+1)} = f^{(e)} + g^{(e)}$ is uniform on a coset of V_e' and independent of $f^{(e)}$, and the epoch sharings are mutually independent given K and the corrupted parties' information.

Step 2: detaching the cipher. Let Hit be the event that some ideal-cipher query of $\mathcal{A}_1$ or $\mathcal{A}_2$ is made at key K. Let G^* be the experiment in which the challenger computes the challenge and all Dec

answers with an independent uniformly random permutation P in place of E_K. By lazy sampling, the real experiment and G^* are identically distributed until Hit, so

$$\left|\Pr[\mathrm{Exp}^{\mathrm{dIND}} = 1] - \Pr[\mathrm{G}^* = 1]\right| \;\le\; \Pr[\mathrm{Hit}],$$

and $\Pr[\mathrm{Hit}]$ is the same in both. In G^* the objects of kind (iv) are functions of P, which is independent of K and of the sharings, so in G^* the only dependence of the view on K is through (iii), and

$$\left|\Pr[\mathrm{G}^* = 1] - \tfrac{1}{2}\right| \;\le\; \mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(q_D),$$

because an adversary in G^* is an AE adversary against the mode instantiated with the random permutation P, given auxiliary information (the custody view) that is independent of P.

Step 3: hybrids over epochs. For $0 \le e \le E+1$ let G^*_e be G^* modified so that, for every epoch $e' < e$, the leakage of kind (iii) is computed not on the real honest shares but on the honest shares of an independent sharing of 0 that agrees with the corrupted shares of epoch e' (such a sharing exists and is uniform given the corrupted shares, since $|C_{e'}| < t$). $\mathrm{G}^*_0 = \mathrm{G}^*$. In G^*_{E+1} nothing in the view depends on K: (i) and (ii) by Step 1, (iii) by construction, (iv) by Step 2.

Claim. For each $e \le E$, the joint distribution of (K, view) in G^*_e and in G^*_{e+1} are within statistical distance $\delta := 2^{-(3\log_2\pi-2)(\gamma-c)(n-k)}$.

Proof of the claim. The two experiments differ only in the epoch-e leakage. Condition on everything the adversary has seen before the destruction of epoch e and on K. Because $\mathcal{A}$ is epoch-static, C_e is a function of that history, which in G^*_e is independent of $f^{(e)}$ given K and the corrupted shares (Step 1 and the hybrid construction for $e' < e$). Thus, conditioned on the history, epoch e is a (t,n) Shamir sharing of K in G^*_e and of 0 in G^*_{e+1}, with a fixed corrupted set $\Theta = C_e$ of size $\le k$ whose shares are revealed in full and a fixed one-bit local leakage function on each remaining share. This is exactly the insider model of Maji, Nguyen, Paskin-Cherniavsky and Wang [88, Thm. 2]: for $(t-|\Theta|)/(n-|\Theta|) \ge \gamma > c = (2\log_2\pi - 1)/(3\log_2\pi - 2)$ and $p \ge p_0(\gamma)$, the joint distribution of the Θ shares and the leakage for secret s_0 is within $2^{-(3\log_2\pi-2)(\gamma-c)(n-|\Theta|)}$ of that for secret s_1, for any $s_0, s_1 \in \mathbb{F}_p$. Since $(t-k)/(n-k) \ge \gamma$ implies $(t-\theta)/(n-\theta) \ge \gamma$ for all $\theta \le k$ (the ratio is non-increasing in θ when $t \le n$) and $n - |\Theta| \ge n-k$, the bound is at most δ. This bounds the distance of the epoch-e leakage pair. Everything after it, in both experiments, is a function of K, the history, that leakage, fresh randomness, and $f^{(e+1)}$; by Step 1, $f^{(e+1)}$ is independent of $f^{(e)}$, hence of the real leakage, given the corrupted information, and it is likewise independent of the substitute zero-sharing, which is a function of $f^{(e)}$'s corrupted values and fresh randomness. So the remaining view is obtained from the leakage pair by a channel that is the same in both experiments, and data processing extends the bound to the whole of (K, view). Averaging over the history and over K gives the claim; the restriction to fixed Θ is where epoch-staticity is used, since the cited theorem averages over uniformly distributed Θ-shares and a set chosen after seeing some of them need not be. □

Step 4: assembling. By the claim and the triangle inequality, (K, view) in G^* and in G^*_{E+1} are within $(E+1)\delta$. In G^*_{E+1}, K is uniform in $\mathbb{F}_p$ and independent of the view, so each ideal-cipher query hits K with probability $1/p$ and $\Pr[\mathrm{Hit} \text{ in } \mathrm{G}^*_{E+1}] \le q_E/p$; since Hit is an event on (K, view), $\Pr[\mathrm{Hit} \text{ in } \mathrm{G}^*] \le q_E/p + (E+1)\delta$. Combining with Step 2,

$$\mathbf{Adv}^{\mathrm{dIND}}_{\Pi_2,\mathcal{A}} \;\le\; \mathbf{Adv}^{\mathrm{ae}}_{\mathsf{AE}}(q_D) + q_E/p + (E+1)\,\delta,$$

which is (a). Klein and Komargodski [79] lower the admissible ratio to 0.69 in the eavesdropping case ($\Theta = \emptyset$); the residual-sharing observation (the honest shares, conditioned on the corrupted ones, form a $(t-k, n-k)$ Shamir sharing of $(K - r(0))/\pi(0)$, where r interpolates the corrupted shares and π vanishes on their points, up to known affine maps per coordinate, which local leakage functions absorb) transports such eavesdropping bounds to the insider case, at the cost of the explicit constant.

