FlowMemory

Whitepaper — Full Reference Corpus

June 2026

- 8. How to reproduce

Use cases and killer products

- 1. Why the use cases, not the primitives
- 2. The Empty Witness — a court-handable proof your AI was never told X
- 3. Discovery Shield — non-release and no-consent, surviving a decryptor
- 4. PAIR — the Provable AI Interaction Record (full-turn, court-handable)
- 5. Grounding Receipt — proof that an answer came from exactly your memory
- 6. Memory Immunity — anti-poisoning proof-of-exclusion for agents
- 7. Forget and consolidate
- 8. Soul Will — consent-bound memory inheritance
- 9. Enterprise audit and individual sovereignty
- 10. Honest negatives — what we proved we cannot do
- 11. The $58 hardware node
- 12. Honest scope, in one place
- 13. Test evidence for this section

Decentralized AI, the SanDisk thesis, and the c0mpute seam

- 1. The question this document answers
- 2. Why it is good for decentralized AI
- 3. The substrate thesis and the two halves
- 4. The defensible position
- 5. The thesis in one line

The privacy-computation layer (ZK / MPC family)

- 1. What this layer is, and what it is not
- 2. The Pedersen impedance (state this honestly)
- 3. The proven primitives
- 4. In-flight (do not rely on as proven yet)
- 5. How this layer composes with the rest of Sovereign Memory

Sovereign Memory — executive summary

The bet. As frontier models commoditize, the durable strategic asset is **memory** — what an AI knows about you, persistently, across sessions and vendors. Today that memory sits in vendor silos (ChatGPT, Claude, Gemini) or developer-owned databases (Mem0, Letta, Zep). In every case *someone other than you* holds it, can read it, and can be compelled to produce it. **Sovereign Memory** is the inverse: a **decentralized, private, user-owned memory layer for AI** — "the SanDisk of decentralized AI memory." It lives across *your own* devices; no operator, no account, no token; delete the company and it keeps working.

> **NOVELTY AUDIT FINDINGS (binding, `research/NOVELTY_AUDIT.md`).** An independent audit found **0 of 8** strongest claims novel-as-PRIMITIVE. The cryptography is known classical-DL/DDH machinery, soundly implemented and independently red-teamed. The contribution is (a) novel **compositions/applications** onto user-owned AI memory, (b) the reusable completeness-over-authenticated-scope pattern, (c) **one** genuinely-interesting standout — **Forgetting-Closure**, a value-blind, fail-closed, operator-untrusted receipt that a forgotten fact is unreachable in the deductive closure of the signed surviving memory under a fixed public monotone rule-set (it fills the external-verifiability gap the DB inference-aware-deletion literature, P2E2 VLDB 2025, names as open; scoped caveat: *unreachable under rule-set R*, not *unrecoverable*), (d) an honest negatives + novelty map, (e) the trust-only-independent-verification meta-lesson. Two earlier results are downgraded to mostly-prior-art and credited: ZK Predicate Memory (→ Camenisch–Chaabouni–shelat ASIACRYPT 2008 + CL/Idemix + CDS 1994 + arXiv 2502.06425) and Consistency / Contradiction-Rejection (→ PoneglyphDB SIGMOD 2025 + ZKSQL VLDB 2023). Tally: **0 novel-as-primitive / 6 novel-as-composition / 2 mostly-prior-art / 1 standout.** Neither downgrade is broken — these are novelty downgrades, not correctness.

The honesty contract is the moat, so it is stated first. Signatures and receipts prove **custody, integrity, ordering, consent, minimization, grounding, non-supply, exclusion, and forgetting** — they do **not** prove confidentiality-during-compute, are **not** FHE, do **not** prove what a model internally attended to, and do **not** prove semantic truth. On-device means *locality*, not homomorphic compute. Novelty here is **fusion-grade and application-grade**, not a new cryptographic primitive — the program hunted for a new primitive across ten frontiers and proved the frontier is literature-occupied. Every claim below is backed by a runnable prototype plus an independent adversarial suite that exits zero, or it is labelled a stated limit.

The proven set

All prototypes are self-contained Node (built-ins only, no npm, no network); each was written by one agent and attacked by an independent agent that imported it unmodified. No check was ever weakened to pass — fixes only ever made a verifier stricter or fail-closed.

Core substrate

- **TRM (threshold-resident memory)** — memory exists *nowhere in full*; **read-consent is cryptographically separate from decryption**. A party holding the recall key *and* the plaintext still cannot forge proof that your devices consented to a read (`trm/` , T2; 17 independent forgeries rejected). Closest prior art: threshold-Schnorr/FROST + Shamir, used unmodified; the wedge is running two independent threshold splits over one device roster.
- **Release Ledger + the Empty Witness** — an append-only RFC-6962 Merkle log of every write / admit / release / forget, plus a court-handable, offline-verifiable **proof of a negative**: *"fact X was never released into AI session S"* (`proof-of-non-supply/` , query-bound, append-only-growth checked). Closest prior art: SCITT refusal-events and ZKPROV (arXiv 2506.20915) prove a *logged* or *positive* event; the Empty Witness occupies the vacated negative direction they admit they cannot reach.
- **QRSM (recomputed-never-stored index)** — the queryable projection is rebuilt in RAM and gated on a live physical-token quorum; seizing the powered-off device yields only ciphertext and an index that was never written (`qrsm/` ; verified *not* to collapse to shares-at-rest).
- **Forget-by-reprojection** — forgetting that works by construction: delete the source op, re-project, the fact is gone — a deterministic alternative to unsolved machine unlearning for the symbolic layer (`reprojection-forget/` ; cross-process

determinism audited). Closest prior art: verifiable federated unlearning (arXiv 2510.00833), ZK-APEX (2512.09953) — model-side, statistical; this is ledger-side and exact.

- **DNG (real-DP collective)** — collective learning under a *real, computational* differential-privacy bound with **no trusted curator for the noise**: distributed discrete-Gaussian shares, a public-coin proof each device sampled its noise, and a signed anti-rollback RDP odometer (`real-dp-collective/`).

Emergent capabilities (compositions nobody ships)

- **Grounding Receipt** — prove an answer used *exactly* a committed subset of your memory and **nothing off-memory** (the closure property), value-hiding (`grounding-receipt/` ; 17/17 + 20-attack red-team).
- **Stapled Transcript (FM-045)** — staples a compute-side proof-of-inference receipt to the Grounding Receipt via an ordered Merkle root over the used entries' commitments: *"the model computed Y from exactly this consented, user-owned subset — and the consumed context had no off-memory entry."* This is the **missing input-provenance half of every proof-of-inference receipt** (`stapled-transcript/` ; 51 independent attacks resisted across three suites). Closest prior art: c0mpute/Shard and Lagrange DeepProve produce the compute half but structurally cannot produce the input half; Anuma *logs* permissions rather than proving grounding.
- **PAIR — Provable AI Interaction Record (FM-048)** — one full turn made third-party-auditable offline and value-blind: *what memory was used* (grounding), *where it ran with nothing off-memory* (staple), *what was provably never supplied* (Empty Witness), the used-set and never-supplied-set proven **disjoint**, all legs sharing one freshness pin, owner-signed (`pair/` ; 28 core checks + 13 adversary checks, all green). Court-handable full-turn data-governance.
- **Memory Immunity Receipt** — an anti-poisoning **proof of exclusion**: the answer's grounding set was reprojected from *exactly* the enrolled-author-signed ops, and every unsigned/forged/non-enrolled op is in a re-derived quarantine set and contributed nothing — value-hiding, content-pinned roster (`memory-immunity/` ; core + three red-teams green, the honesty/overclaim suite finds 0 breaks and 0 overclaims). Closest threat model: OWASP ASI06 / MINJA / MemoryGraft; closest provenance prior art: Provenance Gates (2605.13471), MemLineage (2605.14421) — admission/labelling without an offline value-blind exclusion proof.
- **Provable Consolidation (FM-051)** — "provable sleep": a durable summary is the residue of *exactly* a named set of ephemeral sessions **and** each ephemeral is now information-theoretically unrecoverable (random per-ephemeral key destroyed; ciphertext no longer decrypts) — two-sided set identity, owner-signed custody, checkable offline (`consolidation/` ; 15 core checks + 11 adversary attacks resisted). Stricter than a tombstone receipt: destruction is observed, not asserted (cf. eprint 2014/364).
- **Discovery Shield** — one verifier proving *both* "no byte released" **and** "no consent granted" for the same (X, S), surviving a decryptor (`discovery-shield/` ; 11/11 + decryptor red-team).
- **Soul Will** — consent-bound, scope-partitioned, quorum-gated memory inheritance (`soul-will/`).
- **Graded Consent Spend-Trace** — a verifiable consent/usage budget; an over-tier / foreign-purpose / over-cap / sub-quorum release cannot produce a valid receipt (passed after two adversarial fix rounds).
- **Decentralized Oblivious Recall (FM-042)** — recall whose access pattern reveals neither the queried index **nor the node count**, bound by a signed swarm-membership commitment (`oblivious-recall/` ; order + node-count bound; distinguisher advantage 0.0003).

Your six questions, answered

- **Novel?** Yes — by *fusion and application*, honestly. No new primitive exists on the frontier and the program proved it; three louder primitive-grade claims (Entangled Memory, Semantic Cones, holographic privacy) were *falsified to features* under adversarial fire and are not claimed.
- **Never been done?** The intersection *user-owned + portable + private + verifiable + decentralized* is occupied by **no shipped, funded product**; and a constructive *proof of a negative* about AI custody (Empty Witness, PAIR's never-supplied leg) had no prior art.

- **Useful?** Legal and compliance (Empty Witness, Discovery Shield, PAIR, forget certificate), anti-poisoning attestation (Memory Immunity), inheritance (Soul Will), enterprise audit, individual sovereignty.
- **Better than anything else?** It is the only design where the **user holds the keys**, memory is **never whole**, forgetting is **provable**, and a full turn is **portable + offline-verifiable** — versus Mem0/Letta/Zep (app-owned), QVAC (runtime, no accountability ledger), Anuma (encrypted vault that *logs* rather than *proves*), Filecoin/Arweave (token-coupled blobs), Portable Agent Memory 2605.11032 (readable blob, held whole).
- **Good for decentralized AI?** No operator/account/server owns it; survives company death; P2P across your devices; un-seizable **and** accountable — which is what lets AI operate in regulated and adversarial contexts.
- **The SanDisk of crypto + AI memory?** Own the **substrate** — where and how memory lives — a portable, vendor-neutral object any model mounts, encrypted to your keys: the "SD-card slot every AI plugs into." "Crypto" means cryptography + decentralized storage. **No token.**

Honest limits, and the honest negatives

- **Not confidentiality-during-compute.** At recall the projection is plaintext in RAM on one device for that turn (the L3 locality gap). No FHE.
- **Non-materialization is protocol-level.** Op-log ciphertext is durable; the RAM projection is swap-spillable.
- **Forget = attestation + information-theoretic unrecoverability**, not physical destruction on replicas you do not control.
- **Offline proofs need a freshness anchor** — a pinned epoch, nonce, or witness.
- **Not deployment-ready without an external audit.** The proven receipts now ship as `packages/flowmemory-receipts` (the reusable library) and `packages/sovereign-memory` (the product) — production-shaped, distinct from the git-ignored `research/prototypes/` scratch. But the hand-rolled BigInt crypto is **not constant-time and not professionally audited** — not for deployment without an independent cryptographic audit.
- **DP release needs an honest or threshold aggregator** (the *noise generation* does not).
- **The right-half proofs bind custody, not cognition.** Grounding, Staple, PAIR, and Immunity prove what the custody layer passed to the model and that it carried valid authorship/closure — never what a remote model attended to, ignored, or hallucinated. Live-query prompt injection is out of scope.
- **Recorded negatives (kept, not hidden).** USPIR sublinear DPF recall is real but **parked** — it has no live-execution obliviousness witness. FM-047 (witnessed-sublinear recall) is a **wall that held**: a self-issued offline witness is fabricable, so plain recomputation cannot bind a witness to runtime behavior without a logged interactive protocol or a SNARK/TEE. Memory Immunity's completeness over *omitted* enrolled ops is an out-of-band pin, not a receipt-internal guarantee — and the verifier is honest about it.

The deepest finding of the build: **the adversarial build, not the literature search, is what separates a real result from a plausible one.** Every break across the program lived in composition glue or a fail-open default — never in a red-teamed primitive.

The two halves of no-off-switch AI

Decentralized AI splits into two provable halves. The **compute half** is mature and fast — c0mpute/Shard, VeriLLM, Lagrange DeepProve, Gensyn Verde, PoSP — but every one of those receipts hides data from the *verifier*, never from the *prover*: the node still holds the input in plaintext. The **memory half** has been the missing operand. Sovereign Memory is that half — and the Stapled Transcript is the literal seam: a compute receipt stapled to a value-blind input-provenance proof, with PAIR turning a single turn into a court-handable record. Together they make an AI that no single operator can switch off **and** that can still prove, offline, exactly what it was given, what it used, and what it was never even told.

The pitch. *Sovereign Memory — the SanDisk of AI memory. Your AI's memory, owned by you: un-seizable, recomputed-never-stored, cleanly forgettable, and provable — including proof of what your AI was never even told, what it actually used this turn, and that nothing poisoned got in. No operator. No token. Delete the company and it keeps working.*

Novelty and differentiation

> **Honesty contract (binding on this document).** Signatures and hashes prove *custody, integrity, ordering, consent, minimization, grounding, non-supply, exclusion, and forgetting* — they do **not** prove confidentiality-during-compute, are **not** FHE, do **not** prove what a model internally attended to, and do **not** prove semantic truth. On-device privacy is *locality*, not homomorphic compute; we never claim a server or model "cannot read" data on a path where it can. Offline proofs need a freshness anchor (a pinned epoch, nonce, or witness). The differential-privacy *release* still needs an honest or threshold aggregator. **The novelty of this system is fusion-grade and application-grade — not a new cryptographic primitive.** The program hunted for a new primitive across ten frontiers and proved the frontier is literature-occupied; we say so plainly below. Every novelty claim is named next to its closest prior art. Every capability claim is backed by a runnable prototype plus an independent adversarial suite that exits zero, or it is labelled a stated limit. There is **no token.**

This document answers three questions with citations to runnable prototypes: (1) **why it is novel**, (2) **why it has never been done**, and (3) **how it differs from everything else** — including an explicit list of what we do *not* claim, and the negative results we kept rather than hid.

---

1. Why it is novel

1.1 The honest framing first

Sovereign Memory is **novel by fusion and by application, not by inventing a new primitive.** This is a finding the build program earned, not a hedge. Iteration 9 of the research program (`research/AI_MEMORY_LAYER_DOSSIER.md`, iter 9) deliberately hunted *past* composition-grade novelty for a new primitive across ten experimental frontiers — hyperdimensional/VSA memory, PUF/physical binding, portable KV-cache, VDF time-lock, regenerating codes, neuromorphic sparse distributed memory, zero-knowledge-over-memory, leakage odometers, lossy/importance coding, and the substrate-ownership move itself. The result, recorded honestly: **zero candidates cleared the "truly-new primitive nobody publishes" bar.** Every exotic primitive already exists in the literature — puncturable encryption, honey encryption (Vanish/Ephemerizer), VDF/time-lock (LCS35, tlock, drand), PUF sealing (DPReF/PReF), CHURP/MPSS proactive resharing, OCELOT leakage accounting, FSS-PIR, behavioral fuzzy extractors.

What is genuinely unshipped is the **specific composition** and the **capabilities that composition unlocks**. Three louder ideas that *claimed* primitive-grade novelty — Entangled Memory, Semantic Cones, and holographic privacy — were falsified to features under adversarial fire (§4). That discipline is the credibility: the surviving claims survived a program designed to kill them. The recurring meta-lesson is stated here once and demonstrated throughout: **every break across the program lived in composition glue or a fail-open default — never in a red-teamed cryptographic core.** The adversarial build, not the literature search, is what separated a real result from a plausible one.

1.2 The sharpest novel claims

Each claim below is an expert-surprise, each is backed by a runnable prototype plus an independent adversarial test, and each is stated next to its closest named prior art and the exact wedge that distinguishes it. Where a claim is a *fusion* rather than a *new primitive*, it is labelled as such.

Claim 1 — Read-consent is unforgeable even by a party that already decrypted the data. *(Fusion.)* TRM's two-secret spine separates *confidentiality-at-rest* (a GF(256) Shamir split of the recall key `K_rec`) from *read-consent* (a P-256 threshold-Schnorr signature whose group scalar `s_grp` is **never reconstructed**). The falsifying test T2 (`research/prototypes/trm/`, suite `trm.test.mjs` build-green 22/22; `trm.attack.test.mjs` held 17 independent forgeries) proves the surprising property: **a coordinator holding `K_rec` *and* the plaintext `M` still cannot produce a signature that verifies under the consent verifier without a real `k_a`-of-`n` aggregate.** Knowing the plaintext is *not* consent. The verifier pins each share-public `P_i` to the enrolled roster and checks every partial `z_i·G == R_i + c·λ_i·P_i`, so a lone aggregate or a back-solved partial is rejected. *Closest prior art:* threshold-Schnorr / FROST (Komlo–Goldberg 2020) and Shamir secret sharing (1979) — both off-the-shelf, both

used here unmodified. *The wedge:* running *two independent threshold splits over one device roster* so that "can decrypt" and "did the owner's devices consent to a read" become **cryptographically independent, separately-auditable events.** No memory product treats consent-to-read as a forge-resistant artifact distinct from the decryption key.

Claim 2 — A recomputed-never-stored memory index whose key requires a live physical-token quorum. *(Fusion.)* QRSM (`research/prototypes/qrsm/`, suite `qrsm.test.mjs`) never writes the queryable projection (embeddings, graph, summaries) to disk. It rebuilds the projection in RAM by replaying the encrypted op-log; the replay key is reconstructed live from a `k`-of-`n` threshold-PRF over **off-device** FIDO2/WebAuthn-PRF tokens. The independently red-teamed delta (iter 14): the device at rest persists only public offsets `off_i = share_i XOR HMAC(tokenSecret_i, salt)` with `tokenSecret_i` *off-device*; an after-teardown at-rest scan finds **exactly two files** (ciphertext op-log + public enrollment), and seizing the powered-off devices yields only ciphertext and an index that was never written. The falsifier built the TRM "shares-at-rest" strawman, confirmed it collapses under seizure, and confirmed QRSM does *not*. *Closest prior art:* event-sourced / projection-not-store materialization (standard incremental view maintenance), threshold-PRF over tokens (Bitwarden / `age`-style PRF-to-AES). Both mature. *The wedge:* fusing them so the precondition for the index to *exist at all* is a live physical-token quorum — turning a subpoena from "disclose a stored or decryptable artifact" into "physically produce tokens for an index that was never persisted." Unpublished and unshipped as a fusion. (Honest limit, §4: at-rest only; the RAM projection is plaintext-while-quorum-present and swap-spillable.)

Claim 3 — Forgetting by reprojection: a deterministic alternative to machine unlearning. *(Application of a known pattern to a new layer.)* Because the symbolic memory is a pure, content-addressed *projection* of the op-log, **forgetting = delete the source op + re-project.** The fact is gone *by construction* — none of the "did the weights actually forget?" uncertainty that makes machine unlearning unsolved. The prototype (`research/prototypes/reprojection-forget/`, suite `reproj.test.mjs` 26/26 plus a 16-check independent attack) proves a forgotten fact is provably absent from the re-projected index/facts/summary, a kept fact survives surgically, and same-set re-projection is byte-identical across processes. *Closest prior art:* machine unlearning and verifiable federated unlearning (arXiv 2510.00833), ZK-APEX (arXiv 2512.09953) — all operate *model-side*, on weights, and inherit the open problem of proving the model truly forgot. *The wedge:* moving forgetting to the *deterministic symbolic projection layer*, where deletion is exact, not statistical. (Honest boundary, §4: for a trained neural adapter, cross-hardware bit-for-bit regeneration is impossible — root cause = parallel-reduction order over non-associative float addition — so the adapter is handled by *content-addressed provenance*, "derived from exactly this op-set, which excludes X," not by regeneration. Caches are the implementation hazard: re-projection must be from-scratch, never incremental.)

Claim 4 — A constructive cryptographic proof that your AI was never told X (the Empty Witness). *(Capability-level packaging of one proven layer.)* Proof-of-non-supply (`research/prototypes/proof-of-non-supply/`, suite `nonsupply.test.mjs`, hardened with the V4 query-binding fix and re-attacked) lets a defendant hand a relying party one offline file plus a public key, and a stateless verifier answers a hard YES/NO to *"was fact X ever released into AI session S?"* For a never-supplied fact it returns a verified non-supply result — a portable, per-session, query-bound, operator-less **proof of a negative.** Soundness rests on a non-membership proof over a signed sparse-Merkle release set plus an RFC-6962 consistency proof that the log only grew. *Closest prior art:* SCITT refusal-events and CAP-SRP/VeritasChain (prove a *logged refusal*); ZKPROV (arXiv 2506.20915, proves *positive* training provenance). Each proves the opposite or adjacent direction and *admits it cannot prove what was never logged. The wedge:* the constructive negative — what the AI was *never even told* — which the legal baseline ("absence of evidence is not evidence of absence") says you cannot produce. (Honest limit, §4: proves custody-side non-release, not model conditioning; the verifier must pin its latest checkpoint for freshness.)

Claim 5 — Collective learning with a real, verifiable DP bound and no trusted curator for the noise. *(Fusion.)* FlowMemory-DNG (`research/prototypes/real-dp-collective/`, suites `dng.dp.test.mjs`, `dng.zk.test.mjs`, 20/20) makes ε a real *computational* bound rather than an accounting entry. Each device adds a calibrated *share* of integer-exact discrete-Gaussian noise (Canonne–Kairouz–Steinke) under double-masked secure aggregation; closure-under-summation makes the aggregate provably noised; a public-coin cut-and-choose proof binds each device to having actually sampled its noise; and a user-held, signed, hash-chained RDP-filter odometer (Lécuyer-style) enforces valid composition. Measured: ε sweeps monotonically with σ (1.23→0.15), the membership adversary's advantage (0.038) sits far below the bound (0.199), distributed noise approximates central-curator noise, and a zero-noise free-rider is *rejected* by the sampling proof. This is a strict advance over the earlier accounting-only design (FM-031), whose verifier was byte-invariant to whether noise was actually applied. *Closest prior art:* verifiable DP broadly (VERIDP, VDDP), distributed discrete-Gaussian DP (Kairouz et al.). Real and named. *The wedge:* the *fusion* —

a user-held, signed, anti-rollback RDP odometer composed with public-coin-verified noise *sampling* and distributed noise generation, specifically for user-owned AI memory — which could not be found combined anywhere. (Honest limit, §4: the *release* still needs an honest or threshold aggregator because the cut-and-choose publishes the noise count in cleartext; the *noise generation* does not need a trusted curator; the bound is computational, not information-theoretic, per Biswas–Cormode / Haitner–Omri.)

Claim 6 — The Stapled Transcript: the missing input-provenance half of every proof-of-inference receipt. *(Fusion; the answer to the compute-receipt sector.)* The compute sector (c0mpute/Shard, VeriLLM, Gensyn Verde, Hyperbolic PoSP, Lagrange DeepProve-1) produces a *compute-side* receipt — it proves a model ran a computation correctly. None of those receipts can prove *what input the computation consumed was the user's consented, on-memory context and nothing else*; the receipt hides data from the *verifier*, never the *prover*, and split inference forces every compute node to decrypt. The Stapled Transcript (`research/prototypes/stapled-transcript/`) staples a compute-class proof-of-inference receipt (signed under a separate compute key) to the **unmodified** Grounding Receipt via an **ordered Merkle root over the used entries' commitments**: the verifier recomputes that root over the grounding receipt's own membership-verified closure set and fails closed unless the compute node's `context_hash` matches. An off-memory entry has no commitment in the closure set, so including it changes the leaves and the root differs (rejected); a dropped entry changes the leaf count (rejected); a reorder changes the ordered root (rejected) — all verifier-checkable and value-blind. Proven: 51 independent attacks resisted across three suites (13 builder/attack, 22 red-team, 16 cross-bind adversary, all exit zero; including encoding-malleability, the redundant-second-signature kill-condition, and the Bitcoin CVE-2012-2459 duplicate-leaf), with the underlying Grounding Receipt byte-unchanged and still green (17/17 + 20-attack). *Closest prior art:* CommitLLM (commit-and-audit binds checkpoint+policy+answer, model-side); Lagrange DeepProve / Gensyn Verde / Hyperbolic PoSP (compute-correctness receipts); Anuma (an encrypted cross-model vault that *logs* permissions). All produce the compute half or a permission log; **none produce the value-blind input-provenance half bound to a user-owned memory subset.** *The wedge:* the verifier-checkable, value-blind cross-bind between an independent compute receipt and a user-owned grounding closure — the right operand the whole compute sector structurally cannot produce. (Honest limit, §4: binds committed segments + order, **not** interstitial glue bytes between segments — a named ZK-opening research upgrade — and inherits the compute receipt's left-half trust model; it is not confidentiality-during-compute.)

Claim 7 — The Provable AI Interaction Record (PAIR): a full turn made court-handable, offline, value-blind. *(Fusion of four proven legs.)* PAIR (`research/prototypes/pair/`) renders one AI turn third-party-auditable offline without exposing a single plaintext value, salt, or the operator's model. One owner-signed record makes the complete data-governance of an interaction checkable: *what memory was used* (the Grounding leg's membership + closure), *where it ran with nothing off-memory* (the Stapled compute cross-bind), and *what was provably never supplied* (the Empty Witness non-membership leg) — and it adds three strictly-stricter outer gates: the used-address set and the never-supplied-address set must be **disjoint** (a fact cannot be both used-this-turn and never-supplied), all legs must share **one freshness/epoch pin**, and the owner signs the whole record. The three legs are imported byte-for-byte and never modified; a PAIR is accepted only if every leg verifies in full *and* the outer gates hold. Proven: 28 core checks + 13 adversary checks, all green, including the cross-leg session-splice defense (a custodian that did release X cannot satisfy non-supply against a decoy session, because the verifier independently pins both the session id and the release root out-of-band). *Closest prior art:* Anuma (encrypted cross-model vault with on-chain *logging* of permissions, not an offline proof of grounding + non-supply); SCITT/CAP-SRP transparency logs (logged events, not a value-blind full-turn record). Logging is not proving. *The wedge:* the disjointness + single-pin + owner-signature composition over three independently-proven legs yields an emergent property no single leg has — a portable, offline, value-blind *full-turn* governance record. (Honest limit, §4: inherits every leg's honest limit; binds custody, not model cognition; the verifier must obtain the session and release-root pins from a trusted anchor, not from the record under audit.)

Claim 8 — The Memory Immunity Receipt: an anti-poisoning proof of exclusion. *(Fusion; dual of the Grounding Receipt.)* The Grounding Receipt proves the positive closure "answer = union of exactly these in-set entries"; it is silent on *who authored those entries*. Memory poisoning (OWASP ASI06, MINJA, MemoryGraft) is precisely about an op that *looks* like memory but was not authored by an enrolled principal. The Memory Immunity Receipt (`research/prototypes/memory-immunity/`) adds, per op, an authorship predicate (a valid P-256 signature by a roster-enrolled author over the canonical op body — *not* agent self-signing) and a **negative exclusion proof**: the candidate op-log is partitioned into ADMITTED (enrolled-signed) and a quarantine of REJECTED (unsigned/forged/non-enrolled) ops; the symbolic projection is a deterministic pure function of ADMITTED only; the

verifier **re-partitions from the raw candidate log itself**, re-projects, and checks the answer's grounding set closes over the reprojection that excludes the quarantine. A forged op forced into the prompt cannot be in ADMITTED (no enrolled signature), so closure breaks and the receipt is rejected. Proven value-blind, with the freshness pin being the **roster content hash** (not merely an epoch integer — pinning the epoch alone is insufficient, because the owner can sign a second roster at the same epoch that enrolls a poisoner). Proven: 8 build checks + 19 attack + 16 soundness + 13 roster red-team, all 0 breaks / 0 overclaims. *Closest prior art:* Provenance Gates (arXiv 2605.13471) and MemLineage (arXiv 2605.14421) — signed-op admission and per-principal lineage labelling. Both *admit/label* at write time; **neither emits an offline, value-blind proof of exclusion bound to a specific answer** that a third party re-derives. *The wedge:* the re-derived ADMITTED/quarantine partition + answer-bound closure makes poison *attributable and provably excluded*, offline, without a TEE and without self-signing. (Honest limit, §4: proves *attributability*, not truthfulness — an enrolled author can sign a false fact; covers poison arriving as a stored op, not live-query prompt injection; completeness over *omitted* enrolled ops is an out-of-band pin, not a receipt-internal guarantee, and the verifier says so.)

---

2. Why it has never been done

2.1 The unoccupied intersection

The strategic whitespace, verified across the research program as occupied by **no shipped, funded product**, is the *full* intersection of five properties at once:

> **user-owned + portable across models + private (on-device / E2E) + verifiable (provenance) + decentralized (P2P across your own devices).**

Each property is individually available somewhere. The intersection is empty because the dominant business models pull *against* it: the memory is the moat, so whoever builds it wants to *hold* it, not hand the keys to the user. Mem0's own framing — "memory is becoming a key moat now that LLMs are getting commoditized" — is precisely the incentive to keep memory operator-held. Owning the substrate, by contrast, requires giving up custody, which no incumbent will do voluntarily.

A second whitespace sits on the *accountability* axis: a constructive **proof of a negative** about AI custody — what was never supplied (Empty Witness), and a full turn's worth of it (PAIR's never-supplied leg). The legal and transparency-log baselines prove *positive* or *logged* events and openly admit they cannot prove what was never logged.

2.2 What each competitor lacks (the missing axis)

Player	Has	Missing axis (why it does not occupy the intersection)
Mem0 / Letta / Zep / Cognee / Supermemory	Strong AI-memory DBs, lifecycle, retrieval	**User-owned + decentralized.** Operator holds the keys and the data; centralized or app-owned; no provable forgetting; no cross-vendor portability of *custody*
Tether QVAC	Local-first, P2P, private agent runtime	**A content-addressed, cross-model memory *ledger*** with consent ≠ decryption and proof-of-non-supply. QVAC ships a runtime + session-memory compression, not a sovereign accountability spine
Anuma	Encrypted cross-model memory vault	**Proof, not logging.** Anuma *logs* permissions on-chain; it does not emit an offline, value-blind proof of grounding + non-supply (the PAIR / Stapled Transcript wedge)
Plurality Network	Context-injection UX across apps	**A sovereign *store*.** It is injection, not custody; no cryptographic accountability
Portable Agent Memory (arXiv 2605.11032)	Portable, content-addressed, Merkle, Ed25519, cross-model	**Memory that is never whole + recomputed-never-stored.** Their blob is *readable and held whole*; reading is not an audited quorum event
c0mpute / Shard, Lagrange DeepProve, Gensyn Verde, Hyperbolic PoSP	Verifiable *compute*-side proof-of-inference	**The input-provenance half.** The receipt hides data from the verifier, never the prover; split inference forces every node to decrypt; no proof the consumed context was the user's consented on-memory subset (the Stapled Transcript wedge)
Filecoin / Arweave / Walrus	Decentralized blob storage at scale	**A queryable, private, forgettable AI *memory* — and no token.** They store raw bytes (often token-subsidized; Filecoin ~36% utilized, hunting real customers), not a private forgettable index
SCITT refusal-events / ZKPROV	Proof of logged refusals / positive provenance	**Proof of what the AI was *never told*** (the opposite, unoccupied direction)
Provenance Gates (2605.13471) / MemLineage (2605.14421)	Signed-op admission, per-principal lineage labelling	**An offline, value-blind, answer-bound proof of *exclusion*** (the Memory Immunity wedge) — they admit/label at write time, they do not prove poison was excluded

The pattern: every player is missing *at least one* of the five axes, and the one they are missing is usually the one their business model depends on them missing (custody) or the one nobody else thought to build (the constructive negative; the input-provenance half).

---

3. How it is different from anything else

3.1 The core architectural difference

The difference is not a feature; it is **where and how the memory physically lives.** This is the SanDisk analogy made literal: SanDisk's moat was never a clever feature but a *structural property of the medium a competitor cannot retrofit*. Sovereign Memory's equivalent structural property is **memory whose readability is a function of a live quorum of *your* devices.** A competitor can copy a receipt format in an afternoon; they cannot retroactively make their operator-held database *not* operator-held without rebuilding their trust model from the substrate up.

The honest distinction we always state: SanDisk owns physical silicon; we own a *protocol and architecture*, which a competent team could re-implement in months once described. The moat is therefore **proven fusion + first-mover + an un-weakenable honesty boundary**, not a fab. We say this rather than pretend otherwise.

3.2 Head-to-head differentiation

vs Mem0 / Letta / Zep / Cognee / Supermemory. These are the crowded center: the database the AI company runs, with no user keys. The trust model is *opposite* to ours. We give the *user* the keys, the data lives across *your* devices, forgetting is provable (forget-by-reprojection, §1.2 Claim 3), and custody is portable across models. They optimize memory *quality* under operator custody; we change *who holds it*.

vs Tether QVAC. The closest in spirit — local-first, P2P, private AI — and therefore worth being precise about. QVAC is an agent *runtime* plus session-memory *compression*. It is orthogonal to a **content-addressed, threshold-resident memory ledger** where consent ≠ decryption (TRM T2) and where you can prove non-supply. QVAC is a place to run the agent privately; Sovereign Memory is the accountable, un-seizable substrate the agent's long-term memory rides on. They compose more than they compete.

vs Anuma. Anuma is the closest live competitor on the *user-owned encrypted memory* axis: an encrypted cross-model vault. The difference is precise: Anuma **logs** what happened (on-chain permission records); Sovereign Memory **proves** it offline and value-blind. A log says "this permission was recorded"; the Grounding Receipt, the Stapled Transcript, and PAIR say "here is a verifier-checkable artifact that the model consumed exactly this consented subset and nothing off-memory, and that fact X was never supplied" — without an operator, without a chain, and without revealing a value. Logging trusts the logger; proving does not.

vs the compute-receipt sector (c0mpute/Shard, Lagrange DeepProve, Gensyn Verde, Hyperbolic PoSP). This sector is fast and verifiable-for-correctness, but every proof requires the prover to hold the input in plaintext — the receipt hides data from the *verifier*, never the *prover* — and split inference forces every node to decrypt. They produce the compute half of a proof-of-inference receipt; they structurally cannot produce the **input-provenance half** that binds the consumed context to a user-owned, consented, value-blind memory subset. The Stapled Transcript (§1.2 Claim 6) is exactly that missing operand; we compose with their work rather than compete with it.

vs Plurality Network. Plurality is a centralized context-*injection* UX. It decides what to inject into a model's context. We are a sovereign *store* with cryptographic accountability for what crossed the boundary. Injection without custody cannot produce the Empty Witness; you cannot prove a negative about a context you do not own and do not log under your own keys.

vs Portable Agent Memory (arXiv 2605.11032). The closest *named* artifact in the literature — a readable, content-addressed, Merkle-and-Ed25519, cross-model portable memory blob. It legitimately refutes any generic "portable provenance memory" novelty claim, and we credit it as such. The difference is precise and testable: **their memory is a readable encrypted blob held *whole* somewhere; ours is *never whole* (TRM), and "port to plaintext" is an *audited quorum event*, not a read** — and the queryable index is *recomputed-never-stored* (QRSM), so there is no whole blob to port at rest. Their SDK can hand you the memory; ours can prove your devices consented before anyone read it, and can prove the index was never written.

vs Provenance Gates (2605.13471) / MemLineage (2605.14421). These are the closest prior art to anti-poisoning: signed-op admission and per-principal lineage labelling. They *gate or label at write time*. The Memory Immunity Receipt (§1.2 Claim 8) is the dual that they lack: an **offline, value-blind, answer-bound proof of exclusion** in which the verifier re-derives the ADMITTED/quarantine partition itself, rather than trusting an admission decision. They make poison *labellable*; we make it *provably excluded* from a specific answer.

vs Filecoin / Arweave / Walrus. These are decentralized *blob storage* layers, frequently token-subsidized. We are not competing on raw byte durability; we are a *queryable, private, forgettable AI memory* with **no token.** They answer "where do the bytes live cheaply and durably"; we answer "how does a private, forgettable, consent-gated *index of you* live across your own devices and prove what it did." One could even use such a layer for ciphertext durability beneath us; the value we add is everything above the bytes.

vs SCITT refusal-events / ZKPROV. SCITT-style transparency logs and CAP-SRP prove what an AI *refused* — a logged positive event. ZKPROV (arXiv 2506.20915) proves *positive* training provenance. Both are honest about a shared limit: they cannot prove what was never logged. The Empty Witness occupies exactly that vacated direction — a constructive proof of the *negative*, the thing they admit they cannot do. We are not a better refusal log; we are the opposite-facing proof.

3.3 The emergent capabilities that compound the difference

The layers compose into capabilities nobody ships, which widen the gap beyond any single primitive. Each is backed by a green prototype suite:

- **The Empty Witness** (§1.2 Claim 4) — a court-handable proof of a negative behind a UI; the wedge product (`proof-of-non-supply/`).
- **The Stapled Transcript** (§1.2 Claim 6) — the input-provenance half the compute sector cannot produce (`stapled-transcript/` ; 51 attacks resisted across three suites).
- **PAIR** (§1.2 Claim 7) — a full turn's data-governance, offline and value-blind, court-handable (`pair/` ; 28 + 13 checks green).
- **Memory Immunity** (§1.2 Claim 8) — an answer-bound, value-blind proof of poison exclusion (`memory-immunity/` ; build + three red-teams green, 0 breaks / 0 overclaims).
- **Provable Consolidation** ("provable sleep") — a durable summary proven to be the residue of *exactly* a named set of ephemeral sessions, every one of which is now information-theoretically unrecoverable because its random per-ephemeral key was destroyed and its ciphertext no longer decrypts; two-sided set identity (lineage-in == destroyed-out), owner-signed custody, checkable offline (`consolidation/` ; 15 core + 11 adversary checks green). Stricter than a tombstone receipt: a "destroyed" op that still decrypts is not destroyed, so destruction is *observed*, not asserted (cf. eprint 2014/364, publicly-verifiable deletion via key destruction).
- **Discovery Shield** — one verifier proving *both* "no byte of X was released into session S" *and* "no quorum consent to release X was granted," over the same (X, S), a two-channel negative proof that **survives a decryptor**: a party who reconstructed `K_rec` and decrypted X can forge *neither* leg (`discovery-shield/` ; 11/11 + decryptor red-team holds). Neither layer alone yields "no byte left *and* no authorization given."
- **Decentralized Oblivious Recall (FM-042)** — recall from a byzantine swarm whose on-wire access pattern reveals neither the queried index *nor the node count*; a signed obliviousness witness binds the access set to a public per-epoch schedule and the node order to a P-256-signed swarm-membership commitment (`oblivious-recall/` ; order + node-count bound; trained-distinguisher advantage 0.0003). Honest cost stated, not hidden: full-batch obliviousness has catalog-size blowup, and the windowed variant is *not* oblivious across sessions (so we do not ship it as such).
- **Soul Will** — consent-bound, scope-partitioned, quorum-gated memory inheritance with provable burn-below-recovery (`soul-will/`).
- **Graded Consent Spend-Trace** — a verifiable consent/usage budget in which an over-tier, foreign-purpose, over-cap, or sub-quorum release cannot produce a valid receipt; proven after two adversarial fix rounds (overflow guard + grade-binding gate).

---

4. What we do **not** claim (the falsified ideas and the kept negatives)

The credibility of the surviving claims comes from publicly recording the ones that died. **Three louder ideas were *falsified to features*** under independent adversarial testing and are **not** part of the product claim:

1. **Entangled Memory ("Mnemonic Entanglement," FM-025).** The pitch was a memory corpus and a continually-trained personal adapter cryptographically *co-keyed*, so a stolen memory dump is useless without the live adapter and vice versa. Falsified (iter 10, `research/prototypes/entangled-memory/`): "mutual uselessness" is **one-directional** (a stolen adapter is a fully functional model; only memory *plaintext* is gated, and that gate is the quorum secret, not the corpus); the "live-presence witness" is cryptographically **indistinguishable from shares-at-rest** (a thief with the powered-off devices recombines offline — exactly TRM's existing at-rest property, not new); and the "saliency challenge" is a deterministic HMAC of the adapter, **not an independent third secret.** Only a narrow, conditional "forward-dark on adapter ratchet" survived. It collapses to "TRM + an optional forward-secret adapter-hash binding" — a minor feature.
2. **Semantic Cones (FM-026).** The pitch was keys scoped to *regions of embedding space* — decrypt "work" memories but *provably not* "health." Falsified (iter 11, `research/prototypes/semantic-cone/`): the key layer is just per-label HKDF (identical whether the label is a human tag or a cluster id — "semantic" adds nothing cryptographic), and the real boundary is **non-cryptographic and leaks structurally** — confidentiality reduces to "is the embedding binning correct?", a

geometry/ML question. Every cone pair has a non-empty leak band; a real "medical leave for chemotherapy" memory mis-binned to WORK and a WORK grant *decrypted it.* The honest residual is a useful sharing feature with the caveat that boundary memories can be miscategorized — not provable selective disclosure by meaning.

3. **Holographic privacy (FM-030, keyed holographic memory).** The pitch was memory as a superposition hypervector with a key-free privacy guarantee. The *forget-by-algebra* mechanism is genuine and surgical, but the privacy headline was **falsified**: the energy statistic is a key-free membership oracle (AUC 0.84 at N=16) and the vector norm leaks the exact item count. **Superposition is not encryption.** It remains research-grade and is not claimed as a privacy primitive.

Two negative results on the sublinear-recall frontier are kept on the record because they map the next hard wall:

4. **USPIR — sublinear DPF oblivious recall (FM-046).** A two-party BGI distributed-point-function read over two user-owned ciphertext replicas gives genuinely sublinear recall (`research/prototypes/uspir-dpf/` ; single-key distinguisher advantage ~0.009) with no FHE, TEE, or trusted party. It is **parked**, not shipped, for honest reasons: it is static single-writer only (DPF-PIR needs byte-identical replicas, which a divergent multi-writer CRDT breaks), and — decisively — it has **no recomputable obliviousness witness** on the DPF path. It is real; it is not a claim.

5. **FM-047 — witnessed sublinear recall (a wall that held).** The attempt to give the sublinear DPF read the obliviousness witness USPIR lacked was **broken by the independent red team**: the witness is *self-issued and unbound to the live query* — because the reply is recomputable from public blocks plus a self-chosen key, a node can fabricate a consistent `(key, reply)` pair entirely offline (a node that served zero queries passes). **Plain recomputation cannot bind a self-issued offline witness to runtime behavior** without a logged interactive protocol or a SNARK/TEE. We state this as a wall, not paper over it.

These join the standing honest limits that travel with every claim: **not confidentiality-during-compute** (at recall the projection is plaintext in RAM on one device for that turn — no FHE); **non-materialization is protocol-level** (op-log ciphertext is durable, the RAM projection is swap-spillable); **forget = attestation + information-theoretic unrecoverability**, not physical destruction on replicas you do not control; **the right-half proofs bind custody, not cognition** (Grounding, Staple, PAIR, and Immunity prove what the custody layer passed to the model and that it carried valid authorship/closure — never what a remote model attended to, ignored, or hallucinated; live-query prompt injection is out of scope); **offline proofs need a freshness anchor** (a pinned epoch, nonce, or witness); and the **DP release needs an honest or threshold aggregator** (the noise *generation* does not).

The deepest finding, stated once more because it is the credibility: **every break across the entire program lived in composition glue or a fail-open default — never in a red-teamed cryptographic core.** That is why fixes only ever made a verifier stricter or fail-closed, and never weakened a check to pass.

---

5. The novelty claim, in one paragraph

Sovereign Memory does not introduce a new cryptographic primitive — the program proved, adversarially, that the primitive frontier is literature-occupied, and falsified its own three loudest primitive-grade claims to features. What is genuinely unshipped is the **fusion**: a two-secret threshold spine where read-consent is unforgeable even by a decryptor (TRM T2); a recomputed-never-stored index gated on a live physical-token quorum (QRSM); forgetting that works by reprojection rather than unlearning (FM-029); a constructive proof of what the AI was never told (the Empty Witness); collective learning under a real, verifiable DP bound with no trusted curator for the noise (DNG); the input-provenance half of every proof-of-inference receipt, stapled value-blind to a user-owned grounding closure (Stapled Transcript); a full turn made court-handable, offline, and value-blind (PAIR); and an answer-bound, value-blind proof that poison was excluded (Memory Immunity) — all composed over a user-owned, decentralized, model-agnostic substrate, with the negative results (parked USPIR, the FM-047 wall) kept on the record. No shipped, funded product occupies that intersection, because occupying it means handing the keys to the user, which the incumbent business model is built to avoid. That is why it is novel, why it has not been done, and how it differs from everything else — stated with the limits attached, which is exactly why the surviving claims can be trusted.

Technical architecture

> **Scope.** This document specifies the proven layers of Sovereign Memory at the level a reviewer needs to re-derive them: the on-disk and in-RAM data structures, the exact seal / recall / forget / consolidate / admit / release / aggregate / ground protocols, the new per-answer receipt layer (grounding, stapled transcript, PAIR, immunity, the Empty Witness), oblivious recall, which `sovereign/` kernel primitives are reused, and a citation to the prototype file plus a passing test behind every technical claim. Where a claim has a boundary, the boundary is stated with the same precision as the claim.
> **Honesty contract (binding).** Signatures, hashes, and receipts prove *custody, integrity, ordering, consent, minimization, grounding, non-supply, exclusion, and forgetting* — **not** confidentiality-during-compute, **not** FHE, **not** what a model internally attended to, **not** semantic truth. On-device privacy is *locality*, not FHE. We never claim a server or model "cannot read" data on a path where it can. Offline forget / freshness / non-supply proofs need a pinned epoch, nonce, or witness. The DP *release* needs an honest or threshold aggregator (the *noise generation* does not). Novelty is **fusion-grade and capability-grade**, not a new cryptographic primitive. There is no token.

All prototypes are self-contained Node (v18+, built-ins only — `node:crypto`, `node:assert`, `node:fs`; no npm, no network). Each layer was written by one agent and attacked by an independent agent; verifiers were only ever made stricter, never weakened to pass. The recurring lesson of the build-and-falsify program is named once and applies throughout: **every break that ever landed lived in composition glue or a fail-open default — never in a red-teamed cryptographic core.** Paths below are relative to `research/prototypes/`.

The stack has two halves that compose into one system:

- **Where memory lives (L1–L3, L5):** un-seizable custody — TRM's two-secret spine, the recomputed-never-stored QRSM projection, the collective-learning layer.
- **What crossed the boundary (L2, L4, and the receipt family):** an accountable record — the release ledger and the per-answer receipts that prove what was supplied, what was grounded, what was excluded, and what was forgotten.