Part (b). Here the adversary may be fully adaptive, and Step 1 no longer applies as stated: which parties are corrupted, and when, may depend on earlier leakage, so the corrupted shares need not be independent of K (with $n = t = 2$, leak one bit of sk_2 and corrupt P_1 only if that bit is 1; the distribution of sk_1 then depends on K). We bound the probability of guessing K from the view directly. Fix a deterministic adversary; a randomised one is a convex combination. In G^* the ideal-cipher tapes and the permutation P behind Dec are independent of K and of the sharings (Step 2), so fix them too; the interaction with the custody oracles is then a tree whose nodes are queries, determined by the answers so far, and whose edges are answers. Call an answer *algebraic* if it is a share $f^{(e)}(\alpha_i)$ or a refresh message (a value $g^{(e)}_{j'}(\alpha_i)$ or a corrupted party's own $g^{(e)}_j$), and *leaked* otherwise (a Leak answer, a profile value, or a residue); along any root-to-leaf path the charged leaked answers total at most $B = (E+1)\,n\,\ell$ bits, and the uncharged residues of corrupted shares are determined by shares already on the path (a randomised Destroy is a convex combination of deterministic ones), so they add no branching. Let $\mathbf{r}$ be the sharing randomness ($f^{(0)}$ given K, and all $g^{(e)}_j$), uniform on a set R, and for a node u and a key κ_0 let $N_{\kappa_0}(u)$ be the number of $\mathbf{r}$ consistent with the algebraic answers on the path to u when $K = \kappa_0$. Along a path at most $|C_e| \le t-1$ shares of each epoch, and refresh messages addressed to at most $t-1$ parties, are revealed, so the Shamir and Herzberg arguments of Step 1, applied to the fixed sets that the path determines, give $N_{\kappa_0}(u) = N(u)$ independent of κ_0. At an algebraic node the children partition the consistent $\mathbf{r}$, so $\sum_{\text{children}} N = N(u)$; at a leaked node with a b-bit answer there are at most 2^b children, each with $N = N(u)$. By induction from the root, $\sum_{\text{leaves } v} N(v) \le 2^B N(\text{root}) = 2^B|R|$. For a leaf v, $\Pr[\text{view} = v,\ K = \kappa_0] \le N(v)/(p|R|)$, since consistency with the leaked answers can only remove solutions. Hence any guess of K from the view succeeds with probability at most $\sum_v \max_{\kappa_0} \Pr[\text{view} = v, K = \kappa_0] \le 2^B/p = 2^{-(\kappa-(E+1)n\ell)}$, each ideal-cipher query is such a guess (a query at an interior node is a function of the path prefix, hence of the view), and $\Pr[\mathrm{Hit}] \le q_E \cdot 2^{-(\kappa-(E+1)n\ell)}$; the bound follows as in Step 4. Since corrupted shares are not charged, n may be replaced by the number of honest parties, which only improves the bound.

Part (c). Boudgoust and Simkin [28, Thm. 4] exhibit, for any t-out-of-n scheme with linear reconstruction over $\mathbb{F}_p$ and any $q = O(t)$ with $p > q > 4t$, leakage functions $\mathsf{Leak}_i(s_i) = \lfloor q\,\lambda_i s_i/p \rfloor$ of

$\log_2 q + O(1)$ bits each, where λ_i are the reconstruction coefficients of a fixed t-set, whose sum determines with certainty whether the shared secret is 0 or $(p-1)/2$. Applied to the honest shares of any epoch, this profile separates a sharing of K from a sharing of 0 whenever K lies in the half of $\mathbb{F}_p$ nearer $(p-1)/2$, so the leakage distributions that the claim of Step 3 must treat as close are at statistical distance at least $1/2$; the hybrid argument of (a) therefore fails at that budget. The attack recovers $\lfloor qK/p \rceil$ up to an additive $t/2$, a constant number of bits of K, and no more; whether Π_2 is Destruction-IND-CCA insecure at that budget is open. □

Key-hiding. Both proofs establish more than indistinguishability. In the proof of Theorem 1, conditioned on ¬Hit the adversary's view is independent of K beyond Z, so any output equals K with probability at most $2^{-(1-(E+1)\varepsilon)\kappa}$; with Pr[Hit] added, Π_1 is $(\varepsilon, 0, \mathcal{C}_\infty)$-key-hiding. In G^*_{E+1} of the proof of Theorem 2, the adversary's custody view is independent of K, so Π_2 is key-hiding up to the same $(E+1)\,\delta$ term. This is the property Proposition 4 consumes.

Proof of Theorem 3. Let Solve be the τ-step sequential algorithm that, from the public information of Π_3, outputs the decryption key (the puzzle solution, the VDF output, or the beacon's scheduled release). The adversary $\mathcal{A}_1$ calls Profile with the all-$\perp$ profile, calls Challenge(m_0, m_1) for any distinct m_0, m_1, calls Destroy, and passes c^* to $\mathcal{A}_2$. $\mathcal{A}_2 \in \mathcal{C}$ runs Solve, decrypts c^*, and outputs the bit. It succeeds whenever Solve does, with probability $1 - \mathrm{negl}(\lambda)$ by correctness of Π_3. No corruption or leakage was used. □

Proof of Proposition 4. Extractable witness encryption [64] states: for every PPT distinguisher $\mathcal{D}$ that distinguishes $\mathsf{WE.Enc}(x, m_0)$ from $\mathsf{WE.Enc}(x, m_1)$ with advantage α given auxiliary input *aux*, there is a PPT extractor $\mathcal{E}$ that, given (x, aux), outputs a witness for x with probability polynomially related to α. Let $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ be an admissible adversary against Π_4 under $\mathcal{T}$ in the experiment without Dec, with advantage α, whose custody queries all precede Challenge and which calls Destroy immediately after it. Everything $\mathcal{A}$ sees other than the challenge is then its custody view v (corrupted shares, leakage, refresh views, the profile values at destruction, residues, and the coins of $\mathcal{A}_1$ up to the challenge), which is generated by Π_{cust} from w and x before the challenge ciphertext exists and does not depend on the challenge bit. For each v let $\mathcal{D}_v$ run $\mathcal{A}_1$ to its challenge query, hand it the WE ciphertext, answer Destroy and Residue from v, and output $\mathcal{A}_2$'s bit; its advantage α_v satisfies $\mathbb{E}_v[\alpha_v] = \alpha$, so $\alpha_v \geq \alpha/2$ on a set of v of probability at least $\alpha/2$. For such v, extractability with auxiliary input (x, v) gives $\mathcal{E}$ that outputs a witness w' for x from (x, v) with probability $\mathrm{poly}(\alpha)$. The composed algorithm, which runs Π_{cust}'s oracles exactly as $\mathcal{A}_1$ does to obtain v and then $\mathcal{E}$, is an admissible adversary against Π_{cust} that outputs a witness for x with probability $(\alpha/2) \cdot \mathrm{poly}(\alpha)$; since L has unique witnesses, $w' = w$ is the custodied secret. So that probability is negligible, hence so is α. The ordering restriction is used where $\mathcal{D}_v$ answers from v: a custody query made after the challenge could depend on the ciphertext, and a distinguisher without w could not answer it. The restriction to $\mathcal{C}_\infty$ is because extractability is defined for classical $\mathcal{D}$; the omission of Dec is because WE security is stated without a decryption oracle. Without extractability, the security of WE is stated only for $x \notin L$ and says nothing here. □