L1 — TRM: the two-secret spine (`trm/`)

The canonical memory record `M` exists nowhere in full. Each device at rest holds only `{ciphertext} + {1 recall-key share} + {1 consent share}` — none of which, alone or below threshold, yields `M` or the authority to read it. Two independent secrets ride one device roster.

Data structures

At-rest envelope (per device), after the T1 fix:

```
atRestEnvelope = { iv, ct, tag, ciphertextCommitment }
```

`sealRecord(enc(M), k_r=2, n=3, {ownerPrivateKeyPem})` (`threshold-shared-memory/shamir.mjs`) produces an AES-256-GCM envelope under a per-record recall key `K_rec`. `K_rec` is then **Shamir-split over GF(2^8)** (reduction polynomial `0x11b`, generator `0x03`) into `n=3` shares, threshold `k_r=2`. `provisionRecord` (`trm/trm.mjs`) **strips** the unsalted `plaintextCommitment = SHA256(M)` that `sealRecord` leaves in the envelope: replicated to every device, it was a low-entropy plaintext-recovery target and an equality oracle recoverable from one device with no key share. Integrity at rest survives via the GCM tag plus `ciphertextCommitment` (a high-entropy commitment over `iv||tag||ct`). The recall-key-share holders are bound by a P-256 ECDSA signature over `canonicalBinding({commitment, n, k, shareIds})` (`shamir.mjs`) against a caller-pinned owner key.

Consent group (threshold-Schnorr over P-256):

```
consentGroup = { threshold: k_a, total: n, groupPublicKeyHex, shares: [{id, x, P}] }
```

`createConsentGroup(k_a=2, n=3)` (`trm.mjs`) samples a degree-`(k_a-1)` polynomial over the P-256 scalar field; `coeffs[0]` is the group secret `x_grp`, used to define the polynomial and then **dropped** — nothing returned holds it. Each device stores one share `(id, x_i)` and its public share `P_i = x_i·G`. The binding invariant is `k_a ≥ k_r`: consent is the gate, not a side effect of decryption.

The two-secret spine (consent is not decryption)

This is the novel core. The recall-consent signature is a `k_a`-of-`n` threshold Schnorr whose **group scalar `s_grp` is never reconstructed** — partials are summed via Lagrange-in-the-exponent (`thresholdConsentSign`, `trm.mjs`):

- per-signer nonce `r_i`, commitment `R_i = r_i·G`; aggregate `R = Σ R_i`
- challenge `c = H(R || P_grp || msg) mod n`
- partial `z_i = r_i + c · λ_i · x_i`, where `λ_i` is the Lagrange coefficient at 0 for the active signer set
- aggregate `z = Σ z_i`

Verification (`verifyConsentSignature`, `trm.mjs`) checks **both** the aggregate equation `z·G == R + c·P_grp` **and** every partial `z_i·G == R_i + c·λ_i·P_i`, pinning each `P_i` to the enrolled share and rebuilding `R, z` from the partials, so a free-floating `(R, z)` that merely satisfies the aggregate equation is rejected.

The proven property — test **T2** (`trm/trm.attack.test.mjs`), held 17 independent forgeries — is that a coordinator who has reconstructed `K_rec` **and decrypted** `M` still cannot produce a verifying consent receipt without a real `k_a`-of-`n` aggregate. **Being able to decrypt and being authorized to read are cryptographically independent, separately-auditable events.** The marginal *confidentiality* benefit of two splits over a single threshold-gated key is narrow (`k_a ≥ k_r`); the payoff is *auditability* — provable consent separable from decryption.

Closest prior art. Threshold Schnorr and Shamir secret sharing are textbook; FROST (Komlo-Goldberg) is the production multi-round variant. The fusion that is unoccupied is the *application*: two independent secrets (info-theoretic recall-key split + never-reconstructed consent signature) on one device roster bound to a CRDT memory ledger, with the plaintext-assembly moment surfaced as a first-class receipted transition. TRM is single-coordinator, **not** FROST: it does not support concurrent multi-round signing (Drijvers et al. / the ROS problem). That is a stated limit.

Recall protocol — one event, three artifacts

`recall(…)` (`trm.mjs`) emits:

1. **L3 reconstruction.** Gather `k_r=2` key shares → reconstruct `K_rec` on **exactly one device for one turn** → `openRecord` → `M`. This is the honest L3 locality gap, surfaced as a receipted transition, never hidden.
2. **Consent receipt.** `thresholdConsentSign` over the recall-receipt hash → `{msgHash, sig}`, containing no plaintext.
3. **Replayable recall receipt.** Pins `{scope, lamportFrontier, rankerVersion, surfacedHashes, ledgerHead, custodianPublicKeyPem}` so any holder of the op-log recomputes the byte-identical surfaced set.

The ranker (`deterministicRank`, `trm.mjs`, `RANKER_VERSION = 'trm-ranker/1-lamport-then-opid'`) is a pure function of `(ops, scope, frontier)`: filter to scope/frontier, sort by `(lamport, op_id)`. `replayRecall` (`trm.mjs`) re-derives `op_id = SHA256(canonical{payload, lamport})` for every op, rebuilds the RFC-6962 Merkle root over the full replica log, requires it to equal the receipt's **pinned signed root**, and proves per-op inclusion before ranking — so a replica that keeps honest `op_id` labels over substituted payloads, forks the log, or presents a bogus head fails closed (test **T3**). Replay holds for **deterministic** ranking only; semantic vector recall degrades the guarantee.

Forget protocol — threshold shred + freshness anchor

Shredding `n − k_r + 1 = 2` of 3 recall-key shares leaves one survivor below `k_r`, so `K_rec` is **information-theoretically unrecoverable**. This is robust — a physical fact about the shares. The offline *proof* of forgetting is bounded by freshness: a stateless verifier given only a snapshot cannot distinguish "latest" from a valid past state (the R3 stale-but-genuine-seal break, §L2).

`buildForgetProof` (`trm.mjs`) therefore carries **two independent threshold attestations over one freshness anchor**:

```
forgetAnchorMsg = "FM-TRM|forget|v2|" + [recordId, epoch, verifierNonce, headHash,
                  "kr:"+keyThreshold, "kn:"+keyTotal, "survivors:"+ids, "destroyed:"+ids]
```

1. the **consent quorum** co-signs the anchor (authorization), and
2. a **threshold of recall-key-share holders** each ECDSA-sign "I destroyed share `i`" (`signKeyShredAttestation`).

`verifyForgetProof` (`trm.mjs`) rejects a stale epoch (`proof.epoch < pinned.minEpoch`), a wrong nonce, a head mismatch, `keyThreshold/keyTotal` disagreeing with pinned record params, survivors not strictly below `k_r`, and fewer than `n − k_r + 1` distinct authentic destroyer attestations. The consent quorum **alone** can no longer assert a shred — consent shares and recall-key shares are separate custody (the iter-7 T4 break, now fixed; `trm/trm.test.mjs` = 22/22, `trm.t4.attack.mjs` held). **Honest limit:** this proves *attestation* of destruction plus info-theoretic unrecoverability *if* a true threshold actually dropped its shares — not physical destruction on a replica that secretly kept a copy.

L2 — Release ledger: the accountable spine

Every memory op (write / edit / admit / forget / release) is a signed leaf in an append-only log, and absence is provable.

Append-only Merkle ledger (`memory-ledger/ledger.mjs`)

RFC-6962 layout: leaves are `SHA256(0x00 || op_id)`, interior nodes `SHA256(0x01 || left || right)` (domain separation prevents leaf/node second-preimage confusion). `op_id = SHA256(canonical(body))` with the Lamport clock inside the signed body; checkpoints are hash-chained via `prev_checkpoint_hash` and P-256 signed. The verifier provides **inclusion** (audit path for a leaf) and **consistency** (the old tree is a strict prefix of the new) proofs plus an authenticated head. Hardened across A1 (empty-tree consistency must check `oldRoot == SHA256("")`), A2 (`proof.tree_size` re-bound to the signed checkpoint), A3 (reject trailing-junk audit paths), and B2.3 (strict integer leaf-index guard). `ledger.test.mjs` = 21/21; `attack.test.mjs` = 18/18; `break.test.mjs` = 36/36 over `n=1..40`. The head defeats rollback / truncation / equivocation **relative to a pinned head**; first-contact and cross-party non-equivocation need an external witness. Closest prior art: Certificate Transparency / Trillian (RFC 9162); the wedge is user-owned custody, not a public CA log.

Quorum write-admission (`quorum-write-admission/quorum.mjs`)

No memory becomes durable without `k`-of-`n` of your devices. An admission receipt carries `≥ k` distinct enrolled-device P-256 signatures over the exact canonical body `{content_hash, op_id, epoch}` (prefixed `FM-019|admit|v1|`). A `device_id = SHA256(SPKI-DER)` is a binding fingerprint, so two keys can never share an identity and the verifier counts distinct enrolled signers. The hardened verifier **re-derives** `op_id = SHA256(canonical{content_hash, epoch})` and fails closed before counting signers, so a standalone receipt with a rogue `op_id` plus a real quorum cannot verify (the A10 gap). `quorum.test.mjs` = 10/10; `quorum.redteam2.test.mjs` held 12 fresh attacks (consistent-swap, op_id re-encoding, prototype pollution, foreign quorum, malleability dup-signer, cross-epoch retarget). Closest prior art: Provenance Gates (2605.13471) and MemLineage (2605.14421) for signed-op admission; the wedge is the user's own device quorum as the admission authority.

Quorum crypto-shred (`quorum-crypto-shred/quorum_crypto_shred.mjs`)

Forgetting is itself a receipted threshold-signed event. `forgetStatement({contentHash, epoch, op:"shred"})` is the canonical body `K_SIG=2`-of-`3` devices sign; `contentHash = SHA256(iv||tag||ct)` content-binds the receipt to the exact blob; `DROP_COUNT = N − K_R + 1 = 2` shares dropped leaves 1 survivor below `k_r`, so the data key is info-theoretically gone. Hardened to require `contentHash` (omission previously accepted a receipt for any item) and to return `{ok:false}` uniformly on malformed input rather than throw (a fail-open footgun if the caller try/catches). `quorum_crypto_shred.test.mjs` = 10/10. Pair with the L1 freshness anchor for the offline proof. Closest prior art: Vanish (USENIX Security 2009) for Shamir-then-destroy-a-quorum, and "Deleting Secret Data with Public Verifiability" (eprint 2014/364) for publicly-verifiable key destruction; the wedge is quorum-*authorized* deletion whose event is the threshold receipt on a user-owned ledger.

Non-membership SMT (`non-membership/smt.mjs`)

A 256-bit sparse Merkle tree, `key = SHA256(factId)`, addresses every fact to exactly one of `2^256` leaves; only inserted paths are materialized, all else uses precomputed `DEFAULTS`. Distinct domain prefixes separate empty leaf, occupied leaf (`leafHash = SHA256(TAG_LEAF || key || valueHash)`), and node. An **absence** proof is the audit path showing the leaf at `K` is `EMPTY_LEAF`; **presence** is the same path with the occupied-leaf hash; the root is P-256 signed. Making a present key appear absent requires a SHA-256 second-preimage. `test.mjs` = 8/8, `attack2.mjs` = 15/15 (coercion, prototype-chain, 1e5 brute-force). The `expectAbsent` flag is mandatory and strictly boolean (bare-verifier footgun closed). Residual: split-view / equivocation is per-root only.

Proof-of-non-supply — the Empty Witness (`proof-of-non-supply/nonsupply.mjs`)

Per session, the released content hashes are inserted into a signed SMT: `releaseAddr(contentHash)` → `smt.insert(addr, hashValue("supplied:"+ch))`. `proveNonSupply(sessionId, contentHashX, pinnedCheckpoint, headCheckpoint)` returns a witness `{sessionId, contentHashX, absence, signed_root, ledger_op, inclusion, consistency}`. The decisive hardenings in `verifyNonSupply`:

- **V4 query-binding:** recompute `releaseAddr(contentHashX)` and require `absence.key === it` — without this a custodian answers "is X absent?" with a valid absence proof for an unrelated, never-released Y (the original one-line break).
- **V5 session binding:** `witness.sessionId` is *authenticated*, not trusted — it must equal the `sessionId` inside both the P-256-signed release-set root and the ledger op payload.
- append-only soundness via the RFC-6962 consistency proof: the log only grew since the pinned checkpoint, so retroactive insertion of X is a visible later entry.

`nonsupply.test.mjs` = 13/13; `nonsupply.reattack.mjs` = 14/14 (hex-case malleability, query re-bind, cosmetic flags, type confusion, signed-root edit, chain truncation, stale epoch). **Honest scope:** this proves custody-side non-*release*, not what a remote model internally conditioned on; the verifier must pin its latest checkpoint. Surfaced behind a UI, this is *The Empty Witness*: a stateless, offline, court-handable proof that a named fact was never released into a named session. Closest prior art that misses: SCITT refusal-events and ZKPROV (2506.20915) attest *positive* logged events; neither constructs a proof of a negative over user-owned custody.

L3 — QRSM: recomputed-never-stored + possession root (`qrsm/`)

The queryable projection (inverted index, LWW graph, summaries) is never written to disk. It is rebuilt in RAM by replaying the encrypted op-log, and the replay key is reconstructed live from a `k`-of-`n` threshold-PRF over **off-device physical tokens** (FIDO2/WebAuthn-PRF, modeled by an `hmac-secret` closure).

The possession-root mechanism (why it does not collapse to shares-at-rest)

At enrollment (`enroll`, `qrsm.mjs`), a random 32-byte seed is the `f(0)` of a per-byte degree-`(k-1)` GF(256) polynomial. For each token `i`, the required share `y_i = f(x_i)` is computed, the token's **live** PRF output `prf_i = HMAC(tokenSecret_i, salt)` is taken, and only the **public offset** `off_i = y_i XOR prf_i` is persisted. The seed and polynomial are then zeroized. Persisted at-rest metadata:

```
enrollment.public = { k, n, tokenSlots, offsets:{x→hex}, prfSaltByX, seedCommitment, hkdfSalt, hkdfInfo }
```

`off_i` is a **one-time-pad** encryption of the share under the token's PRF output: given `off_i` alone (no `tokenSecret_i`, which never touches the device), every `y_i` is equally likely. This is the entire delta over TRM. The `T-QUORUM-LOADBEARING` test hands an attacker the full device-at-rest state and confirms no pair or triple of offsets interpolates to the seed, 50k seed-guesses miss the commitment, a byte-grep finds no key/seed/share at rest, and 30k random keys never decrypt (GCM backstop). An at-rest scan after teardown finds **exactly two files** (`oplog.cipher.json`, `enrollment.public.json`) — no derived index. `qrsm.test.mjs` = 6/6; `qrsm.replay.test.mjs` passes.

Live ceremony and materialize

`liveCeremony(pub, presentTokens)` (`qrsm.mjs`) recomputes each present token's `prf_i`, unmasks `y_i = prf_i XOR off_i`, Lagrange-combines `k` shares at `x=0` → seed → `HKDF(seed)` → AES-256-GCM op-log key, all in RAM. It throws (no key materializes) if fewer than `k` tokens are present or the recovered seed mismatches `seedCommitment`. `materialize(opLogKey, sealedOps)` (`qrsm.mjs`) decrypts each op (`op_id = SHA256(iv||tag||ct)`, content-addressed and tamper-evident), applies CRDT order `(lamport, author, id)` with LWW per key, builds the projection, and exposes `query/get/drop`. **Honest limit:** at-rest only — while a quorum is present the projection is plaintext in RAM and swap-spillable; pure software cannot scrub RAM/swap. Closest prior art: Opal (2604.02522) for private AI-memory retrieval (TEE+ORAM); QRSM's wedge is non-materialization plus a user-held possession root with no enclave.

L4 — Forget-by-reprojection (`reprojection-forget/`)

Because the symbolic memory is a **pure, deterministic, content-addressed projection** of the op-log, **forgetting = delete the source op + re-project.** The fact is gone by construction — no machine-unlearning uncertainty.

The `OpLog` (`reproj.mjs`) is a gap-free, hash-chained, P-256-signed sequence; each op is AES-GCM encrypted under a per-op key. `forget(opId)` (`reproj.mjs`) appends a signed REMOVE tombstone **and crypto-shreds the per-op key**. The surviving op-set (asserted ops, not tombstoned, in canonical Lamport order) is the sole input to projection; the projector decrypts there. `verifyOpLog` (`reproj.mjs`) checks gap-free Lamport, hash-chain, `opId == H(body)`, and the owner signature. Proven (`reproj.test.mjs`, 26/26 over 3 processes; 16-attack suite held): the forgotten fact is provably absent from index/facts/summary, a kept fact survives surgically, and same-set re-projection is byte-identical (deterministic content address). Three overclaim fixes are recorded: resolve LWW by Lamport inside projection (the content address was array-order sensitive); the divergence guarantee needs `|op-set| ≥ 2`; the extractive summary is lossy on multi-word keys (the fact map stays correct).

Honest boundaries. Clean for the deterministic **symbolic** layer only. A trained **neural adapter** cannot be regenerated bit-for-bit across hardware (parallel-reduction order over non-associative float add), so it is handled by **content-addressed provenance** — an owner-signed receipt binding `adapter_hash → surviving-op-set` proves "derived from exactly this op-set, which excludes X" without regeneration. Implementation hazard: re-projection must be **from scratch** — a stale cached summary still leaks the forgotten fact. Closest prior art: the published ML-unlearning line (ZK-APEX 2512.09953, Verifiable Federated Unlearning 2510.00833) proves a *model* changed; the wedge here is ledger-side absence over user-owned content-addressed memory with an offline static verifier.

L5 — FlowMemory-DNG: collective memory with a real DP bound (`real-dp-collective/dng.mjs`)

Everyone's AI improves from the collective without any individual's memory being seen, under a real (computational) DP bound rather than a declared one. Four pieces compose:

1. **Calibrated distributed noise, actually sampled.** Each device clips to L2 sensitivity, quantizes to the ring `Z_Q` (`Q = 4611686018427387847`, prime below `2^62`), and adds its own share of integer-exact discrete-Gaussian / truncated-binomial noise (`sampleDiscreteGaussian`, `binomialNoiseRaw`). The aggregate noise exists as a **mathematical consequence of summation** (closure under addition), not a ledger line.
2. **Secure-aggregation masking on the noisy share.** `maskShare` / `aggregateMasked` (Bonawitz-style double mask, `pairMask(pairSeed, round)`) means the aggregator only ever sums masked noisy shares mod `Q`; the un-noised sum **never forms anywhere** — structurally killing the "declare epsilon, release the raw secret" lie that a byte-invariant verifier would permit.
3. **Public-coin proof of correct sampling.** Each device emits a hiding+binding commitment to its noise share plus a cut-and-choose / Fiat-Shamir sigma argument (`proveBinomialNoise` / `verifyBinomialNoise`) bound to a public coin so the prover cannot choose convenient noise. A zero-noise or biased free-rider produces an invalid proof and is **rejected** — the verifier is no longer information-blind to noise application.

4. **User-held RDP-filter odometer.** Each user keeps an append-only, hash-chained, P-256-signed receipt chain. Each contribution records the realized RDP cost `rho_i` at fixed order `alpha`, the noise commitment, and the running total (fixed-point micro-rho, `RHO_SCALE = 1e9`, no float drift). Composition is the RDP **filter**: maintain running `rho` and HALT before exceeding `rho_cap` — valid under adaptive spend with no adaptivity penalty; anti-rollback is the hash chain (rewind = fork = detectable). Accounting: `rho = Δ²/(2σ²)`, `eps_RDP(α) = rho·α`, `eps = rho + 2√(rho·ln(1/δ))`.

Why epsilon is a real bound. Measured (`dng.dp.test.mjs`, `dng.zk.test.mjs`, `dng.*.attack.test.mjs`; 20/20): aggregate variance 407.6 vs target 400; the 20-round aggregate equals clipped-true-sum + sampled-noise exactly (the clean sum never materializes); the honest output is empirically (ε,δ)-DP with a monotone σ knob (ε sweeps 1.23→0.15; membership-adversary advantage 0.038, well under the 0.199 bound); distributed per-device noise ≈ central-curator noise (no trusted curator for the noise); the zero-noise free-rider is rejected.

Honest 4-part trust boundary: (A) the *release* still needs an honest/threshold **aggregator** — the default proof opens the noise count `sumOpening.s` in cleartext, so a malicious aggregator could strip the noise (the A7 overclaim, retracted); **(B)** masking confidentiality needs a non-colluding/threshold layer (full `M-1` collusion → local-DP, never the clean value); **(C)** at least one honest noise contributor (tightness ∝ honest fraction, over-provision by `1/(1−ξ)`); **(D)** the bound is **computational DP**, not information-theoretic (Biswas-Cormode / Haitner-Omri — verifiable DP cannot be trust-free; it rests on commitment hiding+binding). The ZK proof is school-strength cut-and-choose, not a production SNARK.

The receipt layer — per-answer data governance

L1–L5 establish *where memory lives* and the *ledger of what crossed the boundary*. The receipt layer makes a single AI turn third-party-auditable offline, value-blind. Each receipt is a strict composition of already-proven legs imported byte-for-byte; the new code is the outer gate, and each new gate was independently red-teamed. None of these receipts proves what a remote model internally attended to — that boundary requires an on-device / MCP / TEE attestation and is out of scope for a pure signature scheme. This is the honest answer to the c0mpute / DeepProve / Verde compute-receipt sector, which can attest *the computation* but structurally cannot produce the *input-provenance* half: every such proof requires the prover to hold the plaintext input, so the receipt hides data from the verifier, never from the prover.

Grounding receipt — the dual of the Empty Witness (`grounding-receipt/grounding_receipt.mjs`)

Where the Empty Witness proves a fact was *never supplied* (absence), the grounding receipt proves the inverse positive-closure property: an answer was grounded in **exactly** a chosen subset of the owner's own memory, and **nothing off-memory** was smuggled into the prompt — value-blind.

Construction. Each memory entry is a value-hiding salted commitment

```
commitment = SHA256( "FM039-COMMIT-v1" || 0x00 || salt || 0x00 || field_name || 0x00 || value )
```

with `salt ≥ 16 bytes` (a hardening floor; the default `freshSalt()` is 32 random bytes — see the value-hiding caveat below). The commitment is stored as the SMT leaf's `valueHash` at `key = H(field_name)`, so the reused, unmodified non-membership SMT membership verifier binds the leaf to `(H(field_name), commitment)` without ever seeing the value. The signed receipt body commits to `{answer_hash, input_hash, smt_root, epoch, used_entries[]}`, each used entry carrying `{field_name, commitment, membership_proof}`.

The closure anchor. `input_hash = SHA256(canonical(sorted(unique(commitment_hex[]))))`. The prompt-builder publicly commits to grounding the prompt in *exactly* this commitment-set. An off-memory blob must either (a) appear in the hashed set — but then it has no membership proof and **membership** fails — or (b) be omitted — but then `input_hash` no longer reflects the claimed content and **closure** fails. `verifyReceipt` recomputes `input_hash` over **exactly the membership-proven commitment set**, so the hashed set always equals the proven set; off-memory content cannot satisfy membership and closure together. The verifier requires a pinned epoch (fail-closed without it), re-binds the receipt's `smt_root` to the owner-signed root it independently holds, checks the envelope signature, then membership and closure.

Status. `grounding_receipt.test.mjs` = 17/17; `grounding_receipt.attack.test.mjs` = 20 fresh attacks all resisted (closure-desync via duplicate-commitment collapse, mismatched field/commitment pair, `String()`-coercion smuggle, silent-append under-report, off-memory smuggle under an attacker root, absence-flipped-to-membership, SMT proof malleability, epoch type-confusion / epoch-0 / same-epoch cross-root replay / forward-dating, structured-value/salt leak, non-grounded-entry leak). **Honest limits:** proves the *custody layer* supplied exactly these entries, not what the model internally did with them; value-hiding requires a `≥ 128-bit` secret per-field salt (a low-entropy committed value is brute-forceable if the salt is short — the floor is byte-length, not entropy, so the random default salt is the real guard); the verifier must pin the latest epoch. Closest prior art that misses: zkRAG / VeriRAG / V3DB prove retrieval over a *service*-owned DB; selective-disclosure credentials (BBS+/SD-JWT) are for attributes; none bind generation to a *user*-owned memory subset with a closure property.

Stapled transcript — the input-provenance half of a proof-of-inference receipt (`stapled-transcript/stapled.mjs`)

The grounding receipt's `input_hash` is over salted *commitment hexes*, not over the prompt bytes a GPU consumes. A real compute node hashes the *decrypted context* it ran. The stapled transcript joins the two halves honestly and **cross-binds** them.

The kill condition (what is deliberately refused). Coercing both sides to hash the same artificial commitment-set would make the staple a redundant second signature over the grounding receipt's own `input_hash` and prove nothing about what the GPU consumed. The staple is meaningful only if the compute side commits to the same ordered per-segment structure the grounding half closes over, derived **from the actual segment bytes + salt the node decrypts**.

The ordered-Merkle cross-bind (the load-bearing new gate). The released consented slice is an ordered list of `(segment_i, salt_i, field_i)` triples, one per used entry. Each triple reproduces that entry's grounding commitment `C_i = commitEntry(field_i, segment_i, salt_i)` — byte-identical to the commitment in the grounding receipt's closure set. The compute node forms its prompt by concatenating the segments in order and computes `context_hash = orderedMerkleRoot([C_1..C_n])`, a **position-binding** root: `leaf_i = H(LEAF_DOMAIN || 0x00 || u32be(i) || 0x00 || C_i)`, `node = H(NODE_DOMAIN || 0x00 || left || 0x00 || right)`, with index folded into each leaf (so a transposition of equal-shaped subtrees is detectable) and the empty root fail-closed. The verifier recomputes `orderedMerkleRoot` over the grounding receipt's own `used_entries[]` commitments — which it already holds, value-blind — and fails closed unless `compute.context_hash == owner-signed context_root == orderedMerkleRoot(closure commitments)`. An off-memory segment has no commitment in the closure set (root differs → rejected); a dropped entry changes the leaf count (rejected); a reorder permutes positions (rejected). The verifier never sees a value or salt.

Reuse discipline. `grounding_receipt.mjs` is imported byte-for-byte and never modified; the context digest and root are bound in a separate signed augmented record, layered on as a strict addition. An owner-attested salted-bytes `context_digest` is kept *alongside* the Merkle root as a literal-layout binding, but it is the owner's own attestation, not a value-blind-verifier-recomputable guarantee.

Status. Across three independent suites, 51 adversarial attempts resisted, 0 breaks: `stapled.attack.test.mjs` 13/13 (incl. the redundant-signature kill-condition, encoding-malleability, salt-truncation footgun), `stapled.redteam.attack.test.mjs` 22/22, `stapled.crossbind.adversary.test.mjs` 16/16 (incl. the Bitcoin CVE-2012-2459 duplicate-leaf and attacker-chosen-leaves attempts). The unmodified grounding receipt stays 17/17 + 20/20. **Honest limit (residual):** the ordered root binds the committed *segments* and their *order*, **not** the interstitial/glue bytes a host places between segments when formatting the final prompt; binding the exact literal prompt bytes to a value-blind verifier needs a ZK opening proof (named research upgrade, out of scope). The staple is exactly as strong as the compute node's own proof-of-inference binding `context_hash` to the real computation — it inherits, never strengthens, that trust model. This is the right operand the compute-receipt sector cannot produce on its own.

Provable AI interaction record — PAIR (`pair/pair.mjs`)

PAIR makes the *complete* data-governance of one turn auditable offline, value-blind, by orchestrating three byte-for-byte-unmodified legs and adding one strictly-stricter outer gate:

- **what was used** — the grounding receipt (membership + closure);
- **where it ran and that nothing off-memory entered** — the stapled compute cross-bind;

- **what was provably never supplied** — one Empty Witness per never-supplied fact;
- plus the new outer gates: the *used* address-set and the *never-supplied* address-set are **disjoint**; all legs share **one** epoch/freshness pin; every non-supply witness is **session-bound** to a verifier-pinned release session and release root (closing the decoy-session splice, where a custodian that did release X tries to satisfy non-supply against a decoy session that never released it); and the owner signs the whole record.

The shared fact identity across legs is the field-name's 256-bit SMT address `hashKey(field_name)` — a namespace label, never a value or salt — which is exactly what the underlying legs already expose. The verifier holds two anchors out-of-band (an epoch pin and a ledger-checkpoint pin) and fails closed without either. The court-facing record strips the prover→compute consented slice (which carries plaintext segments + salts); the verifier only re-hashes commitments it already holds.

Status. `pair.test.mjs` = 28/28; `pair.adversary2.test.mjs` = 13/13. **Honest limit:** inherits every leg's honest limit and the compute receipt's left-half trust model; per-interaction non-supply is bound to the verifier-pinned release session, but the verifier must obtain that pinned session id and release root from a trusted anchor, not from the record under audit.

Memory immunity receipt — provable exclusion of poison (`memory-immunity/immunity.mjs`)

The grounding receipt is silent on *who authored* the entries it closes over. The poisoning threat model (OWASP ASI06 / MINJA / MemoryGraft) is precisely about an op that looks like memory but was authored by no enrolled principal. The immunity receipt proves the genuinely-new negative property: a named answer's symbolic grounding set was reprojected from **exactly** the enrolled-author-signed ops, and **zero** unsigned / forged / non-enrolled ops contributed.

Construction. An owner-signed roster enrolls external author public keys (not the agent self-signing). Each memory op binds `(key, commitment)` and is signed by its author over the canonical body; `op_id` is content-derived. `partition(candidateLog, enrolledSet)` deterministically splits the candidate log into **admitted** (structural shape ok, `op_id` re-derives, `body.author` in the enrolled set, signature verifies under that author) and **quarantine** (anything else), fail-closed. The symbolic projection (`reproject`, LWW-by-`(lamport, op_id)`) is a pure function of the admitted set only; the receipt commits to both the admitted-set projection root and a quarantine root. The verifier **re-partitions from the raw candidate log itself** (it does not trust the receipt's claimed partition), re-projects, and checks the answer's grounding set is closed over the reprojection that excludes the quarantine. A forged op forced into the prompt cannot be in admitted (no enrolled signature), so closure breaks.

The verifier requires three out-of-band pins, all mandatory and fail-closed: the **roster content hash** (not just the epoch integer — the owner could sign a second roster at the same epoch enrolling a poisoner; that fork carries a valid owner signature and the right epoch, so epoch-only pinning is insufficient), the **candidate-log head**, and the **epoch**.

Status. `immunity.test.mjs` = 8/8; `immunity.attack.test.mjs`, `immunity.soundness.attack.test.mjs`, `immunity.roster.redteam.test.mjs` = 19 + 16 + 13 attacks, **0 soundness breaks** (forged/mutated ops re-derived as quarantined, same-epoch roster fork rejected, re-signed clean-root receipt without the owner key rejected). One companion harness, `immunity.honesty.attack.test.mjs`, reports `BREAKS = 0, OVERCLAIMS = 0` but exits non-zero on five probes: these are *honesty characterizations*, not soundness defects — they assert the verifier makes **no completeness claim** (it is scoped to the offered log; an omitted enrolled op is out of scope, by design) and they call the verifier in ways that correctly trip the mandatory roster-hash pin. The boundary is real and is stated here as a limit, not a pass. **Honest limits:** proves *attributability*, not truthfulness (an enrolled author can sign a false fact — the point is that poison becomes attributable and revocable instead of anonymous); does not cover live-query prompt injection or side-channel poison injected past the receipted custody boundary; the verifier makes no claim about ops it was never shown (completeness is an out-of-band pin). Closest prior art: Provenance Gates (2605.13471) / MemLineage (2605.14421) label-and-gate signed ops; the wedge is the *negative exclusion* proof — value-hiding, answer-bound, re-partitioned by the verifier.

Provable consolidation receipt (`consolidation/consolidation.mjs`)

A durable summary `D` is the distilled residue of exactly a named set of ephemeral session entries `E`, and every one of those ephemerals is now information-theoretically unrecoverable because its per-ephemeral key was crypto-shredded. One owner-signed receipt makes both legs checkable offline over the same source set, fail-closed: **lineage-IN == destroyed-OUT** — "this durable fact distills these now-unrecoverable sessions and nothing else, and nothing off that set survives decryptable."

This is strictly stricter than a tombstone receipt. A tombstone is forgeable: a custodian can append REMOVE ops while leaving the AES keys live and the ciphertext decryptable. The consolidation verifier performs a **real key-destruction check** (per eprint 2014/364, deletion must be publicly verifiable via key destruction, not asserted): for every cited ephemeral it confirms the per-ephemeral key is gone from the exported custody **and** the ciphertext does not decrypt under any key the snapshot carries — destruction observed, not trusted. The prerequisite (enforced, fail-closed): every ephemeral is encrypted under its own random per-ephemeral DEK (`crypto.randomBytes(32)`), not an HKDF derivation over a retained seed — closing break A, where a deterministic HKDF key could be recomputed and a "destroyed" key re-derived. An owner P-256 `custody_seal` authenticates the post-forget custody so an omitted-but-live key cannot fake a shred (the omitted-key half of break A). The ASSERT op's signed body commits to `(field_name, lineage_commitment)` and the verifier requires the cited `op_id` 's authenticated body to commit to the same pair, so a summary's lineage cannot point at one session while the destroyed `op_id` points at another (break B). The lineage leg is the unmodified grounding receipt; the op-log verifier is the unmodified reprojection-forget `verifyOpLog` , with the surviving op-set re-derived from the signed snapshot.

Status (now green). `consolidation.test.mjs` = 15/15; `adversary.attack.test.mjs` = 11 attacks, 0 broke. This supersedes the earlier broken forget-certificate prototype (`forget-certificate/`), whose verifier accepted a "gone-from-recompute" leg for a still-live fact (a real soundness break, recorded in the evidence log); the consolidation verifier recomputes the projection from surviving ops and observes key destruction rather than trusting a flag. **Honest limits:** binds the summary's hash to its source commitments, not the summary's semantic faithfulness (summarization is neural and off the deterministic layer); proves unrecoverability *from this exported state now*, not that no plaintext copy was exfiltrated earlier; per-ephemeral keys are HKDF-derived from a master seed the owner could in principle re-derive from if both the seed and the purpose label survived owner-side — unrecoverability is relative to the exported state (the FM-031 caveat); the verifier must pin the epoch.

Oblivious recall — query-independent access pattern (`oblivious-recall/`)

A byzantine swarm serves fixed-size content-addressed erasure-coded ciphertext blocks with a **query-independent** access pattern and **never decrypts**, plus a signed **obliviousness witness** proving the on-wire access set is `f(epoch, public schedule)` , not the query. The contrast with the compute sector is exact: a split-inference node must decrypt; a FlowMemory storage node carries a proof it learned nothing about which block was wanted.

The full-batch access set is the entire catalog per epoch; the witness binds the on-wire `(nodeId, blockId)` order (length `\|accessSet\| · nodeCount`), so a query-dependent reorder is rejected. The verifier requires a mandatory `pinnedNodeCount` and a P-256-signed swarm-membership commitment `{d, memberCount, k, Merkle-root}` with `witness.nodeCount == memberCount` and `nodeCount ≥ k` , so a self-reported node-count cannot be downgraded (e.g. to `1` , which would bind only node-0). **Status:** `oblivious_recall.test.mjs` = 21/21; `oblivious_recall.attack.test.mjs` = 8/8; redteam suites including the node-count-downgrade probe green; a real trained distinguisher over the full node-tagged served log (4000 trials) measured advantage 0.0003–0.009, under the ~0.04 noise floor; the trace is byte-identical across queries. **Honest cost (stated, not free):** full-batch blowup equals catalog size (8× for 8 items); block existence, count, and sizes are public by design (that is what makes the witness checkable). **Honest negatives recorded for credibility:** the *windowed* variant (cover size below catalog) is **not** oblivious across sessions — a persistent adversary intersects the public per-epoch windows of repeated recalls and de-anonymizes (16→1 in two epochs); do **not** ship windowed as oblivious. The sublinear DPF path (`uspir-dpf/`) is real but **parked** (BGI two-party DPF, O(log N) up / O(1) down, no FHE/TEE) because it has no live-execution obliviousness witness, is single-writer only, and a witnessed-sublinear attempt (`witness.mjs`) **failed the wall**: a self-issued offline witness is fabricable — plain recomputation cannot bind a self-issued witness to runtime behavior; that requires a logged interactive protocol or a SNARK/TEE. This is an honest negative, kept in the record. Closest prior art: Opal (2604.02522) for private retrieval; DPF-PIR over erasure-coded storage and BGI (EUROCRYPT 2015) for the parked path.

Kernel mapping — which `sovereign/` primitives are reused

The layers are not a rewrite; they bind to the existing kernel.

Layer component	Kernel primitive (`sovereign/src/ ...`)	Status
GF(256) Shamir recall-key split / seal / open	`core/shamir.ts` (prototype: `threshold-shared-memory/shamir.mjs`)	proven, red-teamed
Threshold-Schnorr consent (no `s_grp` reconstruction)	`agency/quorum_gate.ts` `signActionWithQuorum` (sums partials)	exists, verified
Content-addressed encrypted blobs / purpose keys	`core/content_store.ts`, `core/hkdf.ts`	exists
Signed CRDT op-log (Lamport, LWW, verify-on-merge)	`mesh/memory_oplog.ts`	exists
Hash-chained signed checkpoints (head pattern)	`core/heartbeat.ts`	exists
P-256 sign/verify, canonicalization, identity	`core/p256.ts`, `core/canonical.ts`, `core/identity.ts`	exists
Replayable recall receipt @ Lamport frontier	net-new vs `infer/cognition_inference_receipt.ts` (frontier-replay binding)	prototype
Retrievability audits (frozen honesty-boundary object)	clone `mesh/reachability_receipt.ts` (`normalizeBoundary` throws if a flag weakens)	pattern reused
Off-device token PRF / possession root (L3)	`core/seed_custody.ts` + WebAuthn-PRF (modeled in `qrsm.mjs`)	net-new binding
Distributed noise + secure-agg + DP accounting (L5)	`learn/secure_agg.ts`, `learn/dp_accounting.ts`, `learn/dp_spend_ledger.ts`	exists; DNG replaces accounting-only ε with a sampled-and-proven bound
Enrolled-author roster (immunity admission predicate)	`agency/roster_receipt.ts`, `agency/quorum_gate.ts`	exists; immunity adds the negative-exclusion re-partition
Grounding / staple / PAIR receipt envelopes	net-new over `core/p256.ts` + `non-membership/smt.mjs` membership	prototype
Inheritance / soul-file (Soul Will roadmap)	`core/inheritance.ts`, `core/will.ts`, `core/soul_file.ts`	exists

Build path into `sovereign/`. The static receipt verifier is the one blocker: `agency/action_transcript_receipt.ts` and `agency/roster_receipt.ts` gate receipts with `assertExactKeys` / `hasExactKeys` over hardcoded key lists, so a new receipt `kind` (admission, non-supply, recall, forget, grounding, staple, immunity, consolidation) is rejected wholesale. **Fork the verifier** — add the new kinds fail-closed rather than relax the exact-keys check. Then: (1) carry the encrypted `ShardManifest` as a blob value under a new `kind` (the `memory_oplog.ts` body validator is closed, so it cannot be a native op field); (2) wire on-device embedding + deterministic recall against `mesh/memory_oplog.ts`; (3) expose the memory over MCP (`agency/mcp_gate.ts`) as the neutral read/write interface; (4) surface the Empty Witness and the grounding receipt behind a UI as the wedge products. The net-new code is the binding layer — a few hundred LOC — not new crypto.

Emergent capabilities (composition of the proven layers)

- **The Empty Witness** — proof-of-non-supply (L2) behind a UI: a stateless verifier answers YES/NO to "was fact X ever released into session S?" and prints VERIFIED NON-SUPPLY for an X that was never supplied.
- **Discovery Shield** (`discovery-shield/combined.test.mjs`, 11/11) — one verifier proves, for the same `(X, S)`, both "no byte released" (L2 non-supply) and "no consent to release granted" (L1 threshold-Schnorr no-consent). It **survives a decryptor**: a party holding `K_rec` + plaintext can forge neither leg. The no-consent leg is open-world (proves no authorizing receipt among those shown under the pinned group; fails safe).

- **The provable context pair** — the Empty Witness (exactly what was *never* supplied) and the grounding receipt (exactly what *was* supplied and nothing else) are both proven; PAIR binds them into one court-handable per-turn record.
- **The provable input-provenance half** — the stapled transcript supplies the operand the compute-receipt sector cannot produce: it cross-binds the consumed context to the consented, value-hiding, user-owned memory subset.
- **Cross-session grounding closure** (`cross-session-closure/`, 14/14 + 11/11 + 20-attack) — a directional HIGH↛LOW non-flow receipt: no memory entry grounded in a sensitive session reappears in a later session's grounding set. Honest limit: disjointness is over *commitments*, not values — re-committing the same value under a fresh salt is a different commitment, so the proof is "custody did not re-supply the committed entry," not "the same value never reappeared."

All inherit the irreducible boundaries: custody-side only (not model cognition), value-hiding is bounded by salt entropy, and the verifier must pin its latest checkpoint/epoch for freshness.

Threat model & honest limits

> Companion to `research/SOVEREIGN_MEMORY_WHITEPAPER.md` and the evidence spine `research/whitepaper/04_proof_and_test_evidence.md`. A security claim is meaningful only against a stated adversary, and honest only if the boundary where it stops is written with the same precision as the claim itself. This document does both. For each adversary it names what is *provably* defended — with the prototype file and the falsifying test that proves it — and what is *not*. It then consolidates the full, binding honest-limits list, including the four limits added by the newest layers (staple glue-bytes, PAIR session pins, immunity roster-hash, consolidation out-of-band copies). It closes with the composition-glue meta-lesson and the four breaks *this build cycle* recorded as fresh evidence that the discipline works.
> **Honesty contract (binding on every line below).** Signatures and hashes prove **custody, integrity, ordering, consent, minimization, grounding, non-supply, exclusion, and forgetting** — never confidentiality-during-compute, never FHE, never what a model internally attended to, never semantic truth. On-device privacy is **locality, not homomorphic computation**. We never claim a server or model "can't read" data on a path where it can. Offline forget/non-supply proofs require a freshness anchor. The DP *release* still needs an honest or threshold aggregator (the noise *generation* does not). Novelty is **fusion-grade**, not a new primitive; the closest prior art is cited wherever a novelty is claimed (§6).

---

1. Scope of the model

The stack defends a **user-owned memory layer** that lives across the user's own devices, with no operator and no account holding the whole. The defended assets are: the **plaintext memory** `M`; the **read-consent** authorization; the **queryable projection** (embeddings / graph / summaries); the **release record** (what crossed the boundary into a model session); the **grounding record** (exactly what was supplied for a given answer, and nothing off-memory); the **interaction record** (the complete data-governance of one turn); the **admission record** (which ops are authored by enrolled principals); the **forget / consolidation** event; and the **collective-learning** contribution.

Every security claim is relative to one of the adversaries in §2. There is no claim of unconditional security, and there is one threat the stack does **not** address and never pretends to: a **compromised live endpoint** — the device that holds the plaintext for the turn it is in use. That gap, the *locality gap*, is the first honest limit (§5.1) and bounds every confidentiality claim in the system. Two further boundaries travel with the entire stack: the proofs are **custody-side** (they constrain what the custody layer supplied, withheld, admitted, and destroyed — not what a remote model internally did with it), and the offline proofs are only as fresh as the anchor the verifier pins (§5.4).

---

2. Adversary models, per layer: defended vs not

2.1 At-rest device seizure (a powered-off device is taken)

The adversary the stack defends *best*, because the architecture is built around it.

- **L1 TRM — defended.** A device at rest holds only the AES-GCM envelope ciphertext, one useless GF(256) Shamir key-share of `K_rec`, and one consent-share. Below the threshold `k_r`, the share carries information-theoretically zero bits of `M`. Test **T1** (`prototypes/trm/trm.attack.test.mjs`, vectors in `trm.attack.t1b.mjs`) hands a verifier one device's *full* at-rest state plus the owner pubkey and confirms it recovers 0 bits of `M` and 0 bits of the group scalar `s_grp`. A hardening break is recorded honestly: the original envelope left an unsalted cleartext `SHA256(M)` commitment on every device (a low-entropy recovery + equality oracle); the fix strips it to a ciphertext-only / HMAC commitment, re-attacked over seven vectors and teeth-checked (re-adding the old commitment recovers `M`). `trm/trm.test.mjs` is 22/22 green.
- **L3 QRSM — defended, and stronger than TRM at rest.** The queryable projection is *never written to disk*. After a session + teardown, an at-rest scan finds exactly two files — the ciphertext op-log and the public enrollment — and no derived index, embeddings, summary, or plaintext (`prototypes/qrsm/qrsm.test.mjs`, 6/6). The replay key is reconstructed live from a `k`-

of-`n` threshold-PRF over **off-device** physical tokens; the device persists only public offsets `off_i = share_i XOR HMAC(tokenSecret_i, salt)`, with `tokenSecret_i` never on the device. The red-team built the "shares-at-rest collapse" strawman (real Shamir shares on disk → seize `k` powered-off devices → recombine) and confirmed QRSM does **not** reduce to it: no pair or triple of offsets interpolates the seed, 50k seed-guesses miss the commitment, a byte-grep finds no key/seed/share at rest, and 30k random keys never decrypt (GCM backstop).

- **Hardware-rooted possession (FM-041 / FM-044) — defended in software-sim, hardware-conditional.** `prototypes/dead-hand-recall/` proves the possession-root in a modeled off-device PRF: power-off seizure of the *entire* at-rest state without a live "tap" → recall fails; a captured PRF transcript from a prior session does not reopen a *new* recall (fresh salt → fresh tap); a stale-epoch replay (counter below the pinned value) is rejected (14/14 + an 11-attack re-attack held). The $58 hardware node (`research/whitepaper/HARDWARE_BUILD.md`) elevates this and the freshness anchor from software-decorative to hardware-enforced — the FIDO2 hmac-secret seed is non-extractable, and an ATECC608B monotonic counter binds freshness into receipts.
- **Not defended even here:** the `|M|` **length leak** (ciphertext length reveals plaintext size) and **metadata** (manifest existence/size, scopes, recall timing, item count). These are not closed by the architecture.

2.2 The decryptor (a party who already holds `K_rec` and the plaintext `M`)

The sharpest adversary in the model, and the source of the system's defining property — decrypt-capability and prove-consent are cryptographically independent. (This is a sound application of standard threshold-Schnorr, not a novel cryptographic primitive; see `research/NOVELTY_AUDIT.md` .)