D Benchmark details for the A3 reference implementation

Cryptographic-overhead benchmarks. To isolate the per-event cryptographic cost from database I/O, we benchmarked the Rust cryptographic core (the April 2026 benchmark branch; Rust 1.93.1 `--release`, criterion-rs 0.5.1, 100 samples per measurement) on an Apple M2 (8-core, 8 GB). Three operations constitute the critical path of a single event append:

(1) **SHA-256 hash-chain step** (the April 2026 version of the ledger append; production now uses a domain-separated per-subject variant of the same cost class): $1.30 \pm 0.02\,\mu$s for a typical 256-byte payload, yielding $\sim 7.7 \times 10^5$ hash-chain steps/s in pure computation.
(2) **AES-256-GCM envelope encryption** (per-record DEK): $3.75 \pm 0.05\,\mu$s for 256 B, scaling to 148 MiB/s at 16 KB. DEK generation (OS entropy) adds 940 ± 50 ns; DEK wrapping under the master key adds $2.62\,\mu$s (separate benchmarks; the composite below is lower than their sum).
(3) **Merkle-tree root construction** (`rs_merkle` SHA-256): $68\,\mu$s for 100 leaves, 8.0 ± 0.2 ms for 10^4 leaves, 79 ± 1 ms for 10^5 leaves, that is 1.3–1.5×10^6 leaves/s across sizes. Inclusion-proof verification is $O(\log n)$: $6.4\,\mu$s at 100 leaves, $13.9\,\mu$s at 10^5 leaves.

The *total per-event cryptographic overhead* (DEK generation, AES-256-GCM encrypt of 256 B, DEK wrap) is $7.02 \pm 0.01\,\mu$s (the `full_seal_crypto_256B` benchmark).

End-to-end system benchmarks on the production cloud VPS (4 vCPU AMD EPYC-Milan, 3.5 GiB, PostgreSQL 16.11, same commit) confirm that the cryptographic layer adds negligible overhead to the database-I/O–dominated path:

(1) **Event append** (Rust-side SHA-256 hash-chain link + PostgreSQL `INSERT`): $390 \pm 10\,\mu$s per event, yielding $\sim$ 2,560 events/s single-threaded. Cryptographic fraction: $7.02/390 = 1.8\%$.
(2) **Cryptographic erasure** (`Kms::destroy_key`: 4 SQL operations: SELECT key, UPDATE zero-fill, INSERT certificate, UPDATE record): 1.23 ± 0.03 ms per record, yielding $\sim$ 810 destructions/s.
(3) **Seal and store DEK** (generate DEK + AES-256-GCM encrypt + wrap DEK + `INSERT` into `kms_keys`): $356 \pm 4\,\mu$s per record, of which $7.02\,\mu$s is cryptography (2.0%).

With connection-pool parallelism (5 connections), projected throughput is $\sim$ 10–12×10^3 events/s; this projection, not a measurement, is the 10^4 figure quoted in the abstract and Table 5. The crypto budget alone permits $\sim 1.4 \times 10^5$ events/s, leaving an order-of-magnitude margin for the database.

Measured cryptographic costs for A4 and A5. To replace two estimated rows by measurements, we benchmarked public implementations on the same Apple M2 (8 GB, macOS 26.0, rustc 1.93.1, criterion-rs 0.5.1, 100 samples, 20 s windows, single-threaded, opt-level 3 with thin LTO). *A5:* with the `tlock` 0.0.10 crate [92] against the drand quicknet chain (round 10^7, signature fetched and checked), time-lock encryption of a 32-byte (256-byte) payload

takes 2.10 ± 0.04 ms (2.09 ± 0.12 ms) and decryption with the round signature 1.53 ± 0.06 ms (1.56 ± 0.05 ms), about 477 encryptions and 650 decryptions per second per core; one BLS12-381 pairing costs 0.81 ms (arkworks) to 0.95 ms (zkcrypto). This is of the same order as the 539 encryptions and 1,014 decryptions per second reported for the Go implementation [53]. *A4:* with `threshold_crypto` 0.4.0 [99] at $(t, n) = (3, 5)$ and $(5, 7)$, encrypting a 32-byte message under the committee key takes 1.97 ± 0.13 ms and 1.92 ± 0.03 ms (about 510 per second); one decryption share costs 0.15 ms without, or 4.2 (4.0) ms with, ciphertext verification (two pairings); combining t shares costs 0.21 and 0.39 ms; and generating a fresh key set with its public commitment, the local part of a reshare, takes 0.43 and 0.71 ms. The cryptographic path of one threshold decryption is therefore 0.35–0.54 ms unverified (one share plus the combine) and 12–20 ms when the combiner verifies all t shares; the network rounds of the DKG and reshare protocols, the drand round wait, and multi-core scaling were not measured. Raw outputs are in the artifact.

D.1 Basis for the estimated rows of Table 5

Rows A1, A2 and A6, and the non-cryptographic parts of A4 and A5, are not measured by us. This section records, per cell, the published number or the arithmetic each estimate rests on, so that a reader can redo it.

Gas (C3). Ethereum charges 21,000 gas per transaction, 16 gas per non-zero calldata byte, and 20,000 gas to set a fresh storage slot [118]. Recording a 32-byte destruction attestation in calldata therefore costs $\approx$ 21.5K gas (A2, A6); the same written to contract storage would cost $\approx$ 41K. For A4, CHURP writes n hashes ($32n$ bytes) to the chain per handoff [90]; for $n = 100$ that is $21{,}000 + 3{,}200 \cdot 16 \approx 72$K gas in calldata and $\approx$ 2M in storage, which we summarise as $\sim 10^5$. A5 has no destruction operation to record (Theorem 3), hence n/a.