- **L1 TRM — defended (test T2).** Read-consent is a P-256 threshold-Schnorr signature whose group key is **never reconstructed**; consent is a `k_a` -of- `n` aggregate. A coordinator who has reconstructed `K_rec` *and decrypted* `M` still cannot produce a receipt that verifies under `verifyConsentSignature` without a real `k_a` -of- `n` aggregate. Test **T2** held all 17 forgeries (`prototypes/trm/trm.attack.test.mjs`): free-floating `(R, z)` , back-solved partials, a 4000-iteration Fiat-Shamir grind, single-secret completion, cross-message transplant, partial mix-match. The verifier pins each `P_i` to the enrolled share and checks per-partial `z_i·G == R_i + c·λ_i·P_i` , rejecting lone aggregates. **Being able to decrypt and being able to prove consent are cryptographically independent events.**
- **Discovery Shield — defended (survives the decryptor on both channels).** A single offline verifier proves for a `(fact, session)` pair both "no byte released" and "no consent to release granted." A party holding `K_rec` + plaintext forges *neither* leg: a decryptor-keyed absence root is rejected under the pinned `smtPub` , an empty-leaf swap on a released `X` fails SMT recomputation by second-preimage resistance, and every consent forgery is rejected (`prototypes/discovery-shield/combined.test.mjs` 11/11; `decryptor-attack.test.mjs` 9/9; `attack.binding.test.mjs` 9/9).
- **Not defended:** confidentiality of `M` against this adversary — *by definition*. A decryptor already has the plaintext. T2's claim is narrow and exact: it defeats forged **consent/authorization**, not disclosure. A holder who **exfiltrates plaintext before shredding**, or a **malicious reconstructing device** at recall time, defeats confidentiality by design (§2.6).

2.3 Untrusted host / tail-truncation (the offline-proof adversary)

This adversary serves a stateless offline verifier a *valid but stale* view of the ledger.

- **L2 Release Ledger — defended relative to a pinned head.** The append-only Merkle log carries RFC-6962 inclusion + consistency proofs and an authenticated head. Against a **pinned** head, rollback, truncation, and equivocation are rejected (`prototypes/memory-ledger/ledger.test.mjs` 21/21, `attack.test.mjs` , and the 36-attack `break.test.mjs` covering the B2.3 non-canonical-index gap).
- **Proof-of-Non-Supply / The Empty Witness — defended after a one-line composition fix.** The original `verifyNonSupply` accepted the queried `contentHashX` but never bound it to `absence.key` , letting a custody layer answer "is X absent?" with an absence proof for a *different* never-released item. The fix re-derives the address and requires `absence.key ≡ releaseAddr(contentHashX)` ; the 14-attack re-attack is clean (`prototypes/proof-of-non-supply/nonsupply.test.mjs` , `nonsupply.reattack.mjs`).
- **Grounding Receipt (FM-039) — defended.** A per-answer receipt binds an answer to *exactly* a committed subset of the user's own memory: subset-membership of each used entry against the signed SMT root, **plus closure** (`input_hash ==`

`hash(canonical union of opened entries)`) so nothing off-memory was smuggled into the prompt. The load-bearing invariant is that the verifier recomputes `input_hash` over exactly the membership-proven set, so hashed-set ≡ proven-set. Subset-substitution, off-memory smuggle, SMT malleability, and value-revelation are all rejected, values never exposed (`prototypes/grounding-receipt/grounding_receipt.test.mjs` 17/17 + a fresh 20-attack red-team).

- **Stapled Transcript (FM-045) — defended (the input-provenance half).** The Grounding Receipt is value-hiding: its `input_hash` is over salted *commitments*, not the prompt bytes a GPU consumes. The staple cross-binds a compute-node proof-of-inference receipt (signed under a *separate* compute key) to the unmodified grounding receipt via an **ordered Merkle root over the used-entry commitments**: the verifier recomputes that root over the grounding receipt's own membership-verified set, so an off-memory entry, a dropped entry, or a reorder all change the root and are rejected — verifier-checkable and value-blind. 13 builder legs + 38 independent attacks resisted (`prototypes/stapled-transcript/`), including encoding-malleability and the Bitcoin CVE-2012-2459 duplicate-leaf attack; the kill-condition (coercing both sides to hash the same artificial commitment-set) is explicitly refused.
- **Provable AI Interaction Record (PAIR, FM-048) — defended (full-turn governance).** PAIR staples three unmodified proven legs into one owner-signed, court-handable record for a single turn: **what was used** (grounding membership + closure), **where it ran and that nothing off-memory entered** (stapled compute cross-bind), and **what was provably never supplied** (Empty Witness non-membership) — with a cross-leg disjointness gate (no fact is both used-this-turn and never-supplied), a single shared epoch pin, and an owner signature over the whole record. `verifyPAIR` fails closed unless every leg verifies *and* the new outer gates hold (`prototypes/pair/pair.test.mjs` 28/28; `pair.adversary2.test.mjs` 13/13; `pair.adversary3.probe.mjs` green).
- **Graded Consent Spend-Trace (FM-040) — defended.** A semiring of grades (sensitivity tier × purpose-set × ε-increment × use-count) compiled into a P-256-signed, hash-chained, monotone budget ledger atop quorum-write-admission and RFC-6962, so an over-tier / foreign-purpose / over-cap / sub-quorum release cannot produce a verifiable receipt; a portable offline auditor re-folds the chain and confirms order-independence, monotonicity, and tamper-evidence (`prototypes/expedition-graded-consent-spend-trace-the-a/`: grade-laws 15/15, spend-trace 16/16, attack battery 16/16, plus an 18-attack grade-binding re-attack). Two fix rounds were required to reach proven (§7).
- **Not defended by a *stateless* verifier — a fundamental freshness limit.** The single-key forget-proof (FM-002) was broken twice: first by tail-truncation (no authenticated head), then — after adding a signed head/length — by **stale-but-genuine seal replay** (R3): pairing an old, validly-signed seal with a truncation of the current ledger whose prefix matches forges a positive "forgotten" proof for live data. The root cause is information-theoretic: a fully stateless, non-interactive verifier given only a snapshot **cannot** distinguish "latest" from "a valid past state." Soundness requires a **freshness anchor** — a pinned monotonic epoch (verifier state), a fresh quorum-co-signed nonce (interactivity), or an external witness. This is why every offline-proof claim travels with §5.4, and why TRM's T4 forget proof binds two threshold attestations over one freshness anchor.

2.4 Memory-poisoning adversary (an op that looks like memory but was not authored by you)

The OWASP ASI06 / MINJA / MemoryGraft threat: a forged or non-enrolled op is forced into the memory log so it conditions a later answer.

- **Memory Immunity Receipt — defended (proof-of-exclusion).** The dual of the Grounding Receipt: it proves a named answer's symbolic grounding set was reprojected from *exactly* the enrolled-author-signed ops, and that every unsigned / forged / non-enrolled op is in a quarantine set that contributed nothing. The verifier **re-partitions** the raw candidate log itself (admitted vs rejected), re-projects from admitted-only, and checks the answer's grounding set is closed over that reprojection — so a forged op forced into the prompt cannot be in ADMITTED (no enrolled signature) and closure breaks. Value-hiding throughout (`prototypes/memory-immunity/immunity.test.mjs` 8/8; `immunity.attack.test.mjs` 19/19, soundness breaks = 0; `immunity.soundness.attack.test.mjs` 16/16; `immunity.roster.redteam.test.mjs` 13/13, soundness breaks = 0).
- **Self-Defending Mesh (FM-036) — partial / roadmap.** Threshold Byzantine quarantine of poisoning across the mesh detects and quarantines via consistency + quorum; it is **not** a poison *oracle* (it cannot tell a true fact from a false one) and is

bounded by the honest-quorum assumption. An earlier "ghost corroboration" desync (a forget that purged only the primary endorser) was a glue bug, since fixed by making the belief store a deterministic projection of surviving ops.

- **Not defended: semantic truth.** An enrolled author can sign a *false* fact; immunity proves **attributability**, not truthfulness — the point is that poison becomes attributable and revocable instead of anonymous. **Side-channel poison** outside the op-log (a blob injected past the receipted custody boundary) and **live-query prompt injection** are out of scope; this covers poison arriving as a stored memory *op*.

2.5 Malicious aggregator / colluding peers (collective learning)

- **L5 FlowMemory-DNG — partially defended; one honest retraction.** Each device adds a calibrated *share* of integer-exact discrete-Gaussian noise under secure aggregation; closure-under-summation makes the aggregate provably noised, and a public-coin cut-and-choose proof binds each device to having actually sampled its noise. A zero-noise / biased free-rider is rejected by the sampling proof, and the clean sum **never materializes** — structurally closing the earlier "declare noise, release raw" lie (`prototypes/real-dp-collective/dng.test.mjs` 6/6; the (ε,δ) sweep and collusion/adaptive-composition suites carry the measured bound).
- **Colluding peers (the masking).** A single honest masking partner hides a victim's contribution against a non-colluding seed aggregator (chi-square uniform over the full ring; distinguishing advantage 0.0024 inside the fair-coin band). Above the threshold, full `M-1` collusion degrades the guarantee to local-DP — the standard secure-aggregation bound, stated.
- **Not defended — and stated loudly:** the **release** still requires an honest or threshold aggregator. Build-attack A7 ("a malicious aggregator cannot strip the noise") was **falsified and retracted**: the cut-and-choose publishes the noise count `sumOpening.s` in cleartext (`real-dp-collective/dng.mjs:453`), so a malicious aggregator at the release step is *not* defeated. What is defended is the **noise generation** (no trusted curator for the noise itself), not the release path. The DP is **computational**, not information-theoretic (forced by Biswas-Cormode / Haitner-Omri).

2.6 Malicious reconstructing device / pre-shred exfiltrator

- **Not defended — by design, and this is irreducible.** At recall, exactly one device assembles `K_rec` and holds `M` in plaintext for one turn (the locality moment). A device that is honest-but-curious at *that* moment, or any holder that copies its key-share before a shred, breaks confidentiality. No pure-software layer can prevent this without FHE or trusted hardware; we do not claim to.

2.7 Subpoena / seizure of the recall path (access-pattern adversary)

- **Decentralized Oblivious Recall (FM-042) — defended (access-pattern obliviousness, at an honest cost).** A byzantine swarm serves fixed-size content-addressed erasure-coded *ciphertext* blocks with a **query-independent** access pattern and never decrypts; a signed obliviousness witness proves the on-wire access set is `f(epoch, public schedule)` , not the query — and now also binds the per-node on-wire *order* and the pinned node count. The trace is byte-identical across queries (a trained distinguisher scores advantage 0.003–0.009, below the noise floor); four witness forgeries are rejected (`prototypes/oblivious-recall/` : 21 checks + 8 attacks + red-team probes). **Honest cost, stated not hidden:** full-batch blowup equals catalog size (8× for 8 items); the windowed variant is **not** oblivious across sessions (a persistent adversary intersects per-epoch windows of repeated recalls and de-anonymizes) and must **not** be shipped as oblivious. Obliviousness here is access-pattern, **not** confidentiality-during-compute; block existence/count/sizes are public by design (that is what makes the witness checkable).

2.8 Cross-session leakage (a sensitive session contaminating a later one)

- **Cross-Session Grounding Closure (FM-043) — defended (directional non-flow over commitments).** Given a HIGH (sensitive) session `S1` and a later LOW session `S2` , a signed receipt proves no memory entry grounded in `S1` appears in `S2` 's grounding set. Soundness is "re-derive, don't trust": the verifier recomputes the HIGH and LOW commitment sets from the two owner-signed session receipts and intersects *those*, ignoring attacker-supplied arrays; directionality is by strict epoch ordering with no numeric-coercion hole (`prototypes/cross-session-closure/` : 14/14 + 11/11 + a 20-attack red-team). **Honest boundary (not a bug):** disjointness is over **commitments**, not **values** — re-committing the same value with a fresh

salt is a different commitment and verifies, so the proof is "custody did not *re-supply* the committed entry," not "the same value never reappeared," and not a model-internal non-flow claim.

2.9 Colluding heirs / unauthorized inheritance (Soul Will — core proven, trigger roadmap)

- **Soul Will (FM-035) — core defended; liveness trigger is roadmap.** Memory passes only per pre-signed, scope-partitioned consent, gated by a quorum of heirs. The core holds: scope-partition (cross-scope graft and duplicate-shareId collision refused at the gate), heir-pin, dead-hand (pre-trigger unseal refused), BURN (post-trigger + resurrection refused), and confidentiality (no attack produced a false plaintext; the Shamir key-commitment is the hard backstop) — `prototypes/soul-will/` 25/25 baseline + an 11-attack and a 12-attack red-team held, after a one-iteration availability fix (bind sorted share-id digests into the canonical will hash; return `{ok:false}` instead of throwing). **Honest limit:** the BURN scope is information-theoretic unrecoverability *if* a true threshold drops shares, **not** physical destruction; the **death-trigger needs a liveness oracle** and is not yet a proven layer.

---

3. Per-layer summary: provably defended vs not

Layer / capability	Provably defended (test)	Provably *not* (honest boundary)
L1 TRM	At-rest seizure → 0 bits of `M` / `s_grp` (T1); consent unforgeable by a decryptor (T2, 17 forgeries)	Plaintext-at-recall on one device; `|M|` length leak; malicious reconstructor / pre-shred exfiltrator; single-coordinator Schnorr, not FROST
L2 Release Ledger	Rollback/truncation/equivocation vs a **pinned** head; non-membership of release; query-bound non-supply	First-contact fork acceptance (needs pinned head/witness); cross-party non-equivocation (needs witness); stateless freshness
L3 QRSM	Non-materialization (exactly 2 files at rest); possession-root survives powered-off seizure	Confidentiality-during-compute (RAM plaintext while quorum present; swap-spillable); modeled authenticator
L4 Forget-by-reprojection	Symbolic: forgotten fact absent by construction; same-set re-projection byte-identical (26/26 + 16-attack)	Neural adapter: no bit-for-bit cross-HW regen (provenance-bounded only); **stale cache leaks the forgotten fact**; not byte-erasure on uncontrolled replicas
L5 FlowMemory-DNG	Real (ε,δ) bound with a monotone σ knob; no trusted curator for the *noise*; free-rider rejected	Release needs honest/threshold aggregator (noise count cleartext, `dng.mjs:453`); secure-agg collusion threshold; computational, not info-theoretic DP
Grounding Receipt (FM-039)	Answer = exactly a committed memory subset + closure; value-hiding (17/17 + 20-attack)	Custody-side supply, not what the model internally used; needs on-device/MCP/TEE to bind the real prompt
Stapled Transcript (FM-045)	Ordered-Merkle cross-bind: off-memory/dropped/reordered entry rejected, value-blind (13 + 38 attacks)	Binds committed segments + order, **not interstitial glue bytes** (needs a ZK opening proof); inherits the compute receipt's left-half trust model
PAIR (FM-048)	One owner-signed turn record: used + ran-here + never-supplied, disjoint, one epoch (28/28 + 13/13)	Pins (session_id + release root) must come from a trusted anchor, not the record; inherits every leg's limit
Memory Immunity	Reprojection from enrolled-author-signed ops only; forged op quarantined, closure breaks (8/8 + 19 + 16 + 13)	Attributability, not semantic truth; live-query injection + side-channel poison out of scope; roster-hash pin required
Provable Consolidation (FM-051)	lineage-IN == destroyed-OUT with a *real* key-destruction check (random DEK, no re-derivation) + signed custody (15/15 + 11 attacks)	Not semantic faithfulness; no-prior-leak unprovable; destruction of copies you do not control; seed-level recovery if seed + label both survive owner-side
Oblivious Recall (FM-042)	Byte-identical access trace; order + node-count bound; swarm never decrypts (21 + 8 attacks)	Full-batch blowup = catalog size; windowed variant NOT oblivious across sessions; access-pattern only, block sizes public
Graded Consent (FM-040)	Over-tier/foreign-purpose/over-cap/sub-quorum cannot mint a receipt; auditor re-folds (15 + 16 + 16 + 18 attacks)	Governs access/usage accounting, not confidentiality-during-use; grades are declared tiers, not a confidentiality proof
Soul Will (FM-035) core	Scope-partition, heir-pin, dead-hand, BURN, confidentiality (25/25 + 11 + 12 attacks)	BURN = info-theoretic, not physical; death-trigger needs a liveness oracle (roadmap)

---

4. Honest negatives carried in the threat model

Two results are recorded as **negatives** so the proven set is not over-read. They are evidence of the discipline, not gaps quietly shipped.

- **USPIR sublinear DPF recall (FM-046) — real but PARKED.** A two-party BGI distributed-point-function read over two user-owned ciphertext replicas gives genuine sublinear recall (O(log N) up + O(1) down) with **no FHE/TEE/trust**, and a single-key view does not localize the queried index (distinguisher advantage ≈ 0.009). It is parked, not claimed, for three honest reasons

(`prototypes/uspir-dpf/uspir.mjs`): the crossover is N ≥ 8 (not N ≥ 4); it is **static single-writer only** (a divergent replica fails GCM auth, so it does not cross the multi-writer wall); and the DPF path carries **no recomputable obliviousness witness**. Closest prior art: BGI (EUROCRYPT 2015), DPF-PIR over erasure-coded storage, Plinko (eprint 2024/318).

- **Witnessed-sublinear-witness (FM-047) — wall held.** The attempt to give the sublinear read the recomputable obliviousness witness USPIR lacks (a node reveals its single DPF key; an auditor re-expands over all N to confirm a full query-independent scan) was **broken by the independent red-team** (`prototypes/uspir-dpf/witness.attack.test.mjs`). The witness is *self-issued and unbound to the live query*: because the reply is recomputable from public blocks + a self-chosen key, a node can fabricate a consistent `(key, reply)` pair entirely offline — a node that served zero queries passes, and a key for an index never queried passes. **Plain recomputation cannot bind a self-issued offline witness to runtime behavior.** What genuinely held (value-hiding, single-key index-hiding, value-level recomputability, sound Merkle commitment, dense scan-binding) is recorded; the required fix turns it into a *logged interactive protocol* or needs a SNARK/TEE. This is one of the precisely-named uncracked walls, not a shipped feature.

---

5. The full honest-limits list (binding)

These travel with every claim in every document, white paper and marketing alike. They are the reason the proven claims are trustworthy.

1. **Locality, not FHE (plaintext-at-recall).** At recall the projection is plaintext in RAM on one device for that turn. On-device privacy is *locality*, not homomorphic computation. The stack defeats cold theft, seizure, and operator access — **not** a compromised live endpoint or a malicious reconstructing device.
2. **Non-materialization is protocol-level and swap-spillable.** "Never materialized" (L3) means no *durable derived store* is written; the op-log ciphertext is durable, and the RAM projection is OS-swap- and crash-dump-spillable. Pure software cannot scrub RAM/swap, and JS strings are immutable. Do not compress this into "memory that cannot exist at rest."
3. **Forget = attestation + information-theoretic unrecoverability, not physical destruction.** Threshold-forget is info-theoretically irreversible once `n − k_r + 1` holders genuinely shred their key-shares. The *proof* is an attestation-of-destruction; it cannot prove bytes were erased on a malicious replica that secretly kept a copy. For the neural adapter, forgetting is provenance-bounded ("derived from exactly this op-set, which excludes X"), not regeneration.
4. **Freshness anchor required for offline proofs.** A stateless verifier given only a snapshot can be fooled by replaying an old valid state (the R3 break). Sound offline forget / non-supply / grounding proofs require a pinned monotonic epoch, a fresh quorum-co-signed nonce, or an external witness. The verifier MUST pin its latest checkpoint.
5. **DP is computational, and the release needs an honest/threshold aggregator.** The (ε,δ) bound rests on commitment hiding + binding (Biswas-Cormode / Haitner-Omri force computational, not information-theoretic, DP). The *noise generation* needs no trusted curator; the *release* does. The ZK proof is school-strength cut-and-choose, not a production SNARK.
6. **Collusion thresholds.** TRM/QRSM read requires `k` -of- `n` ; below it, nothing. The DNG masking hides individuals only below its collusion threshold (full `M−1` collusion → local-DP). At least one honest noise contributor is required, with bound tightness proportional to the honest fraction.
7. **Metadata leakage.** Manifest existence and size, scopes, recall timing, ciphertext length (`|M|`), and item count all leak. For Oblivious Recall, block existence/count/sizes are public **by design** (that is what makes the witness checkable). The signed receipts authenticate *content* and *ordering*, not the absence of side-channel structure.
8. **Cache hazard (L4).** Re-projection after a forget must be *from scratch*. A stale pre-forget cached summary still contains the forgotten fact; incremental caches are a correctness-and-leakage bug, not an optimization.
9. **Replay / determinism holds only for the symbolic, deterministic layer.** The replayable recall receipt reproduces a byte-identical surfaced set only for deterministic ranking; semantic vector recall degrades the guarantee to "reproducible given this exact model + index version." The neural adapter is content-addressed provenance, not bit-for-bit cross-hardware regeneration.
10. **The staple binds segments + order, not glue bytes (new).** The Stapled Transcript cross-binds the *committed used-entry segments* and their *order* between the grounding and compute halves, so no off-memory or dropped or reordered entry

survives. It does **not** bind the interstitial / glue bytes a node may place *between* segments; binding the exact full prompt bytes to a value-blind verifier needs a ZK opening proof (a named research upgrade, out of scope). The staple also inherits the compute receipt's left-half trust model (same-engine vs third-party) and is not confidentiality-during-compute.

11. **PAIR's session pins come from a trusted anchor, not the record (new).** PAIR's per-interaction non-supply leg is sound only because the verifier independently pins both `pinnedSessionId` and `pinnedReleaseRoot` for the interaction **out-of-band** (e.g. from the release-ledger checkpoint), never from the record under audit. The cryptographic legs prevent forging contents and the decoy-session splice; the *which receipt belongs to which session* binding rests on the owner signature + the shared `session_id` + the externally-pinned release root. Without both pins, `verifyPAIR` fails closed.
12. **Memory Immunity needs the roster *content hash* out-of-band (new).** The verifier MUST pin, out-of-band, the roster **content hash** (SHA-256 over the canonical roster body — not just the epoch integer), the candidate-log head, and the epoch — all three required, fail-closed. Pinning the epoch number alone is insufficient: the owner can sign a *second* roster at the same epoch that enrolls a poisoner, and that fork carries a valid owner signature and the right epoch. Absent the content-hash pin, the verifier rejects rather than falling back to epoch-only.
13. **Consolidation proves unrecoverability of the exported state, not of copies you do not control (new).** Provable Consolidation proves `lineage-IN == destroyed-OUT` with a *real* key-destruction check — the per-ephemeral key is gone from the authenticated custody **and** the ciphertext does not decrypt under any key the snapshot carries (random per-ephemeral DEKs, so there is no re-derivation path). It does **not** prove no plaintext copy was exfiltrated *before* consolidation, nor destroy a copy held on a replica you do not control. A residual **seed-level** caveat holds for any HKDF-derived (non-random-DEK) variant: unrecoverability is relative to the *exported* state — if both the master seed and the purpose label survive on the owner side, the owner could re-derive (the FM-031 caveat). The proven FM-051 prototype avoids this by using random DEKs; the limit is stated so no HKDF variant is over-read.
14. **Novelty is fusion-grade, not a new primitive** — and three louder ideas (Entangled Memory, Semantic Cones, holographic privacy) were *falsified to features* and are **not** claimed. Entangled Memory's "mutual uselessness" was one-directional and its presence-witness decorative; Semantic Cones' confidentiality reduced to a non-crypto embedding-binning question with structural leak bands; holographic memory's energy statistic was a membership oracle (AUC 0.84 at N=16). The frontier of brand-new primitives is literature-occupied (§6).