Latency (C2). CHURP reports a handoff for a committee of 100 in 2.92 s on a LAN and 4.54 s on a WAN, plus one on-chain write per node, which at the time of that paper took about 15 s on Ethereum [90]; destruction is a quorum refusal followed by share erasure and lands in the same 5–30 s band; the cryptographic work of one decryption-share round, measured above, is under 25 ms, so the band is set by the network. For A5 the relevant latency is the configured delay τ itself: the drand default chain emits a round every 30 s and quicknet every 3 s (chain info endpoints, read 2026-10-11), so τ is a multiple of those. A6 inherits the latency of whatever custody holds the witness.

Throughput (C1). A1 is bounded by SHA-256 over the record, which our own measurement puts at $\sim 7 \times 10^5$ hash-chain steps per second (§D); 10^5 allows for I/O. A2 assumes an HSM-resident AES key; vendor-rated AES throughput is well above 10^4 operations per second, and the figure is set by the surrounding I/O as in A3. A4 and A5 are measured above; C1 counts encryptions, the operation an append performs, at about $5 \cdot 10^2$ per core for both; a deployment that verifies decryption shares (as a CCA-secure one must) decrypts at 50–80 per core, one that does not at $2\text{–}3 \cdot 10^3$. A6 is derived: decryption under Derler–Slamanig witness encryption verifies a Groth–Sahai proof, which costs tens of pairings for the statements of interest; at 0.8–0.95 ms per pairing (measured above) that is $\sim 10^2$ decryptions per second. We have not run the Derler–Slamanig construction ourselves.

Leakage budget (C8). A2 and A4 take the attested-zeroisation budget $\ell = 0$ of Table 3. A3's budget is dominated by its backup window of up to ten minutes (§6.5), after which the software allowance applies. A5 destroys nothing and A6 destroys the witness, so their budget is not a property of the gate: n/a and "as custody" respectively.

Regulatory criteria (C9–C11). C9 is on-chain for A4 because quorum refusal is recorded on the chain the committee lives on [23, 90]; vendor for A2 and the operator's own certificate for A3; beacon for A5; weak for A1 and A6 because verifying a deletion or the destruction of a witness is proving a non-existence statement. C10 follows Def. 3 and the residue rule of §5.5: A1 fails because it has no key to destroy and its plain hash is linkable; A3 is "after 10 min" because during its backup window the declared budget is the whole key (Table 3); A5 fails by Theorem 3; A6 is partial because its security is conditional on extractability (Proposition 4), which the construction is not stated to provide. C11 is our judgement of whether a selective-disclosure proof can be attached in the same session without new trust assumptions.

References

[1] Aydin Abadi. 2024. Tempora-Fusion: Time-Lock Puzzle with Efficient Verifiable Homomorphic Linear Combination. arXiv:2406.15070; IACR eprint 2024/1013.

[2] Aydin Abadi, Dan Ristea, Artem Grigor, and Steven J. Murdoch. 2025. Scalable Time-Lock Puzzle. In *ACM AsiaCCS*.

[3] Shweta Agrawal, Giulio Malavolta, and Tianwei Zhang. 2024. Time-Lock Puzzles from Lattices. In *CRYPTO*. doi:10.1007/978-3-031-68382-4_13

[4] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. 2009. Simultaneous Hardcore Bits and Cryptography Against Memory Attacks. In *TCC*. doi:10.1007/978-3-642-00457-5_28

[5] Jannik Albrecht and Ghassan Karame. 2026. On the Effectiveness of Mempool-based Transaction Auditing. In *WWW*.

[6] Martin R. Albrecht, Benjamin Bencina, and Russell W. F. Lai. 2025. Hollow LWE: A New Spin — Unbounded Updatable Encryption from LWE and PCE. In *EUROCRYPT*.

[7] Martin R. Albrecht, Benjamin Dowling, and Daniel Jones. 2025. Formal Analysis of Multi-Device Group Messaging in WhatsApp. In *EUROCRYPT*.

[8] Martin R. Albrecht, Nicolas Gama, James Howe, and Anand Kumar Narayanan. 2025. Post-Quantum Online/Offline Signatures. In *CT-RSA*. doi:10.1007/978-3-031-88661-4_12

[9] Martin R. Albrecht, Lenka Mareková, Kenneth G. Paterson, and Igors Stepanovs. 2026. Four Attacks and a Proof for Telegram. *Journal of Cryptology* (2026).

[10] Shams Mhmood Abd Ali, Mohd Najwadi Yusoff, and Hasan Falah Hasan. 2023. Redactable Blockchain: Comprehensive Review, Mechanisms, Challenges, Open Issues and Future Research Directions. *Future Internet (MDPI)* 15, 1 (2023), 35. doi:10.3390/fi15010035

[11] Andris Ambainis, Mike Hamburg, and Dominique Unruh. 2019. Quantum Security Proofs Using Semi-classical Oracles. In *Advances in Cryptology – CRYPTO*. 269–295. doi:10.1007/978-3-030-26951-7_10

[12] Giuseppe Ateniese, Bernardo Magri, Daniele Venturi, and Ewerton Andrade. 2017. Redactable Blockchain – or – Rewriting History in Bitcoin and Friends. In *IEEE EuroS&P*.

[13] Gennaro Avitabile, Vincenzo Botta, Daniele Friolo, and Ivan Visconti. 2026. Data Redaction in Smart-Contract-Enabled Permissioned Blockchains. *Blockchain: Research and Applications* 7, 4 (2026), 100363. doi:10.1016/j.bcra.2025.100363

[14] Gennaro Avitabile, Nico Döttling, Bernardo Magri, Christos Sakkas, and Stella Wohnig. 2024. Signature-Based Witness Encryption with Compact Ciphertext. In *ASIACRYPT*.

[15] Asaph Azaria, Ariel Ekblaw, Thiago Vieira, and Andrew Lippman. 2016. MedRec: Using Blockchain for Medical Data Access and Permissions Management. In *IEEE OBD*.

[16] Mariarosaria Barbaraci, Michael Senn, Noah Schmid, Orestis Alpos, and Christian Cachin. 2026. Thetacrypt: A Distributed Service for Threshold Cryptography. In *Latin-American Symposium on Dependable and Secure Computing (LADC 2025), CCIS*. doi:10.1007/978-3-032-11539-3_10

[17] James Bartusek and Dakshita Khurana. 2023. Cryptography with Certified Deletion. In *Advances in Cryptology – CRYPTO*. 192–223. doi:10.1007/978-3-031-38554-4_7

[18] Amik Raj Behera, Pierre Meyer, Claudio Orlandi, Lawrence Roy, and Peter Scholl. 2025. Privately Constrained PRFs from DCR: Puncturing and Bounded Waring Rank. In *TCC*.

[19] Rahime Belen-Saglam, Enes Altuncu, Yang Lu, and Shujun Li. 2023. A Systematic Literature Review of the Tension between the GDPR and Public Blockchain Systems. *Blockchain: Research and Applications* 4, 2 (2023), 100129. doi:10.1016/j.bcra.2023.100129

[20] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. 1998. Relations Among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology – CRYPTO '98 (Lecture Notes in Computer Science, Vol. 1462)*. Springer, 26–45. doi:10.1007/BFb0055718

[21] Juan Benet. 2014. IPFS — Content Addressed, Versioned, P2P File System. arXiv:1407.3561.

[22] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. 2018. On the Local Leakage Resilience of Linear Secret Sharing Schemes. In *CRYPTO*. doi:10.1007/978-3-319-96884-1_18

[23] Fabrice Benhamouda, Craig Gentry, Sergey Gorbunov, Shai Halevi, Hugo Krawczyk, Chengyu Lin, Tal Rabin, and Leonid Reyzin. 2020. Can a Public Blockchain Keep a Secret?. In *Theory of Cryptography (TCC)*. 260–290. doi:10.1007/978-3-030-64375-1_10

[24] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. 2018. Verifiable Delay Functions. In *CRYPTO*.

[25] Dan Boneh and Richard J. Lipton. 1996. A Revocable Backup System. In *6th USENIX Security Symposium*.

[26] Jan Bormet, Jonas Hofmann, and Hussien Othman. 2025. Traceable Threshold Encryption Without a Trusted Dealer. In *LNCS*. doi:10.1007/978-981-95-5119-4_15

[27] Vincenzo Botta, Vincenzo Iovino, and Ivan Visconti. 2022. Towards Data Redaction in Bitcoin. *IEEE Transactions on Network and Service Management* 19, 4 (2022), 3872–3883. doi:10.1109/TNSM.2022.3214279

[28] Katharina Boudgoust and Mark Simkin. 2026. Scale, Round, Break: Simple Leakage Attacks on Secret Sharing Schemes. In *ASIACRYPT*. IACR ePrint 2026/833.

[29] Lennart Braun, Guilhem Castagnos, Ivan Damgård, Fabien Laguillaumie, Kelsey Melissaris, Claudio Orlandi, and Ida Tucker. 2024. An Improved Threshold Homomorphic Cryptosystem Based on Class Groups. In *SCN*.

[30] Anne Broadbent and Rabib Islam. 2020. Quantum Encryption with Certified Deletion. In *Theory of Cryptography (TCC)*. 92–122. doi:10.1007/978-3-030-64381-2_4

[31] Randal Burns and Zachary N. J. Peterson. 2010. Security Constructs for Regulatory-Compliant Storage. *Commun. ACM* (2010).

[32] Sergiu Bursuc, Reynaldo Gil-Pons, Sjouke Mauw, and Rolando Trujillo-Rasua. 2024. Software-Based Memory Erasure with Relaxed Isolation Requirements. In *IEEE Computer Security Foundations Symposium (CSF)*.

[33] Federico Calandra, Marco Bernardo, Andrea Esposito, and Francesco Fabris. 2025. Redactable Blockchains: An Overview. arXiv:2508.08898.

[34] Matteo Campanelli, Bernardo David, Hamidreza Khoshakhlagh, Anders Konring, and Jesper Buus Nielsen. 2022. Encryption to the Future: A Paradigm for Sending Secret Messages to Future (Anonymous) Committees. In *Advances in Cryptology – ASIACRYPT*. 151–180. doi:10.1007/978-3-031-22969-5_6

[35] Ran Canetti, Ivan Damgård, Sebastian Kolby, Divya Ravi, and Sophia Yakoubov. 2025. Deniable Secret Sharing. In *TCC*. doi:10.1007/978-3-032-12293-3_13

[36] Ignacio Cascudo, Bernardo David, Lydia Garms, and Anders Konring. 2022. YOLO YOSO: Fast and Simple Encryption and Secret Sharing in the YOSO Model. In *LNCS (Asiacrypt)*. doi:10.1007/978-3-031-22963-3_22

[37] Michele Ciampi, Aggelos Kiayias, and Yu Shen. 2025. Universally Composable Transaction Order Fairness: Refined Definitions and Adaptive Security. In *ASIACRYPT*.

[38] Katriel Cohn-Gordon, Cas Cremers, and Luke Garratt. 2016. On Post-Compromise Security. In *IEEE Computer Security Foundations Symposium (CSF)*. 164–178. doi:10.1109/CSF.2016.19

[39] Commission Nationale de l'Informatique et des Libertés (CNIL). 2018. Blockchain: Solutions for a Responsible Use of the Blockchain in the Context of Personal Data. https://www.cnil.fr/sites/cnil/files/atoms/files/blockchain_en.pdf. September 2018; English version 29 October 2018.

[40] David Derler and Daniel Slamanig. 2018. Practical witness encryption for algebraic languages or how to encrypt under Groth–Sahai proofs. *Designs, Codes and Cryptography* (2018). doi:10.1007/s10623-018-0460-y

[41] Dominic Deuber, Bernardo Magri, and Sri Aravinda Krishnan Thyagarajan. 2019. Redactable Blockchain in the Permissionless Setting. In *IEEE S&P*.

[42] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. 2008. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM J. Comput.* 38, 1 (2008). doi:10.1137/060651380

[43] Nico Döttling, Lucjan Hanzlik, Bernardo Magri, and Stella Wohnig. 2023. McFly: Verifiable Encryption to the Future Made Practical. In *Financial Cryptography*.