---

6. Closest prior art (every novelty named)

Novelty here is composition at the application altitude — user-custody, per-turn, offline-verifiable — not a new cryptographic primitive. The closest published work for each axis:

- **Private retrieval / encrypted AI memory:** Opal (arXiv 2604.02522, TEE + ORAM user-memory confidentiality); Anuma (encrypted cross-model vault, on-chain *logging* in place of proof). FlowMemory's wedge: a memory node that *never decrypts* + an offline-verifiable receipt, vs an enclave that decrypts or a log that records permissions.
- **Signed-op admission / provenance:** Provenance Gates (arXiv 2605.13471) and MemLineage (arXiv 2605.14421) for signed-op admission and per-principal lineage; Portable Agent Memory (arXiv 2605.11032) for content-addressed + Merkle + capability memory. Wedge: sovereign custody + quorum write-admission + proof-of-exclusion, not label-and-gate.
- **Proof-of-non-membership / forgetting:** ZK-APEX (2512.09953), ZKPoU, Verifiable Federated Unlearning (2510.00833) — all **model-side** weight unlearning. Wedge: **ledger-side** absence over user-owned content-addressed memory + an offline static verifier (novelty here is retired to this wedge; we do not claim "nobody proves absence").
- **Compute receipts / proof-of-inference:** c0mpute / Shard and Lagrange DeepProve, VeriLLM, Gensyn Verde, Hyperbolic PoSP — fast and verifiable for *correctness*, but every proof requires the prover to hold the input in **plaintext** (the receipt hides data from the verifier, never the prover). FlowMemory supplies the structurally-missing **input-provenance** operand (Stapled Transcript / PAIR) that the compute sector cannot produce on its own.
- **Grounding / commit-and-audit:** CommitLLM (model-side commit), zkRAG / VeriRAG / V3DB (retrieval over a *service-owned* DB). Wedge: closure bound to a **user-owned** memory subset.

- **Crypto-shred / certified deletion:** Vanish (USENIX Sec 2009) and "Deleting Secret Data with Public Verifiability" (eprint 2014/364, the standard that a deletion must be publicly verifiable via key destruction, not asserted by a tombstone) — which the Provable Consolidation key-destruction check explicitly honors.

---

7. The meta-lesson, with this cycle's breaks as evidence

Across the program's iterations and roughly 260 build-and-falsify agents, the breaks sort by *where they lived* into one finding:

> **Every break across the program lived in composition glue or a fail-open default — never in a red-teamed cryptographic primitive.**

The earlier evidence is uniform: T2 — the genuinely novel core — never broke (17 forgeries on first contact). Every other historical break was integration or a footgun: TRM's T1 (an unsalted cleartext commitment in the envelope), T3 (unauthenticated op labels trusted as free-standing), and T4 (forget bound to the *consent* quorum instead of the recall-*key* holders); FM-018 non-supply (a queried hash never bound to the absence key, a one-line fix); FM-002 (a missing authenticated head, then the fundamental freshness limit); the DNG A7 overclaim (a cleartext field in the release envelope); and the FM-016/019/023 **fail-open defaults** (a bare verifier that defaulted to accept, content-binding enforced only when a hash was supplied, malformed inputs that *threw*) — every one re-hardened to **fail closed**.

This build cycle produced four fresh breaks while building the newest layers, and every one fits the pattern and is now closed:

- **The witness break (FM-047).** The recomputable obliviousness witness was **self-issued and unbound to the live query** — a node could fabricate a consistent `(key, reply)` pair entirely offline, certifying *a property of a value* rather than *that the node did the work on the real query*. This is the program's cleanest statement of the rule: a self-consistency check mistaken for an execution attestation is glue, not a primitive. Recorded as a negative (§4); the fix requires a logged interactive protocol or a SNARK/TEE.
- **The PAIR session splice.** A custodian that *did* release `X` this turn could try to satisfy the non-supply leg with a genuine Empty Witness built against a **decoy session** that never released `X`. The break lived entirely in the cross-leg binding, not in any leg's crypto. Closed by pinning `session_id` + the release root **out-of-band** and requiring every witness to be session-bound (§5.11): the decoy `session_id` ≠ the pinned one (rejected), and relabeling to the real session makes the absence proof fail against the real release root where `X` *was* released (rejected).
- **The roster fork.** The immunity verifier originally pinned only the epoch *integer*; the owner could sign a **second roster at the same epoch** that enrolls a poisoner — a valid owner signature with the right epoch. A fail-open default in the freshness pin, not a flaw in the authorship predicate. Closed by requiring the roster **content hash** (plus head + epoch), all three, fail-closed (§5.12).
- **Tombstone-without-destruction and seed-rederivation.** The first consolidation receipt (FM-049) proved destruction by checking a signed REMOVE *tombstone* and absence from the recomputed survivors — forgeable, because a custodian can append REMOVE ops while leaving the AES keys live and the ciphertext decryptable. A related break: deriving the per-ephemeral key via `HKDF(seed, "ephemeral/<sid>")` over a *retained* seed left a re-derivation path, so a "destroyed" key could be recomputed. Both are glue / key-schedule seams. Closed in FM-051 by random per-ephemeral DEKs (no derivation path), a *real* key-destruction check (key gone from authenticated custody **and** ciphertext does not decrypt), and an owner-signed `custody_seal` so an omitted live key cannot be faked off-snapshot (§5.13).

Two operating principles follow, and they are the product discipline:

1. **Proven primitives are not a proven system.** The Shamir split, threshold-Schnorr, RFC-6962 ledger, SMT non-membership, BGI DPF, and discrete-Gaussian sampling were all sound off the shelf. The integration layer that wires them together needs *its own* adversarial test suite, because that is precisely where the breaks were — and where this cycle's four breaks were.
2. **No security check was ever weakened to pass a test.** Across every hardening pass, verifiers were only ever made *stricter* / fail-closed. The documented honest negatives — the tail-truncation break, the R3 stale-seal replay, the A7 noise-count retraction, the FM-047 self-issued-witness wall, the Entangled/Semantic/holographic downgrades — are recorded in the

dossier as *refuted*, not quietly shipped. (One current artifact of strictness, recorded plainly: five assertions in `immunity.honesty.attack.test.mjs` now report "FAIL" because the roster-hash hardening fails closed *before* the condition those probes were written to observe; the authoritative roster red-team is green at 13/13 with zero soundness breaks, and the probe text says "fail-closed" — a stale probe against a stricter verifier, not a soundness regression.)

The honest summary of the threat model is therefore not "this is unbreakable." It is: **the primitives held under fire; the composition is where danger lives; the composition now has its own red-team and fails closed; the two negatives (USPIR parked, FM-047 wall) are stated as negatives; and every place the architecture stops is written down in §5 in the same plain language as the places it holds.**

Proof and test evidence

> **Scope.** This document is the evidentiary spine of the white paper. It states the verification methodology, lists the full test matrix, and — for every proven claim — names the prototype file and the falsifying assertion that would have caught the claim being false if it were false. It also records the **honest negatives**: the breaks the program found and did not paper over, the two recall results that are parked or walled rather than claimed (USPIR, FM-047), and the three primitive-grade ideas that were honestly falsified to features. This is the credibility core: nothing in the rest of the white paper is asserted that is not either backed here by a prototype plus a passing adversarial suite, or labelled a stated limit.
> **Honesty contract (binding on every line below).** Signatures and hashes prove **custody, integrity, ordering, consent, minimization, grounding, non-supply, exclusion, and forgetting** — never confidentiality-during-compute, never FHE, never what a model internally attended to, never semantic truth. On-device privacy is **locality, not homomorphic computation**. We never claim a server or model "can't read" data on a path where it can. Offline forget / non-supply / grounding proofs require a freshness anchor. The DP *release* still needs an honest or threshold aggregator (the noise *generation* does not). Novelty is **fusion-grade and application-grade**, not a new cryptographic primitive; the closest prior art is cited wherever a novelty is claimed (§7).

All prototype paths below are relative to `research/prototypes/`. The full iteration log, idea registry, and verification ledger are in `research/AI_MEMORY_LAYER_DOSSIER.md`; the adversary models that pair with these proofs are in `03_threat_model_and_honest_limits.md`. Every suite named in §3 and §4 was re-run live this session on **Node v24.15.0** and exits `0` unless explicitly marked.

---

1. Verification methodology

The discipline, not the result, is what makes the proven claims trustworthy. Four rules held across the entire program (24+ build-and-falsify iterations, roughly 260 agents).

1. Self-contained Node, built-ins only. Every prototype runs as plain Node (`node:crypto`, `node:assert`, `node:fs`, `node:child_process`) — no npm dependency, no network call, no test framework. A reviewer reproduces any result with one command (`node <suite>.mjs`) and an exit code: `0` means the property held; non-zero means a check failed loud. This removes the two commonest sources of a false green — an over-mocked harness and a supply-chain dependency silently doing the work the prototype claims to do. The cross-process determinism audit for forget-by-reprojection goes further and spawns *separate* `node` processes (`reprojection-forget/attack.test.mjs`, section A), so any hidden per-process state (a randomized hash seed, an address, `Date`, `Math.random`) would surface as a digest mismatch.

2. Build / independent-red-team separation. Each prototype was written by one agent and attacked by an *independent* agent that did not write it. The attacker imports the **unmodified** prototype and adds only an attack file — for example `discovery-shield/decryptor-attack.test.mjs` opens with the explicit note that nothing in the prototype is modified, that the file only imports and attacks it, and the combined suite confirms both legs come from the proven prototypes imported unmodified. Where the attacker needed elliptic-curve arithmetic to craft a precise forgery, they re-implemented P-256 from scratch *inside the attack file* (`trm/trm.attack.test.mjs`, `discovery-shield/decryptor-attack.test.mjs`) rather than calling the prototype's curve code, so the attack cannot accidentally borrow the defense. Where a newer capability staples proven legs (PAIR, Stapled Transcript, Cross-Session Closure), the legs are imported byte-for-byte and a `git diff` over them is asserted empty; the new file adds only an outer gate.

3. No check was ever weakened to pass. When a red-team found a break, the fix only ever made the verifier **stricter / fail-closed** — it never relaxed an assertion to turn a red suite green. The clearest example is the crypto-shred re-attack, which found three malformed-input shapes that *threw* (a fail-open foot-gun if a caller wraps the call in `try/catch`); the fix made the verifier return `{ok:false}` uniformly. Several suites also carry explicit "teeth checks" — a positive control that the harness *can* recover the secret when the defense is removed — so a vacuously-passing test is detectable (e.g. the TRM T1 re-attack re-adds the stripped commitment and confirms `M` is then recoverable, `trm/trm.attack.t1b.mjs`).

4. Refute before claiming; record every break. The loop's bias was to falsify. Breaks are logged in the dossier's findings log and verification ledger as *recorded refutations*, never deleted. The recurring meta-lesson — confirmed across iterations 4, 6, 7, and again on this cycle's four fresh breaks — is that **every break lived in composition glue or a fail-open default, never in a red-teamed primitive**. Proven primitives are not a proven system; the integration layer needs its own adversarial tests. That is exactly why the capstone and composite suites (TRM T1–T4, Discovery Shield, PAIR, Memory Immunity) each carry their own integration-level red-teams.

---

2. The proven set at a glance

Three tiers, in the order a reviewer should read them: the substrate that everything stands on, the emergent capabilities composed from it, and the supporting layers. Each line names the suite(s) and the single property each suite would have falsified.

- **Core substrate.** L1 TRM (two-secret spine), L2 Release Ledger + the Empty Witness, L3 QRSM (recomputed-never-stored), L4 forget-by-reprojection, L5 DNG (real-DP collective).
- **Emergent capabilities (compositions nobody ships).** Grounding Receipt; Stapled Transcript; PAIR (Provable AI Interaction Record); Memory Immunity Receipt; Provable Consolidation; Discovery Shield; Cross-Session Grounding Closure; Decentralized Oblivious Recall; Graded Consent Spend-Trace; Soul Will; Dead-Hand Recall (the $58 hardware node's possession-root, in software-sim).
- **Honest negatives (recorded, not claimed).** The single-key forget-proof (refuted by tail-truncation, then by stale-seal replay); USPIR sublinear recall (real but parked, no live-execution witness); FM-047 witnessed-sublinear-witness (a wall that held — a self-issued offline witness is fabricable); and three primitive-grade ideas falsified to features (Entangled Memory, Semantic Cones, holographic privacy).

---

3. Test matrix (re-run live this session, Node v24.15.0)

Every suite below was executed from its prototype directory this session. The result column quotes the suite's own terminal line.

3.1 Core substrate

Layer / capability	Primary suite	Independent red-team	Result (this session)
L2 Memory Ledger	`memory-ledger/ledger.test.mjs`	`memory-ledger/attack.test.mjs`, `memory-ledger/break.test.mjs`	`21 passed` / `18 held, 0 broke` / `36 held, 0 broke` (incl. n=1..40 sweep)
L2 Non-membership SMT	`non-membership/test.mjs`	`non-membership/attack2.mjs`	`8 checks passed` / `15 attacks rejected`
L1 Threshold-shared memory	`threshold-shared-memory/shamir.test.mjs`	`threshold-shared-memory/attack2.test.mjs`	`11 checks passed` / `16 held, 0 broke`
L2 Quorum write-admission	`quorum-write-admission/quorum.test.mjs`	`quorum-write-admission/quorum.redteam2.test.mjs`	`10 checks` / `12 red-team checks survived`
L2 Quorum crypto-shred	`quorum-crypto-shred/quorum_crypto_shred.test.mjs`	`quorum-crypto-shred/quorum_crypto_shred.attack2.test.mjs`	`10 checks passed` / `10 attacks defeated` (fail-closed on malformed input)
L2 Proof-of-non-supply (Empty Witness core)	`proof-of-non-supply/nonsupply.test.mjs`	`proof-of-non-supply/nonsupply.reattack.mjs`	`13 checks` / `14 attacks defeated, no soundness break`
L1 TRM (T1–T4)	`trm/trm.test.mjs`	`trm/trm.attack.test.mjs` (T2), `trm.attack.t1b.mjs`, `trm.attack.t3.mjs`, `trm.t4.attack.mjs`	`22 assertions, all passed` / **17 consent forgeries rejected**
L3 QRSM	`qrsm/qrsm.test.mjs`	`qrsm/qrsm.replay.test.mjs`	`6 checks passed`
L4 Forget-by-reprojection	`reprojection-forget/reproj.test.mjs`	`reprojection-forget/attack.test.mjs`	`26 checks passed` / `16 adversarial checks passed` (incl. cross-process determinism, §A)
L5 FlowMemory-DNG	`real-dp-collective/dng.test.mjs`	`dng.dp.test.mjs`, `dng.kcollusion.attack.test.mjs`, `dng.adaptive.attack.test.mjs`	`CORE: 6 checks passed` (the (ε,δ) sweep + collusion/adaptive suites carry the measured bound)

3.2 Emergent capabilities

Capability	Primary suite	Independent red-team	Result (this session)
Grounding Receipt (FM-039)	`grounding-receipt/grounding_receipt.test.mjs`	`grounding-receipt/grounding_receipt.attack.test.mjs`	`17 checks passed` / `20 attacks resisted, 0 breaks`
Stapled Transcript (FM-045)	`stapled-transcript/stapled.attack.test.mjs`	`stapled.crossbind.adversary.test.mjs`, `stapled.redteam.attack.test.mjs`	`13` / `16` / `22` attacks resisted, 0 breaks (51 total)
PAIR — Provable AI Interaction Record (FM-048)	`pair/pair.test.mjs`	`pair/pair.adversary2.test.mjs`, `pair/pair.adversary3.probe.mjs`	`28 checks passed` / `13 checks passed` (every attack resisted, honest control verifies)
Memory Immunity Receipt	`memory-immunity/immunity.test.mjs`	`immunity.attack.test.mjs`, `immunity.soundness.attack.test.mjs`, `immunity.roster.redteam.test.mjs`, `immunity.honesty.attack.test.mjs`	`8` / `19` / `16` / `13` passed, **soundness breaks = 0**; honesty probe: `0 breaks, 0 overclaims` (see §5.4)
Provable Consolidation (FM-051)	`consolidation/consolidation.test.mjs`	`consolidation/adversary.attack.test.mjs`	`15 checks passed` / `11 attacks, 0 broke`
Discovery Shield (FM-034)	`discovery-shield/combined.test.mjs`	`discovery-shield/decryptor-attack.test.mjs`, `attack.binding.test.mjs`	`11 checks passed / 9` + `9` attacks, shield survived
Cross-Session Grounding Closure (FM-043)	`cross-session-closure/cross_session_closure.test.mjs`	`cross_session_closure.attack.test.mjs`, `cross_session_closure.fresh_attack.test.mjs`	`14 checks passed` / `11` + `20` attacks resisted, 0 breaks

Capability	Primary suite	Independent red-team	Result (this session)
Decentralized Oblivious Recall (FM-042)	`oblivious-recall/oblivious_recall.test.mjs`	`oblivious_recall.attack.test.mjs`, `.redteam.test.mjs`, `.redteam2.test.mjs`, `.redteam3.test.mjs`	`21 checks passed` / `8 attacks defended` + 3 red-team probes green
Graded Consent Spend-Trace (FM-040)	`expedition-graded-consent-spend-trace-the-a/spendtrace.test.mjs` (+ grade-laws)	the 16-attack battery + an 18-attack grade-binding re-attack	grade-laws `15/15`, spend-trace `16/16`, battery `16/16`, binding `18/18` (proven after two fix rounds, §6)
Soul Will (FM-035) core	`soul-will/soul_will.test.mjs`	`soul-will/soul_will.attack.test.mjs`	`25 checks passed` / 11-attack + 12-attack red-team held; confidentiality always held
Dead-Hand Recall (FM-044, $58-node possession-root, software-sim)	`dead-hand-recall/dead_hand_recall.test.mjs`	`dead-hand-recall/dead_hand_recall.attack.test.mjs`	`14 checks passed` / key property held under 11 fresh attacks

3.3 Honest negatives (recorded, not claimed)

Result	Suite(s)	What the suite shows
Single-key forget-proof — **refuted**	`forget-and-prove/attack.test.mjs`, `attack_round2.test.mjs`	Tail-truncation forges a positive forget proof (A/B); a signed head/length is *still* broken by stale-but-genuine seal replay (R3) — a fundamental freshness limit (§5.1)
USPIR sublinear DPF recall — **real but parked**	`uspir-dpf/uspir.test.mjs` (`14 checks passed`)	Genuine sublinear recall, single-key index-hiding (advantage ≈ 0.009), no FHE/TEE/trust — but crossover N ≥ 8, static single-writer only, and **no recomputable obliviousness witness on the DPF path** (the suite's own check (F) records this honest downgrade)
FM-047 witnessed-sublinear-witness — **wall held**	`uspir-dpf/witness.test.mjs` vs `witness.attack.test.mjs`, `witness.bind.test.mjs`, `witness.keymal.test.mjs`	The builder's value-level suite passes 14/14, but the independent red-team **reports a break** (the expected, asserted outcome): the witness is self-issued and unbound to the live query, so a node fabricates a consistent `(key, reply)` pair entirely offline (§5.2)
Forget Certificate (FM-038) — soundness gaps found, then closed	`forget-certificate/forget_certificate.test.mjs`, `attack_fresh.test.mjs`	Two soundness gaps were found and fixed fail-closed; the suite is now `15 checks passed` and the four fresh attacks are correctly rejected (§6)

---

4. Proven claims, with the falsifying assertion behind each

For each load-bearing claim, the named test would have *failed* if the claim were false. Citations are the file plus the test the assertion lives in (test names, not line numbers, so the citation survives edits).

4.1 Consent is unforgeable even by a decryptor (TRM T2)

The defining novel property: a party holding the recall key `K_rec` *and* the plaintext `M` still cannot forge a valid proof that the user's devices consented to a read. **Being able to decrypt and being able to prove consent are cryptographically independent events.**

- **Premise established as a positive, not assumed.** `trm/trm.test.mjs` (T2 block) first asserts the attacker genuinely reconstructed `K_rec` and decrypted `M` (`Rplaintext === SECRET`). The test only becomes meaningful because the attacker actually holds the plaintext.
- **The falsifying assertion.** `forged === false` for an attacker holding the plaintext plus one consent share (`< k_a`); a consent signature hand-fabricated from a `K_rec` -derived scalar must not verify. The dedicated red-team `trm/trm.attack.test.mjs` held **17 distinct forgery attempts** — free-floating `(R, z)` with no partials, back-solved partials, a 4000-iteration Fiat-Shamir grind, single-secret completion, cross-message transplant, and partial mix-match — each asserting the verifier returns false.
- **Why it holds.** `verifyConsentSignature` (`trm/trm.mjs`) checks *both* the aggregate equation `z·G == R + c·P_grp` *and* every per-partial `z_i·G == R_i + c·λ_i·P_i` against the **enrolled** `P_i`, and rebuilds `(R, z)` from the partials — so the classic Schnorr "self-fulfilling" `(R, z)` (pick `z`, set `R = z·G − c·P_grp`) is rejected by the partial cross-check (the same defense holds in `discovery-shield/decryptor-attack.test.mjs`, consent legs).
- **Positive control + binding.** A genuine 2-of-3 aggregate *does* verify, and the same signature does **not** verify over a different message (no cross-message replay).

4.2 Recomputed-never-stored, possession-rooted (L3 QRSM)

- **Non-materialization.** `qrsm/qrsm.test.mjs` scans the at-rest directory after a session and teardown and asserts exactly two files remain — the ciphertext op-log and the public enrollment — and no derived index, embeddings, summary, or plaintext. Replay is rebuilt in RAM from the op-log (`qrsm/qrsm.replay.test.mjs`).
- **Possession-root is load-bearing, not a shares-at-rest collapse.** The red-team built the TRM "shares-at-rest" strawman (real Shamir shares on disk → seize `k` powered-off devices → recombine) and confirmed it collapses, then showed QRSM does **not**: it persists only public offsets `off_i = share_i XOR HMAC(tokenSecret_i, salt)` with `tokenSecret_i` *off-device*. No pair or triple of offsets interpolates the seed, 50k seed guesses miss the commitment, a byte-grep finds no key/seed/share at rest, and 30k random keys never decrypt (the GCM backstop).
- **Honest scope (stated, not hidden).** At-rest only — not confidentiality-during-compute (the projection is plaintext in RAM while the quorum is present, and swap is spillable); the authenticator is *modeled* (an HMAC closure standing in for WebAuthn-PRF), so the guarantee holds iff real hardware keeps `tokenSecret` off-device. The $58 hardware node (`HARDWARE_BUILD.md`) and `dead-hand-recall/` elevate this from software-decorative to hardware-enforced.

4.3 Forgetting that actually works (L4 forget-by-reprojection)

- **Symbolic forget is gone-by-construction.** `reprojection-forget/reproj.test.mjs` : delete the source op, re-project over the surviving op-set, and the forgotten fact is provably absent from the index / facts / summary while the kept fact survives (surgical). Same op-set → **byte-identical** projection (a pure function → a deterministic content address). This is a genuine deterministic alternative to machine unlearning for the queryable symbolic memory — no "did the weights forget?" uncertainty.
- **Cross-process determinism is proven, not assumed.** `reprojection-forget/attack.test.mjs` (section A) spawns separate `node` processes and asserts the projection digest is identical across them — ruling out any per-process state leak.
- **Provenance binding for the neural adapter.** From `{op-log, receipt}` alone, a third party verifies the adapter derives from *exactly* the surviving op-set and that a forgotten op is absent; tamper, forgery, staleness, op-set swap, unsealing, and truncation are all rejected. The honest boundary (§5) is that bit-for-bit cross-hardware regeneration of the adapter fails — the adapter is content-addressed provenance, not regeneration — and stale caches must be purged or the forgotten fact leaks.