[44] Raphael Esu. 2025. Blockchain and data protection: developing a functional erasure framework for storage limitation. *Information & Communications Technology Law* (2025). doi:10.1080/13600834.2025.2600197

[45] European Data Protection Board. 2025. Guidelines 02/2025 on Processing of Personal Data through Blockchain Technologies.

[46] European Data Protection Board. 2026. Guidelines 02/2026 on Anonymisation (version for public consultation). Adopted 7 July 2026.

[47] Tiago M. Fernández-Caramés and Paula Fraga-Lamas. 2020. Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks. *IEEE Access* (2020).

[48] Michèle Finck. 2018. Blockchains and Data Protection in the European Union. *European Data Protection Law Review* (2018).

[49] Michèle Finck. 2019. *Blockchain and the General Data Protection Regulation: Can Distributed Ledgers Be Squared with European Data Protection Law?* Technical Report PE 634.445. European Parliamentary Research Service, Scientific Foresight Unit (STOA). doi:10.2861/535

[50] Michèle Finck and Frank Pallas. 2020. They who must not be identified — distinguishing personal from non-personal data under the GDPR. *International Data Privacy Law* (2020). doi:10.1093/idpl/ipz026

[51] Flashbots. 2023. SUAVE: Single Unifying Auction for Value Expression. https://writings.flashbots.net/the-future-of-mev-is-suave.

[52] Martin Florian, Sebastian Henningsen, Sophie Beaucamp, and Björn Scheuermann. 2019. Erasing Data from Blockchain Nodes. In *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 367–376. doi:10.1109/EuroSPW.2019.00047

[53] Nicolas Gailly, Kelsey Melissaris, and Yolan Romailler. 2023. tlock: Practical Timelock Encryption from Threshold BLS. IACR ePrint 2023/189.

[54] Simson L. Garfinkel and David J. Malan. 2006. One Big File Is Not Enough: A Critical Evaluation of the Dominant Free-Space Sanitization Technique. In *Privacy Enhancing Technologies (PET)*. 135–151. doi:10.1007/11957454_8

[55] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. 2014. On the Implausibility of Differing-Inputs Obfuscation and Extractable Witness Encryption with Auxiliary Input. In *Advances in Cryptology – CRYPTO*. 518–535. doi:10.1007/978-3-662-44371-2_29

[56] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. 2013. Witness Encryption and Its Applications. In *STOC*.

[57] Sanjam Garg, Shafi Goldwasser, and Prashant Nalini Vasudevan. 2020. Formalizing Data Deletion in the Context of the Right to be Forgotten. In *EUROCRYPT*.

[58] Roxana Geambasu, Tadayoshi Kohno, Amit A. Levy, and Henry M. Levy. 2009. Vanish: Increasing Data Privacy with Self-Destructing Data. In *18th USENIX Security Symposium*. 299–316.

[59] Pierre Ghaly, Harald Gjermundrød, and Ioanna Dionysiou. 2025. Privacy-Preserving "Right to Be Forgotten" in Configurable Blockchain Tokenization. In *Conference on Blockchain Research & Applications (BRAINS)*. doi:10.1109/BRAINS67003.2025.11302955

[60] Reynaldo Gil-Pons, Sjouke Mauw, and Rolando Trujillo-Rasua. 2025. Empirical Evaluation of Memory-Erasure Protocols. In *SECRYPT*.

[61] Jonathan Godin and Philippe Lamontagne. 2021. Deletion-Compliance in the Absence of Privacy. In *Privacy, Security and Trust (PST)*. doi:10.1109/PST52912.2021.9647774

[62] Mateusz Godyn, Michal Kedziora, Yingying Ren, Yongxin Liu, and Houbing Herbert Song. 2022. Analysis of solutions for a blockchain compliance with GDPR. *Scientific Reports* 12 (2022), 15021. doi:10.1038/s41598-022-19341-y

[63] Oded Goldreich. 2004. *Foundations of Cryptography, Volume II: Basic Applications*. Cambridge University Press.

[64] Shafi Goldwasser, Yael Tauman Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. 2013. How to Run Turing Machines on Encrypted Data. In *CRYPTO*. doi:10.1007/978-3-642-40084-1_30

[65] Shafi Goldwasser and Silvio Micali. 1984. Probabilistic Encryption. *J. Comput. System Sci.* 28, 2 (1984), 270–299. doi:10.1016/0022-0000(84)90070-9

[66] Rishab Goyal, Venkata Koppula, and Brent Waters. 2017. Lockable Obfuscation. In *FOCS*.

[67] Vipul Goyal, Abhiram Kothapalli, Elisaweta Masserova, Bryan Parno, and Yifan Song. 2022. Storing and Retrieving Secrets on a Blockchain. In *PKC*. doi:10.1007/978-3-030-97121-2_10

[68] Matthew D. Green and Ian Miers. 2015. Forward Secure Asynchronous Messaging from Puncturable Encryption. In *IEEE S&P*. 305–320. doi:10.1109/SP.2015.26

[69] Mathias Hall-Andersen, Mark Simkin, and Benedikt Wagner. 2025. Foundations of Data Availability Sampling. In *IACR Communications in Cryptology*. doi:10.62056/a09qudhdj

[70] Feng Hao, Dylan Clarke, and Avelino Francisco Zorzo. 2016. Deleting Secret Data with Public Verifiability. *IEEE Transactions on Dependable and Secure Computing* 13, 6 (2016), 617–629. doi:10.1109/TDSC.2015.2423684

[71] A. K. M. Bahalul Haque, A. K. M. Najmul Islam, Sami Hyrynsalmi, Bilal Naqvi, and Kari Smolander. 2021. GDPR Compliant Blockchains – A Systematic Literature Review. *IEEE Access* 9 (2021), 50593–50606. doi:10.1109/ACCESS.2021.3069877

[72] Amir Herzberg, Stanisław Jarecki, Hugo Krawczyk, and Moti Yung. 1995. Proactive Secret Sharing Or: How to Cope With Perpetual Leakage. In *CRYPTO*. doi:10.1007/3-540-44750-4_27

[73] Peter Hillmann, Marcus Knüpfer, Erik Heiland, and Andreas Karcher. 2020. Selective Deletion in a Blockchain. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*. doi:10.1109/ICDCS47774.2020.00160

[74] Md Shariful Islam and M. Sohel Rahman. 2025. LogStamping: A Blockchain-Based Log Auditing Approach for Large-Scale Systems. arXiv:2505.17236.

[75] Nikolaos P. Karvelas and Aggelos Kiayias. 2014. Efficient Proofs of Secure Erasure. In *Security and Cryptography for Networks (SCN)*. 520–537. doi:10.1007/978-3-319-10879-7_30

[76] Alireza Kavousi, Aydin Abadi, and Philipp Jovanovic. 2024. Timed Secret Sharing. In *LNCS (ASIACRYPT)*. doi:10.1007/978-981-96-0941-3_5

[77] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. 2020. Order-Fairness for Byzantine Consensus. In *CRYPTO*. doi:10.1007/978-3-030-56877-1_16

[78] Mehmet Sabir Kiraz, Enrique Larraia, and Owen Vaughan. 2024. How to Redact the Bitcoin Backbone Protocol. In *IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. Full version: IACR ePrint 2024/813.

[79] Ohad Klein and Ilan Komargodski. 2023. New Bounds on the Local Leakage Resilience of Shamir's Secret Sharing Scheme. In *CRYPTO*. IACR ePrint 2023/805.

[80] Eleftherios Kokoris-Kogias, Enis Ceyhun Alp, Linus Gasser, Philipp Jovanovic, Ewa Syta, and Bryan Ford. 2021. CALYPSO: Private Data Management for Decentralized Ledgers. *Proceedings of the VLDB Endowment* 14, 4 (2021), 586–599. doi:10.14778/3436905.3436917

[81] Chelsea Komlo and Ian Goldberg. 2020. FROST: Flexible Round-Optimized Schnorr Threshold Signatures. IACR eprint 2020/852.

[82] League of Entropy. 2022. drand: Distributed Randomness Beacon. https://drand.love.

[83] Xinyu Li, Jing Xu, Xiong Fan, Yuchen Wang, and Zhenfeng Zhang. 2020. Puncturable Signatures and Applications in Proof-of-Stake Blockchain Protocols. *IEEE Transactions on Information Forensics and Security* (2020).

[84] Yiming Li and Shengli Liu. 2024. Tagged Chameleon Hash from Lattices and Application to Redactable Blockchain. In *Public Key Cryptography (PKC)*. doi:10.1007/978-3-031-57725-3_10

[85] Jia Liu, Tibor Jager, Saqib A. Kakvi, and Bogdan Warinschi. 2018. How to Build Time-Lock Encryption. *Designs, Codes and Cryptography* 86, 11 (2018), 2549–2586. doi:10.1007/s10623-018-0461-x

[86] Wenhua Lyu, Shujie Li, Hongbing Zhang, Jingyu Li, Zhuo Huang, and Fuyou Lin. 2026. Chameleon Hash-Based Redactable Blockchains: A Systematic Literature Review. In *Algorithms and Architectures for Parallel Processing (ICA3PP 2025)*. Springer. doi:10.1007/978-981-95-8399-7_17

[87] Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, Tom Suad, and Mingyuan Wang. 2021. Leakage-Resilience of the Shamir Secret-Sharing Scheme Against Physical-Bit Leakages. In *EUROCRYPT*. doi:10.1007/978-3-030-77886-6_12

[88] Hemanta K. Maji, Hai H. Nguyen, Anat Paskin-Cherniavsky, and Mingyuan Wang. 2022. Improved Bound on the Local Leakage-resilience of Shamir's Secret Sharing. In *IEEE International Symposium on Information Theory (ISIT)*. doi:10.1109/ISIT50566.2022.9834695

[89] Deepak Maram, Harjasleen Malvai, Fan Zhang, Nerla Jean-Louis, Alexander Frolov, Tyler Kell, Tyrone Lobban, Christine Moy, Ari Juels, and Andrew Miller. 2021. CanDID: Can-Do Decentralized Identity with Legacy Compatibility, Sybil-Resistance, and Accountability. In *IEEE S&P*.

[90] Sai Krishna Deepak Maram, Fan Zhang, Lun Wang, Andrew Low, Yupeng Zhang, Ari Juels, and Dawn Song. 2019. CHURP: Dynamic-Committee Proactive Secret Sharing. In *ACM CCS*.

[91] Roman Matzutt, Jens Hiller, Martin Henze, Jan Henrik Ziegeldorf, Dirk Müllmann, Oliver Hohlfeld, and Klaus Wehrle. 2018. A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin. In *Financial Cryptography and Data Security (FC)*. doi:10.1007/978-3-662-58387-6_23

[92] Thibault Meunier. 2026. tlock: Rust Implementation of Timelock Encryption (crate `tlock` 0.0.10). https://crates.io/crates/tlock.

[93] Moni Naor and Gil Segev. 2009. Public-Key Cryptosystems Resilient to Key Leakage. In *CRYPTO*. doi:10.1007/978-3-642-03356-8_2

[94] National Institute of Standards and Technology. 2024. FIPS 203: Module-Lattice-Based Key-Encapsulation Mechanism Standard. NIST FIPS 203. doi:10.6028/NIST.FIPS.203

[95] NIST. 2014. SP 800-88 Rev. 1: Guidelines for Media Sanitization.

[96] Anand Manojkumar Parikh, Shamik Sural, Vijayalakshmi Atluri, and Jaideep Vaidya. 2025. Enabling Right to be Forgotten in a Collaborative Environment Using Permissioned Blockchain. In *LNCS*. doi:10.1007/978-3-031-96590-6_9

[97] Daniele Perito and Gene Tsudik. 2010. Secure Code Update for Embedded Devices via Proofs of Secure Erasure. In *ESORICS*. 643–662. doi:10.1007/978-3-642-15497-3_39

[98] Radia Perlman. 2005. File System Design with Assured Delete. In *Third IEEE International Security in Storage Workshop (SISW)*. 83–88. doi:10.1109/SISW.2005.5 Also as Sun Microsystems Labs TR-2005-140, "The Ephemerizer: Making Data Disappear".

[99] POA Network. 2020. threshold_crypto: Pairing-Based Threshold Cryptosystem for Collaborative Decryption and Signatures (Rust crate 0.4.0). https://github.com/poanetwork/threshold_crypto.