4.4 Proof of a negative — the Empty Witness (proof-of-non-supply)

- **Query-binding is the load-bearing check.** `proof-of-non-supply/nonsupply.test.mjs` plus the re-attack `nonsupply.reattack.mjs` (14/14 held): the verifier recomputes `releaseAddr(contentHashX)` and requires `absence.key === releaseAddr(contentHashX)` , so a malicious custodian cannot answer "is X absent?" with a valid absence proof for a *different* never-released Y. This one-line binding closes the original composition break (§6).
- **Append-only soundness.** RFC-6962 consistency from the pinned checkpoint shows the log only grew, so retroactive insertion of `X` is detected; rollback, key-substitution, and cross-session relabel are all rejected. The honest boundary: this proves custody-side **non-release**, not what a remote model was conditioned on, and the verifier must pin its latest checkpoint (freshness).

4.5 Exactly what was supplied, and nothing off-memory (Grounding Receipt)

- **Closure is the load-bearing invariant.** `grounding-receipt/grounding_receipt.test.mjs` (17/17) binds an answer to *exactly* a committed subset of the user's own memory: subset-membership of each used entry against the signed SMT root, **plus** `input_hash == hash(canonical union of opened entries)` so nothing off-memory was smuggled in. The verifier recomputes `input_hash` over exactly the membership-proven set, so the hashed set is identical to the proven set.
- **The falsifying suite.** `grounding_receipt.attack.test.mjs` resisted **20 fresh attacks** with zero breaks — closure-desync, off-memory smuggle, SMT proof malleability, freshness/epoch confusion, value-revelation — values never exposed. Honest boundary: this proves the custody layer *supplied* exactly these entries, not what the model internally did with them (needs on-device/MCP/TEE to bind the actual prompt bytes).

4.6 The input-provenance half of a compute receipt (Stapled Transcript)

- **The kill-condition is refused, on purpose.** The tempting cheat — coercing both sides to hash the same artificial commitment-set — would make the staple a redundant second signature over the grounding receipt's own `input_hash` and prove nothing about what a GPU consumed. The prototype explicitly rejects it.
- **The real binding.** The compute node forms its prompt by concatenating the released, consented segments in order and computes `context_hash = orderedMerkleRoot([C_1..C_n])` over exactly the per-segment commitments. `verifyStapled` recomputes that ordered Merkle root over the grounding receipt's own used-entry commitment set (which it already holds, in order) and fails closed unless they match. An off-memory entry, a dropped entry, or a reorder all change the root.
- **The falsifying suites. 51 independent attacks** across `stapled.attack.test.mjs` (13), `stapled.crossbind.adversary.test.mjs` (16), and `stapled.redteam.attack.test.mjs` (22) were resisted with zero breaks — including encoding-malleability, a redundant-signature kill-condition, and the Bitcoin CVE-2012-2459 duplicate-leaf attack. Honest boundary: it binds the committed segments and their order, **not** the interstitial glue bytes between segments (a named ZK-opening-proof upgrade), and it inherits the compute receipt's trust model.

4.7 A full turn, court-handable and value-blind (PAIR)

- **One owner-signed record over three unmodified legs.** PAIR (`pair/pair.mjs`) staples the Grounding Receipt (what was used + closure), the Stapled Transcript (where it ran + nothing off-memory), and the Empty Witness (what was provably never supplied) into one record, adding only an outer gate: a cross-leg **disjointness** check (no fact is both grounded-this-turn and never-supplied, compared over field-name *addresses*, never values), a single shared epoch pin, and an owner signature over the whole record.
- **The falsifying suite.** `pair/pair.test.mjs` (28 checks) requires the honest record to verify and rejects every cheat: smuggled off-memory entry, a staple from a different turn, a forged compute signature, an Empty Witness that claims never-supplied for a fact that *is* in the ledger, a cross-leg overlap, an epoch splice, a freshness downgrade, and any plaintext-value or salt leak. The independent `pair.adversary2.test.mjs` (13 checks) adds the **cross-leg session splice** — a custodian that *did* release a fact this turn builds a genuine Empty Witness against a decoy session — and confirms it is rejected once the verifier pins the real session id + real release root out-of-band (§5.3).

4.8 Anti-poisoning proof-of-exclusion (Memory Immunity Receipt)

- **The new negative property.** The dual of the Grounding Receipt: it proves a named answer's grounding set was reprojected from *exactly* the enrolled-author-signed ops, and that every unsigned / forged / non-enrolled op is in a re-derived **quarantine** set that contributed nothing. The verifier re-partitions the raw candidate log itself (admitted vs rejected), re-projects from admitted-only, and checks the answer's grounding set is closed over that reprojection — so a forged op forced into the prompt cannot be in ADMITTED (no enrolled signature) and closure breaks.
- **The falsifying suites.** `immunity.test.mjs` (8), `immunity.attack.test.mjs` (19), `immunity.soundness.attack.test.mjs` (16), and `immunity.roster.redteam.test.mjs` (13) all pass with **soundness breaks = 0** — including attacker owner-key, head-receipt from a different log, roster forks, `__proto__` / `constructor` grounded keys, and last-writer-wins poison contention. Honest boundary: this proves **attributability**, not semantic truth (an enrolled author can sign a false fact); the verifier must pin the roster **content hash** out-of-band (§5.3), and completeness over *omitted* enrolled ops is an out-of-band pin, not a receipt-internal guarantee (§5.4).

4.9 Provable consolidation — distilled and destroyed (FM-051)

- **Two-sided set identity, with a real key-destruction check.** `consolidation/consolidation.mjs` proves a durable summary is the residue of *exactly* a named set of ephemeral sessions **and** each ephemeral is now information-theoretically unrecoverable: `lineage-IN == destroyed-OUT`. Crucially, destruction is **observed, not asserted** — for every cited ephemeral the verifier confirms the per-ephemeral key is gone from the authenticated custody *and* the ciphertext does not decrypt under any key the snapshot still carries. Each ephemeral uses its own **random** DEK (`crypto.randomBytes(32)`), so there is no re-derivation path; an earlier HKDF-over-retained-seed draft was a break and is recorded (§6).

- **The falsifying suites.** `consolidation.test.mjs` (15 checks) and `adversary.attack.test.mjs` (11 attacks, 0 broke) reject a tombstone-without-destruction, an op_id unbound from its lineage triple, and a declared-destroyed superset of the lineage. Honest boundary: it proves unrecoverability of the *exported* state, not of copies held on a replica you do not control, and not that no plaintext was exfiltrated before consolidation.

4.10 The remaining emergent capabilities

- **Discovery Shield (dual negative that survives a decryptor).** `discovery-shield/combined.test.mjs` (11/11) proves, for one `(X, S)` pair, *both* "no byte released" and "no consent granted." `decryptor-attack.test.mjs` (9) plays a party holding `K_rec` + plaintext and confirms decryption capability buys **neither** leg; `attack.binding.test.mjs` (9) confirms the two legs are bound to the same `(X, S)`. The `FINALE` confirms a genuinely never-released, never-consented `(X, S)` still verifies — the decryptor proved nothing.
- **Cross-Session Grounding Closure (directional non-flow).** `cross-session-closure/cross_session_closure.test.mjs` (14) proves no entry grounded as HIGH in session `S1` appears in a later LOW session `S2`. Soundness is "re-derive, don't trust": the verifier recomputes the HIGH and LOW commitment sets from the two owner-signed session receipts and intersects *those*. `attack.test.mjs` (11) + `fresh_attack.test.mjs` (20) resisted every attack. Honest boundary: disjointness is over **commitments**, not values — re-committing the same value with a fresh salt verifies, so the proof is "custody did not *re-supply* the committed entry."
- **Decentralized Oblivious Recall (access-pattern obliviousness, at an honest cost).** `oblivious-recall/oblivious_recall.test.mjs` (21) proves a byte-identical on-wire access trace across queries (a trained distinguisher scores ~0.0003–0.009, below the noise floor); the witness binds the per-node on-wire order and the pinned node count, and `oblivious_recall.attack.test.mjs` (8) + three red-team suites reject query-dependent reorders and witness forgeries. Honest cost, stated: full-batch blowup equals catalog size; the windowed variant is **not** oblivious across sessions and must not be shipped as oblivious.
- **Graded Consent Spend-Trace (a verifiable consent/usage budget).** A semiring of grades compiled into a P-256-signed, hash-chained, monotone budget ledger, so an over-tier / foreign-purpose / over-cap / sub-quorum release cannot mint a valid receipt; a portable auditor re-folds the chain (order-independence, monotonicity, tamper-evidence). Proven after two adversarial fix rounds (an integer-overflow guard and a grade-binding check, §6). Honest boundary: it governs access/usage *accounting*, not confidentiality-during-use.
- **Soul Will (consent-bound inheritance).** `soul-will/soul_will.test.mjs` (25) + an 11- and a 12-attack red-team hold scope-partition, heir-pin, dead-hand, BURN, and confidentiality (the Shamir key-commitment is the hard backstop). Honest boundary: BURN is information-theoretic unrecoverability if a true threshold drops shares, not physical destruction; the death-trigger needs a liveness oracle (roadmap).
- **Dead-Hand Recall (the hardware possession-root, software-sim).** `dead-hand-recall/dead_hand_recall.test.mjs` (14) + an 11-attack re-attack prove power-off seizure of the entire at-rest state without a live "tap" fails, a captured prior-session PRF transcript does not reopen a new recall (fresh salt → fresh tap), and a stale-epoch replay is rejected. Honest boundary: protects key at rest + requires a live tap, not confidentiality-during-compute; the off-device PRF is modeled (a real FIDO2 hmac-secret key per the $58 BOM).

4.11 The collective layer (L5 DNG), with its largest honest scope

`real-dp-collective/dng.test.mjs` (`CORE: 6 checks passed`): the aggregate carries calibrated discrete-Gaussian noise as a consequence of closure-under-summation, the clean sum never materializes, and a zero-noise free-rider is rejected by the public-coin sampling proof. The empirical (ε,δ) sweep (`dng.dp.test.mjs`) and the collusion / adaptive-composition attacks (`dng.kcollusion.attack.test.mjs` , `dng.adaptive.attack.test.mjs`) carry the measured bound (a monotone σ knob; membership-adversary advantage well below the analytic bound) and the four-part trust boundary recorded in the dossier. The honest scope on L5 is the largest in the stack and is stated wherever DNG is claimed: epsilon is a **computational** bound (not information-theoretic, forced by Biswas-Cormode / Haitner-Omri), the **release still needs an honest or threshold aggregator** (build-attack A7 "a malicious aggregator cannot strip the noise" was falsified and retracted — the cut-and-choose publishes the noise count in cleartext at `dng.mjs:453`), the masking degrades to local-DP under full `M-1` collusion, and at least one honest noise contributor is required. The *noise generation* needs no trusted curator; the *release* does.

5. Honest negatives — recorded refutations and stated walls

These are not failures of the program; they *are* the program. The bias was to falsify, and the falsifications were kept.

5.1 Single-key forget-proof — refuted by tail-truncation, then by stale-seal replay

The first forget-proof (FM-002, `forget-and-prove/`) tried to issue an offline-verifiable "this entry is forgotten" receipt from a single-key crypto-shred plus Merkle exclusion. It was **refuted**, and the refutation is preserved:

- **Attack A/B (`attack.test.mjs`).** The verifier validates the chain forward from genesis with no signed commitment to the ledger's length/head, so *any contiguous prefix* of a valid ledger is itself fully valid. The owner stores, forgets, then *resurrects* an entry (a PUT after the FORGET). A malicious host drops the trailing resurrection op and shows the auditor `…PUT, FORGET`; `verifyLedger` passes and `proveForgotten` returns `forgotten: true` for data the owner still holds — a **forged positive forget proof**. Attack B reproduces it in a multi-entry ledger.
- **The deeper, fundamental limit.** A v2 with a signed head/length was *still* broken (`attack_round2.test.mjs`, R3): a genuine, validly-signed *past* seal can be replayed against a truncation of the current ledger whose prefix matches. The root cause is information-theoretic — a **stateless offline verifier handed only a snapshot cannot distinguish "latest" from "a valid past state."** Soundness requires a freshness anchor: a pinned monotonic epoch, a fresh quorum-co-signed nonce, or an external witness.
- **What survived.** The robust forget primitive is the *threshold* one: TRM's T4 requires two independent threshold attestations — the consent quorum **and** a threshold of recall-key-share holders attesting destruction (a consent quorum alone cannot assert a shred), a `< k_r` surviving-share count, and a freshness anchor (the stale-but-genuine seal is rejected on epoch + nonce). The honest boundary stays explicit: T4 proves *attestation* of destruction plus information-theoretic unrecoverability once a true threshold drops shares — **not** physical destruction on a malicious replica that secretly kept a copy. The integration breaks found during the TRM capstone (T1's unsalted cleartext commitment, T3's free-standing op labels, T4's wrong-quorum binding) were broken first, fixed at the TRM layer with the proven legs unmodified, and re-attacked to green (`trm.test.mjs`, 22 assertions).

5.2 USPIR parked, and the FM-047 wall

Two recall results are recorded as **negatives** so the proven set is not over-read.

- **USPIR sublinear DPF recall (FM-046) — real but parked.** A two-party BGI distributed-point-function read over two user-owned ciphertext replicas gives genuine sublinear recall (O(log N) up, O(1) down) with **no FHE/TEE/trust**, and a single-key view does not localize the queried index (`uspir-dpf/uspir.test.mjs`, 14/14; distinguisher advantage ≈ 0.009). It is parked, not claimed, for three honest reasons: the crossover is N ≥ 8 (not N ≥ 4); it is **static single-writer only** (a divergent replica fails GCM auth, so it does not cross the multi-writer wall); and the DPF path carries **no recomputable obliviousness witness** — the suite's own check (F) records this downgrade.
- **FM-047 witnessed-sublinear-witness — a wall that held.** The attempt to give the sublinear read the witness USPIR lacks (a node reveals its single DPF key; an auditor re-expands over all N to confirm a full query-independent scan) was **broken by the independent red-team**. The builder's value-level suite passes (`witness.test.mjs`, 14/14), but `witness.attack.test.mjs`, `witness.bind.test.mjs`, and `witness.keymal.test.mjs` each *report a break as their asserted, expected outcome* (they exit `0` because finding the break is the test's success condition). The witness is **self-issued and unbound to the live query**: because the reply is recomputable from public blocks plus a self-chosen key, a node can fabricate a consistent `(key, reply)` pair entirely offline — a node that served zero queries passes, and a key for an index never queried passes. **Plain recomputation cannot bind a self-issued offline witness to runtime behavior.** What genuinely held (value-hiding, single-key index-hiding, value-level recomputability, a sound Merkle commitment, dense scan-binding) is recorded; the required fix turns it into a logged interactive protocol or needs a SNARK/TEE. This is the program's cleanest statement of the rule that a self-consistency check mistaken for an execution attestation is glue, not a primitive.

5.3 The four breaks this build cycle found and closed

Every one fits the composition-glue pattern and is now closed fail-closed (full detail in `03_threat_model_and_honest_limits.md` §7):

- **The witness break (FM-047)** — the self-issued, query-unbound witness above. Recorded as a negative; not shipped.
- **The PAIR session splice** — a custodian that *did* release `X` this turn could try to satisfy the non-supply leg with a genuine Empty Witness built against a **decoy session**. Closed by pinning `session_id` + the release root out-of-band and requiring every witness to be session-bound; `pair.adversary2.test.mjs` confirms the splice is rejected and the honest control still verifies.
- **The immunity roster fork** — the verifier originally pinned only the epoch *integer*, so the owner could sign a *second* roster at the same epoch enrolling a poisoner. Closed by requiring the roster **content hash** (plus head + epoch), all three, fail-closed; `immunity.roster.redteam.test.mjs` is green at 13/13.
- **Tombstone-without-destruction and seed-rederivation (consolidation)** — the first receipt proved destruction by checking a signed REMOVE tombstone (forgeable while keys stay live), and an HKDF-over-retained-seed draft left a re-derivation path. Both closed in FM-051 by random per-ephemeral DEKs, a real key-destruction check, and an owner-signed custody seal.

5.4 A note on stricter-than-the-probe: the immunity honesty suite

`immunity.honesty.attack.test.mjs` reports `8 passed, 5 failed` with its own tally `BREAKS = 0, OVERCLAIMS = 0`, and is the one suite in the matrix that exits non-zero. This is **not** a soundness regression and is recorded plainly: five of its assertions were written to observe a rejection arriving via a *specific* code path (e.g. a signature-mismatch or a quarantine-count comparison), but the roster-hash hardening now fails closed *earlier* — "no pinned roster hash (fail-closed)" — before the condition those probes expected to read. The probes are stale against a stricter verifier. The authoritative roster red-team (`immunity.roster.redteam.test.mjs`, 13/13) and the soundness suite (`immunity.soundness.attack.test.mjs`, 16/16) are green with zero soundness breaks, and the remaining honesty-probe failures are explicit out-of-scope completeness checks (omission of an enrolled op is an out-of-band pin, not a receipt-internal guarantee — the verdict makes no completeness claim). The honest reading: the verifier is stricter than the probe, not weaker.

5.5 Three ideas falsified to features — not claimed

The program deliberately hunted past composition-grade for a new primitive across ten experimental frontiers and came up empty — the frontier is literature-occupied. Three of the louder candidates were falsified to features by their own adversarial suites and are **not** claimed anywhere in the white paper:

- **Entangled Memory (FM-025).** Headline: corpus ⇄ adapter as cryptographic co-keys with mutual uselessness and live-presence gating. **Falsified** (`entangled-memory/adapter-standalone.test.mjs`, `attack_presence.test.mjs`, `saliency_decisive.test.mjs`): "mutual uselessness" is one-directional — a stolen adapter is a fully functional model; the "live-presence witness" is cryptographically indistinguishable from shares-at-rest (it collapses to TRM's existing property and limit); the "saliency challenge" is `HMAC(adapter_bytes, public_challenge)` — deterministic and replayable, not a third secret. Net: collapses to "TRM plus an optional forward-secret adapter-hash binding," a minor feature.
- **Semantic Cones (FM-026).** Headline: decrypt only memories whose embedding falls in a granted region ("work" but provably not "health"). **Crypto sound but headline downgraded** (`semantic-cone/pertag_equiv.test.mjs`, `metadata.test.mjs`): the per-cone key is `HKDF(master, label)` — identical whether `label` is a human tag or a cluster id, so "semantic" adds nothing cryptographic; and the real boundary is *non-crypto* embedding correctness — every cone pair has a structural leak band, and a real "medical leave for chemotherapy" memory (margin 0.005) mis-binned to WORK was decrypted by a WORK grant. Honest claim: a grant decrypts exactly the memories *assigned* to the granted cones; assignment is a heuristic and boundary memories can be miscategorized.
- **Holographic privacy / keyed holographic memory (FM-030).** Forget-by-algebra (subtracting a bound term) is genuine and surgical, but the privacy headline is **falsified** (`keyed-holographic/leakage.test.mjs`): the energy statistic `‖corr(H, v)‖` is a key-free membership oracle (AUC 0.84 at N=16) and `‖H‖` leaks the exact item count. Superposition is not encryption; the idea is research-bounded.

6. The composition-glue meta-lesson, with the build record as evidence

Across the program, the breaks sort by *where they lived* into a single finding:

> **Every break across the program lived in composition glue or a fail-open default — never in a red-teamed cryptographic primitive.**

The genuinely novel core (TRM T2) never broke — 17 forgeries on first contact. Every other historical break was integration or a footgun, and every one was re-hardened to fail closed: TRM's T1 (an unsalted cleartext `SHA256(M)` commitment in the at-rest envelope), T3 (unauthenticated op labels trusted as free-standing), and T4 (forget bound to the consent quorum instead of the recall-key holders); FM-018 non-supply (a queried hash never bound to the absence key — a one-line fix); FM-002 (a missing authenticated head, then the fundamental freshness limit); the DNG A7 overclaim (a cleartext noise-count field in the release envelope); the FM-016/019/023 fail-open defaults (a bare verifier defaulting to accept, content-binding enforced only when a hash was supplied, malformed inputs that threw); the Graded Consent two rounds (an integer-overflow magnitude bypass, then a grade-binding bypass where the spender swapped the grade the devices never signed); the Forget Certificate's two soundness gaps (a recompute-absence leg that trusted a flag, then a last-writer-wins residue); and this cycle's four breaks (§5.3). In each case the proven legs were left byte-unmodified and the fix lived in the outer gate.

Two operating principles follow, and they are the product discipline:

1. **Proven primitives are not a proven system.** The Shamir split, threshold-Schnorr, RFC-6962 ledger, SMT non-membership, BGI DPF, and discrete-Gaussian sampling were all sound off the shelf. The integration layer that wires them together needs *its own* adversarial test suite — because that is precisely where the breaks were.
2. **No security check was ever weakened to pass a test.** Across every hardening pass, verifiers were only ever made stricter / fail-closed. The recorded honest negatives — the tail-truncation break, the R3 stale-seal replay, the A7 noise-count retraction, the FM-047 self-issued-witness wall, the three Entangled/Semantic/holographic downgrades, and the stale immunity honesty probe (§5.4) — are kept as refutations, not quietly shipped.

---

7. Closest prior art (every novelty named)

Novelty here is composition at the application altitude — user-custody, per-turn, offline-verifiable — not a new cryptographic primitive. The closest published work for each axis:

- **Private retrieval / encrypted AI memory:** Opal (arXiv 2604.02522, TEE + ORAM user-memory confidentiality); Anuma (encrypted cross-model vault, on-chain *logging* in place of proof). Wedge: a memory node that *never decrypts* plus an offline-verifiable receipt, vs an enclave that decrypts or a log that records permissions.
- **Signed-op admission / provenance:** Provenance Gates (arXiv 2605.13471) and MemLineage (arXiv 2605.14421) for signed-op admission and per-principal lineage; Portable Agent Memory (arXiv 2605.11032) for content-addressed + Merkle + capability memory. Wedge: sovereign custody + quorum write-admission + an offline value-blind proof-of-exclusion (Memory Immunity), not label-and-gate.
- **Proof-of-non-membership / forgetting:** ZK-APEX (2512.09953), ZKPoU, Verifiable Federated Unlearning (2510.00833) — all **model-side** weight unlearning. Wedge: **ledger-side** absence over user-owned content-addressed memory plus an offline static verifier. Novelty here is retired to this wedge; we do **not** claim "nobody proves absence."
- **Compute receipts / proof-of-inference:** c0mpute / Shard, Lagrange DeepProve, VeriLLM, Gensyn Verde, Hyperbolic PoSP — fast and verifiable for *correctness*, but every proof requires the prover to hold the input in **plaintext** (the receipt hides data from the verifier, never the prover). FlowMemory supplies the structurally-missing **input-provenance** operand (Stapled Transcript / PAIR) that the compute sector cannot produce on its own.
- **Grounding / commit-and-audit:** CommitLLM (model-side commit); zkRAG / VeriRAG / V3DB (retrieval over a *service-owned* DB). Wedge: closure bound to a **user-owned** memory subset.

- **Crypto-shred / certified deletion:** Vanish (USENIX Security 2009) and "Deleting Secret Data with Public Verifiability" (eprint 2014/364, the standard that a deletion must be publicly verifiable via key destruction, not asserted by a tombstone) — which the Provable Consolidation key-destruction check explicitly honors.
- **Server-side AI memory (the incumbents):** Mem0, Letta, Zep — app-owned memory the operator can read and be compelled to produce; the contrast that defines the user-custody wedge.