[100] Eugenia Politou, Fran Casino, Efthimios Alepis, and Constantinos Patsakis. 2021. Blockchain Mutability: Challenges and Proposed Solutions. *IEEE Transactions on Emerging Topics in Computing* 9, 4 (2021), 1972–1986. doi:10.1109/TETC.2019.2949510

[101] Ivan Puddu, Alexandra Dmitrienko, and Srdjan Capkun. 2017. μchain: How to Forget without Hard Forks. IACR Cryptology ePrint Archive, Report 2017/106.

[102] Kopo M. Ramokapane and Awais Rashid. 2023. ExD: Explainable Deletion. In *New Security Paradigms Workshop (NSPW)*. doi:10.1145/3633500.3633503

[103] Kopo M. Ramokapane, Awais Rashid, and Jose M. Such. 2016. Assured Deletion in the Cloud: Requirements, Challenges and Future Directions. In *CCSW (Cloud Computing Security Workshop)*. doi:10.1145/2996429.2996434

[104] Joel Reardon. 2016. *Secure Data Deletion*. Springer, Information Security and Cryptography. doi:10.1007/978-3-319-28778-2

[105] Joel Reardon, David Basin, and Srdjan Capkun. 2013. SoK: Secure Data Deletion. In *IEEE S&P*.

[106] Ronald L. Rivest, Adi Shamir, and David A. Wagner. 1996. *Time-lock Puzzles and Timed-release Crypto*. Technical Report. MIT LCS.

[107] Phillip Rogaway. 2002. Authenticated-Encryption with Associated-Data. In *ACM CCS*. doi:10.1145/586110.586125

[108] Vincent Schlatt, Johannes Sedlmeir, Simon Feulner, and Nils Urbach. 2021. Designing a Framework for Digital KYC Processes Built on Blockchain-Based Self-Sovereign Identity. *Information & Management* (2021).

[109] Johannes Sedlmeir, Jonathan Lautenschlager, Gilbert Fridgen, and Nils Urbach. 2022. The transparency challenge of blockchain in organizations. In *Electronic Markets*. doi:10.1007/s12525-022-00536-0

[110] Adi Shamir. 1979. How to Share a Secret. In *CACM*.

[111] Yang Tang, Patrick P. C. Lee, John C. S. Lui, and Radia Perlman. 2012. Secure Overlay Cloud Storage with Access Control and Assured Deletion. *IEEE Transactions on Dependable and Secure Computing* 9, 6 (2012), 903–916. doi:10.1109/TDSC.2012.49

[112] Mohit Bhasi Thazhath, Jan Michalak, and Thang Hoang. 2022. Harpocrates: Privacy-Preserving and Immutable Audit Log for Sensitive Data Operations. arXiv:2211.04741.

[113] Peter Todd. 2016. OpenTimestamps: Scalable, Trust-Minimized, Distributed Timestamping with Bitcoin. https://opentimestamps.org.

[114] Dominique Unruh. 2015. Revocable Quantum Timed-Release Encryption. *J. ACM* 62, 6 (2015), 49:1–49:76. doi:10.1145/2817206

[115] W3C. 2025. Verifiable Credentials Data Model v2.0. W3C Recommendation, https://www.w3.org/TR/vc-data-model-2.0/.

[116] Zhipeng Wang, Orestis Alpos, Alireza Kavousi, Harry W. H. Wong, Sze Yiu Chau, Duc Viet Le, and Christian Cachin. 2025. DSKE: Digital Signatures with Key Extraction. In *CT-RSA*.

[117] Scott Wolchok, Owen S. Hofmann, Nadia Heninger, Edward W. Felten, J. Alex Halderman, Christopher J. Rossbach, Brent Waters, and Emmett Witchel. 2010. Defeating Vanish with Low-Cost Sybil Attacks Against Large DHTs. In *Network and Distributed System Security Symposium (NDSS)*.

[118] Gavin Wood. 2014. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Project Yellow Paper. Appendix G, fee schedule: $G_{transaction} = 21000$, $G_{sset} = 20000$, $G_{txdatanonzero} = 16$.

[119] Chunhui Wu, Lishan Ke, and Yusong Du. 2021. Quantum-resistant key-exposure-free chameleon hash and applications in redactable blockchain. *Information Sciences* (2021). doi:10.1016/j.ins.2020.10.008

[120] Yang Xu, Shihao Xiao, Huiling Wang, Cheng Zhang, Zhifei Ni, Wenjun Zhao, and Guojun Wang. 2024. Redactable Blockchain-Based Secure and Accountable Data Management. *IEEE Transactions on Network and Service Management* (2024). doi:10.1109/TNSM.2023.3255265

[121] James Hsin yu Chiang, Ivan Damgård, William R. Duro, Sunniva Engan, Sebastian Kolby, and Peter Scholl. 2025. Post-Quantum Threshold Ring Signature Applications from VOLE-in-the-Head. In *ACM CCS*.

[122] Ammar Zafar. 2025. Reconciling Blockchain Technology and Data Protection Laws: Regulatory Challenges, Technical Solutions, and Practical Pathways. *Journal of Cybersecurity* 11, 1 (2025). doi:10.1093/cybsec/tyaf002

[123] Fu Zhang, Yuqing Zhang, Zhaofeng Ma, Lin Sun, and Yushi Shen. 2026. A Regulatable Blockchain Rewriting Scheme for Identity-Aware Data Modification and User Identity Update. *ACM Transactions on the Web* (2026). doi:10.1145/3765896

[124] Qionglu Zhang, Shijie Jia, Bing Chang, and Bo Chen. 2018. Ensuring data confidentiality via plausibly deniable encryption and secure deletion — a survey. *Cybersecurity* (2018). doi:10.1186/s42400-018-0005-8

[125] Zongyang Zhang, Bin Hu, Han Chen, You Zhou, Huazu Jiang, and Jianwei Liu. 2026. DyCAPS: Asynchronous Dynamic-Committee Proactive Secret Sharing. In *Chinese J. Electronics*. doi:10.23919/cje.2025.00.072

[126] Guy Zyskind, Oz Nathan, and Alex Pentland. 2015. Decentralizing Privacy: Using Blockchain to Protect Personal Data. In *IEEE Security and Privacy Workshops (SPW)*. 180–184. doi:10.1109/SPW.2015.27