---

8. How to reproduce

From `research/prototypes/`, run any suite with `node`:

```
cd trm && node trm.test.mjs                          # L1 TRM T1–T4 (22 assertions)
cd pair && node pair.test.mjs                         # full-turn record (28 checks)
cd memory-immunity && node immunity.test.mjs          # anti-poisoning exclusion (8 checks)
cd consolidation && node consolidation.test.mjs       # provable sleep (15 checks)
cd discovery-shield && node combined.test.mjs         # dual negative proof (11 checks)
cd reprojection-forget && node attack.test.mjs        # cross-process determinism audit
```

Exit `0` is the property holding; any non-zero exit is a loud, named failure (the one deliberate exception is `memory-immunity/immunity.honesty.attack.test.mjs`, whose stale probes are explained in §5.4). The full matrix in §3 was re-run green this session on Node v24.15.0. No suite depends on npm or the network, and no verifier in this docset was ever weakened to make a red suite green.

Use cases and killer products

> This section explains what the proven stack lets a person, a defendant, an enterprise, or a regulator *do* — the concrete capabilities the composition unlocks, none of which a competitor ships today. Every capability below is backed by a runnable prototype and a passing adversarial test, or it is labelled a limit or a target. Test counts and exit codes were re-run live while writing this section. The honest scope of each capability is stated inline, not buried, because the scope is the source of the credibility.
> **Honesty contract (binding throughout).** Signatures and receipts prove custody, integrity, ordering, consent, minimisation, grounding, non-supply, exclusion, and forgetting. They do **not** prove confidentiality during compute, they are **not** FHE, they say nothing about what a model internally attended to, and they make no claim about semantic truth. On-device means *locality*, not FHE. We never claim a server or model cannot read data once it has been released. There is no token.

---

1. Why the use cases, not the primitives

The build program established — across an aggressive multi-iteration hunt — that the frontier of *new cryptographic primitives* for private memory is already literature-occupied (puncturable/honey/time-lock encryption, DPF-PIR, CHURP, fuzzy extractors, leakage odometers). The defensible novelty therefore lives at the **capability and composition level**: emergent properties of the whole stack that are present in no single layer and shipped by no incumbent. The strongest of these is a constructive proof of a *negative* about AI custody — the Empty Witness — and a family of full-turn, court-handable governance records built on top of it.

Two recurring honesty rules shaped every result here:

1. **Every break in the program lived in composition glue or a fail-open default, never in a red-teamed core.** The adversarial build, not the literature search, is the real filter. Proven primitives are not a proven system; the integration layer earns its own attack suite.
2. **Receipts prove custody, not cognition.** The boundary between "what the custody layer supplied / withheld / admitted / forgot" (provable) and "what the model did with it" (not provable offline) is the honest line under every product below.

---

2. The Empty Witness — a court-handable proof your AI was never told X

2.1 The problem nobody has a primitive for

When an AI system is *accused* of having seen, stored, or leaked something it never did — a leaked trade secret, a piece of regulated PII, a defamatory "fact" the model supposedly emitted — the accused has essentially nothing to offer. The legal baseline is the weak maxim "*absence of evidence is not evidence of absence.*" You cannot point at a log of everything that did not happen.

The closest prior art proves the **opposite** direction:

- **SCITT refusal-events and VeritasChain CAP-SRP** log what an AI *refused* to do — a positive record of refusals that explicitly cannot speak to what was never logged at all.
- **ZKPROV** (arXiv 2506.20915) proves *positive* training provenance — what a model *was* trained on.
- **Provenance Gates** (arXiv 2605.13471) and **MemLineage** (arXiv 2605.14421) attest signed-op *admission and lineage* — what entered memory — not the absence of an entry.

No shipped or published system hands a defendant a constructive cryptographic **proof of a negative**: "*this exact fact X was never released into this AI session S.*" That is the position the Empty Witness fills. The honest novelty is at the **capability/product** level: it is the packaging of one already-proven layer (proof-of-non-supply, FM-018) into a courtroom artifact — not a new fusion

and not a new primitive.

2.2 What it proves, and what it does not

The Empty Witness is a thin product surface over the proven proof-of-non-supply layer (`research/prototypes/proof-of-non-supply/nonsupply.mjs`). The honesty contract is written into the prototype's own header and is binding here.

It proves (offline, from only a public key and one witness file):

- **A signed release set.** A session has a *release set* — the content-hashes the custody layer actually supplied to a model during that session — committed to a sparse-Merkle-tree (SMT) root, P-256 signed (`nonsupply.mjs` , `openSession`).
- **Non-supply of a queried fact X.** An SMT *absence* witness proves the address of X resolves to the empty leaf under that signed root: X is not a member of the release set. The witness is bound to the exact query — the verifier recomputes `addr(contentHashX)` and requires `absence.key` $\equiv$ `addr` , so a custody layer cannot answer "is X absent?" with a valid absence proof for an unrelated, genuinely-absent Y (`verifyNonSupply` , check V4).
- **Authenticated session scope.** The `sessionId` is signed inside the release-set-root payload and independently carried in the ledger op; both must match. An answer scoped to session S2 cannot be relabelled as an S1 answer (check V5).
- **No retroactive revision.** The signed release-set root is an entry in an append-only RFC-6962 Merkle ledger. An RFC-6962 *consistency* proof from a checkpoint the verifier **pinned** shows the log only grew. Inserting X after the fact and re-rooting produces a new, later, visible entry — it cannot silently overwrite the pinned absence claim (checks V1–V3).

It does not prove (residual limits, never overclaimed):

- **Not what the model internally conditioned on.** This is a custody-layer *release* fact. If a byte left custody it counts as released; the proof says nothing about attention, activations, or what the model could infer.
- **Not confidentiality during compute.** Committed values are hashes; on-device privacy here is locality, not FHE.
- **Freshness is required.** A stateless verifier given only an old snapshot can be fooled by replaying a valid past state. Soundness requires the verifier to pin the latest checkpoint it has authenticated — which is exactly the mechanism that defeats retroactive insertion.

2.3 The data structure

A released item is addressed by `releaseAddr(contentHash) = hashKey(contentHash)` and stored as a real, non-empty leaf. An item that was never released resolves to the precomputed empty-subtree leaf, so an absence proof is the Merkle path showing the leaf at `addr(X)` is that empty hash; forging a non-empty key to look empty would require a SHA-256 second preimage.

```
session release set ──insert(addr(ch), H("supplied:"+ch))──▶ SMT
                                          │
                                    SMT.root() ── P-256 sign{root, sessionId,
                                          │                    kind:"release-set-root"}
                                          ▼
                        append-only RFC-6962 ledger entry:
                        { sessionId, release_set_root, signed_root }
                                          │
                                    checkpoint() ── signed, hash-chained head
```

The witness handed to a relying party is `{ sessionId, contentHashX, absence, signed_root, ledger_op, inclusion, consistency }` (`nonsupply.mjs` , `proveNonSupply`).

2.4 The live demo script

This is the demonstration to run in front of a judge, a regulator, or an acquirer. It surfaces an already-passing prototype behind a UI; no step is staged.

Setup (custody side, done once). The custody layer has been running a release ledger. During session `S` it released some facts to the model and committed the session's release-set root into the append-only ledger, then issued a signed checkpoint. The relying party (court / regulator) has previously authenticated and **pinned** that checkpoint.

1. **Type the claimed fact.** The plaintiff types the exact fact they allege the AI was told. The verifier computes its content-hash and then `addr = hashKey(contentHash)`.
2. **Recompute the address, find the empty leaf.** The verifier walks the absence proof to the leaf at `addr` under the signed release-set root and confirms it is the empty leaf — X is not in the release set (`verifyProof(absence, rootHex, { expectAbsent: true })`, V6).
3. **Bind the answer to the question.** The verifier independently recomputes `addr(contentHashX)` and requires `absence.key === addr` (V4) — no answering about X with a proof for an unrelated absent Y.
4. **Confirm the session.** The authenticated `sessionId` inside the P-256 signed root must equal the queried session (V5).
5. **Confirm the log only grew.** An RFC-6962 consistency proof from the pinned checkpoint to the current head verifies append-only growth, and the release-set root's ledger entry is shown included in the authenticated head (V1–V3).
6. **Print the verdict.** The verifier returns `VERIFIED NON-SUPPLY` — *X was not released into this session, and the committed log has only grown since you last looked.*

Then break it live. The demonstrator now tries to forge it, and every forgery is rejected on screen. These are real assertions in `nonsupply.test.mjs` (13/13 PASS) and the re-attack suites:

Forgery attempt	Result
Claim non-supply of an X that *was* released	rejected (no absence leaf for a present X)
Retroactively insert X and re-root after the pin	rejected (consistency proof)
Equivocate on the pinned checkpoint position	rejected
Forge an absence leaf for a present key	rejected (second-preimage)
Sign the release-set root with the wrong SMT key	rejected
Sign the head with the wrong ledger key	rejected
Tamper the ledger op payload	rejected (inclusion re-derives op_id)
Roll back to a head older than the pin	rejected
Key-substitute: answer about X with a proof for Y	rejected (V4)
Cross-session relabel S2 → S1	rejected (V5)

The honest line for the courtroom: this is a proof of **custody-side non-release**, query-bound and per-session — *not* a claim about what a remote model internally computed, and the verifier must be pinned to the latest checkpoint.

---

3. Discovery Shield — non-release and no-consent, surviving a decryptor

The Empty Witness proves no byte of X left custody into S. It says nothing about *authorisation*. **Discovery Shield** is the strongest emergent fusion in the program: one offline verifier that proves, for the same `(X, S)`, **both** legs bound to a single join key (`research/prototypes/discovery-shield/discovery-shield.mjs`; `combined.test.mjs`, 11/11 PASS, plus binding, freshness, and decryptor attack suites green).

- **Non-release (Leg 1):** the proof-of-non-supply above.
- **No consent (Leg 2):** there exists no valid `k_a`-of-`n` threshold-Schnorr consent receipt that ever authorised releasing X into S, checked against a pinned consent group key over a canonical, domain-separated, length-prefixed message `consentMsg(X, S)`.

3.1 The emergent property: it survives a decryptor

The defining result — proven, not asserted — is that a party who reconstructed the recall key `K_rec` and decrypted X (and therefore holds the plaintext) still cannot forge either leg (`combined.test.mjs`, `T-DECRYPTOR-SURVIVAL`):

- It cannot forge **Leg 1**: inserting X into S's release set changes the signed SMT root, surfacing as a new, later ledger entry that the consistency proof from the pinned checkpoint catches; no absence witness can exist for a present X.
- It cannot forge **Leg 2**: a verifying consent receipt needs a genuine `k_a`-of-`n` threshold-Schnorr aggregate whose group key is never reconstructed. Knowing `K_rec` or the plaintext is not a consent share. A bundle below `k_a`, or built from attacker keys, fails `verifyConsentSignature` — the same TRM property whose consent core held 17 forgeries.

Neither layer alone yields the claim. The combined verifier enforces a single join key: the queried `(X, S)` must match the witness's authenticated `contentHashX` and `sessionId`, and the consent leg is checked over `consentMsg(X, S)` derived from those same fields (`verifyDiscoveryShield`, leg-binding check 3a; `T-LEG-BINDING`).

Honest scope. Custody-side only (not model cognition); the verifier must pin the latest checkpoint (`T-FRESHNESS` proves a stale-only verifier is foolable and a current pin catches a later release); the no-consent leg is open-world — it proves no authorising receipt *among those shown, under the pinned group* — and fails safe.

---

4. PAIR — the Provable AI Interaction Record (full-turn, court-handable)

The Empty Witness proves what was *never* supplied. The Grounding Receipt (§5) proves *exactly* what was supplied. PAIR (FM-048, `research/prototypes/pair/pair.mjs`) staples the whole turn into one owner-signed, value-blind record a court or regulator can verify offline. It composes three proven legs **byte-for-byte, never modified**, and adds only a strictly-stricter outer gate:

- **What memory was used** — Grounding Receipt (FM-039): membership + closure.
- **Where it ran, and nothing off-memory** — Stapled Transcript (FM-045): the compute receipt cross-bound to the grounding closure set by an ordered Merkle root.
- **What was provably never supplied** — Empty Witness (FM-018): one non-membership witness per never-supplied field.
- **Cross-leg consistency** — the used-address set and the never-supplied-address set must be **disjoint**; all legs must share **one epoch pin** and a **pinned ledger checkpoint**; the owner signs the whole record.

A fact's identity across legs is its field-name *address* in the shared 256-bit SMT space (`addr = hashKey(field_name)`). The disjointness gate compares only those addresses — never a value, never a salt. Field-names are namespace labels, exactly as the underlying legs already expose them.

`verifyPAIR` fails closed unless every leg verifies in full *and* the new outer gates hold: grounding verifies against the verifier-held signed root at the pinned epoch; the staple's embedded grounding receipt is byte-identical to the record's; each Empty Witness is session-bound to the verifier-pinned release root (closing the cross-leg "decoy-session splice"); the used and never-supplied address sets are disjoint; there is a single shared epoch; and the owner signature binds the record.

Test evidence. `pair.test.mjs` — 28 checks PASS; `pair.adversary2.test.mjs` — 13 attacks resisted, honest control verifies.

Honest limits (inherited, not strengthened). PAIR inherits every leg's limit and the compute receipt's left-half trust model (same-engine vs third-party); it does not strengthen them. The staple binds committed *segments and order*, not the interstitial glue bytes between segments (a named ZK opening upgrade, out of scope). It requires two anchors — an epoch pin and a ledger-checkpoint pin — and fails closed without either. The verifier must obtain `pinnedSessionId` and `pinnedReleaseRoot` for the interaction from an independent anchor (the release-ledger checkpoint), never from the record under audit. As with every layer: this proves what the custody layer supplied / withheld this turn, not what the model internally attended to.

Closest prior art: c0mpute-class proof-of-inference receipts (c0mpute/Shard, Lagrange DeepProve, Gensyn Verde, Hyperbolic PoSP) prove *compute correctness* but every such receipt requires the prover to hold the input in plaintext — the receipt hides data from the *verifier*, never the *prover* — and none binds the input to a *user-owned* memory subset with a closure property. Anuma logs encrypted cross-model permissions on-chain; it does not prove grounding. PAIR is the input-provenance half those systems structurally cannot produce.

---

5. Grounding Receipt — proof that an answer came from exactly your memory

The Grounding Receipt (FM-039, `research/prototypes/grounding-receipt/`) is the dual of the Empty Witness: a per-answer, offline-verifiable, value-hiding receipt binding a model answer to **exactly** a committed subset of the user's own memory, with a **closure** property proving nothing off-memory was smuggled into the prompt.

It proves: (1) subset-membership of each used entry against the signed SMT release root, and (2) closure — `input_hash` equals the hash of the canonical union of the opened entries, so no off-memory content entered. The load-bearing invariant is that the verifier recomputes `input_hash` over *exactly* the membership-proven entries, so the hashed set equals the proven set. The receipt reveals field-names and salted commitments only — never values.

Test evidence. `grounding_receipt.test.mjs` — 17/17 PASS; `grounding_receipt.attack.test.mjs` — 20 attacks resisted (closure-desync, off-memory smuggle, SMT malleability, freshness/epoch, value-revelation, regression).

Honest limit. Proves the custody layer *supplied* exactly these entries — not what the model internally did with them. Binding the *actual prompt bytes* a GPU consumes needs the Stapled Transcript (FM-045), and binding the interstitial glue bytes between segments needs a ZK opening proof (out of scope). Closest prior art: zkRAG / VeriRAG / V3DB prove retrieval over a *service-owned* DB; selective-disclosure credentials (BBS+/SD-JWT) are for credentials; none binds *generation* to a *user-owned* memory subset with closure.

---

6. Memory Immunity — anti-poisoning proof-of-exclusion for agents

Memory poisoning (OWASP ASI06, MINJA, MemoryGraft) is the threat that an op which *looks like* memory was not authored by an enrolled principal and is silently admitted before projection. The Memory Immunity Receipt (`research/prototypes/memory-immunity/immunity.mjs`) is the dual of the Grounding Receipt on the *authorship* axis: it proves a named answer's symbolic grounding set was reprojected from **exactly** the enrolled-author-signed memory ops, and that **zero** unsigned, forged, or non-enrolled ops contributed.

Per committed op it adds: (1) an **authorship predicate** — a valid P-256 signature by a roster-enrolled author over the canonical op body (not agent self-signing); and (2) a **negative exclusion proof** — the candidate op-log is partitioned into ADMITTED (enrolled-signed) and a QUARANTINE root of REJECTED ops, the symbolic projection is a deterministic pure function of ADMITTED only, and the verifier **re-partitions from the raw candidate log itself** and re-projects. A forged op forced into the prompt cannot be in ADMITTED, so closure breaks.

Test evidence. `immunity.test.mjs` — 8/8 PASS (value-hiding); `immunity.soundness.attack.test.mjs` — 16/16, soundness breaks = 0; `immunity.attack.test.mjs` — 19/19, breaks = 0; `immunity.roster.redteam.test.mjs` — 13/13, breaks = 0. The verifier requires pinning the **roster content hash** (not just the epoch integer), the candidate head, and the epoch — all three required, fail-closed — because an owner can sign a second roster at the same epoch enrolling a poisoner.

Honest limits (the boundary is the credibility). This proves what the custody layer *passed* to the model from authenticated ops, not what a remote LLM attended to, ignored, or hallucinated; **prompt-injection in the live query is out of scope** — this covers poison arriving as a stored memory *op*. It proves *attributability*, not truthfulness: an enrolled author can sign a false fact; the point is that poison becomes attributable and revocable instead of anonymous. A separate honesty-boundary probe (`immunity.honesty.attack.test.mjs`) records, with zero breaks and zero overclaims, that the receipt is **scoped to the offered log** and makes **no completeness claim** — it cannot self-certify that an enrolled op was *not omitted*; completeness is an out-of-band pin, not a receipt-internal guarantee. Side-channel poison past the receipted custody boundary is out of scope. Closest prior art: Provenance Gates (2605.13471) and MemLineage (2605.14421) attest signed-op admission and lineage but gate-and-label rather than emit a value-hiding, answer-bound *proof of exclusion*.

---

7. Forget and consolidate

7.1 Regulator-grade forget certificate

The forget certificate (FM-031, `research/prototypes/forget-certificate/`) composes three proven legs, unmodified, into one portable offline certificate a regulator can verify: forget-by-reprojection (FM-029), a non-membership SMT absence proof, and the append-only memory ledger as the freshness anchor.

A prior soundness break (recorded honestly in the program log: an early verifier accepted a "gone-from-recompute" leg that trusted a self-reported flag, so a cert could be minted for a fact still live) has been **fixed and re-attacked**. The verifier now actually recomputes the projection from the surviving op-set and checks the fact is absent — destruction is observed, not trusted.

Test evidence. `forget_certificate.test.mjs` — 15/15 PASS (including: a certificate naming an op still present in the surviving set is rejected); `attack_fresh.test.mjs` — 4/4 fresh attacks rejected.

It proves, all relative to a verifier that pins the latest checkpoint: presence-in-history (the fact was once asserted), a recorded forget event, gone-from-recompute (the projection derives from a surviving op-set excluding the forgotten op), absent-from-root (non-membership against the current signed root), and freshness (one shared epoch, compared against the pinned one).

Honest boundaries. "Delete the op" means a signed REMOVE tombstone plus key crypto-shred — **not** physical byte-erasure on replicas you do not control; re-projection must be from scratch (a stale cache still leaks the forgotten fact); and for a trained neural adapter, cross-hardware bit-for-bit regeneration is impossible (root cause: non-associative float reduction order across hardware), so the adapter is handled by content-addressed *provenance* — "derived from exactly this op-set, which excludes X" — not regeneration. Closest prior art: ZK-APEX (2512.09953), ZKPoU, and Verifiable Federated Unlearning (2510.00833) certify *model-side* weight unlearning; this is *ledger-side* absence over user-owned content-addressed memory, with an offline static verifier.

7.2 Provable consolidation ("provable sleep")

Provable Consolidation (FM-051, `research/prototypes/consolidation/`) makes "sleep-time" memory distillation auditable. One owner-signed receipt proves a two-sided set identity — `lineage-IN == destroyed-OUT` — i.e. *"this durable fact distills exactly these now-unrecoverable sessions and nothing else, and nothing off that set survives decryptable."*

It is strictly stricter than a tombstone receipt. A naive receipt proves destruction by checking only that each source op carries a signed REMOVE tombstone — forgeable, since a custodian can append REMOVE ops while leaving the AES keys live and the ciphertext fully decryptable. This verifier performs a **real key-destruction check**: each ephemeral is encrypted under its own random per-ephemeral DEK (no re-derivation path from any retained seed); the verifier confirms (a) that key is gone from the owner-signed custody, and (b) the ciphertext genuinely does not decrypt under any key the authenticated snapshot still carries. Destruction is observed, not asserted. The receipt also binds each cited `op_id` to its own lineage triple, so a summary cannot be lineaged to one session's commitment while the destroyed `op_id` points at another.

Test evidence. `consolidation.test.mjs` — 15/15 PASS; `adversary.attack.test.mjs` — 11 attacks, 0 broke. Closest prior art: "Deleting Secret Data with Public Verifiability" (eprint 2014/364) establishes that deletion must be publicly verifiable via *key destruction*, not asserted by a tombstone — this lifts that principle into a value-hiding, lineage-bound consolidation receipt over user-owned memory.

Honest limits. Same as the forget certificate: key destruction in the owner-signed custody and information-theoretic undecryptability follow, but physical destruction on replicas you do not control is out of scope, and this is custody-side, not confidentiality-during-compute.

---

8. Soul Will — consent-bound memory inheritance

Your memory should outlive you only on your pre-signed terms. Soul Will (FM-035, `research/prototypes/soul-will/`) lets memory pass to heirs under a signed will:

- **Scope-partitioned** — bequeath the FAMILY scope, never the MEDICAL scope.
- **Quorum-of-heirs** — reading requires a threshold of the named heirs, mirroring the `k`-of-`n` consent gate.
- **Freshness-triggered** — a dead-hand release needs a liveness/freshness oracle to fire; a pre-trigger unseal is refused.
- **Provable BURN scope** — a scope can be dropped below its recovery quorum, making it information-theoretically unrecoverable.

Test evidence. `soul_will.test.mjs` — 25/25 PASS (positive + falsifying); `soul_will.attack.test.mjs` — a fresh red-team held, including cross-scope graft, duplicate-`x` shareId collision, pre-trigger unseal, post-trigger resurrection, and confidentiality (no attack produced a false plaintext; the Shamir keyCommitment is the hard backstop). One narrow availability/API-contract residual was found and fixed (canonical will hash now binds sorted share-id digests; `unsealScope` returns a structured `{ ok:false, gate:"integrity" }` instead of throwing). Confidentiality always held.

Honest boundary. BURN is information-theoretic unrecoverability *once a true threshold drops shares*, not physical destruction; the death trigger needs a real liveness oracle; and the inherited neural adapter is content-addressed, not claimed to be bit-for-bit regenerable across hardware. Closest prior art: Deadhand and Cipherwill reconstruct or custodially hold the master secret; SLIP-39 and CT-DAP (2603.07933) lack threshold + scope + non-supply + non-materialisation together. Soul Will's delta is posthumous scope-partitioned access with provable withholding, off seed-reconstruction.

---

9. Enterprise audit and individual sovereignty

9.1 Enterprise audit

Every memory write, edit, admission, release, and deletion is a signed leaf in the append-only Merkle ledger. For a regulated enterprise this yields an offline-verifiable answer to the four questions auditors actually ask:

- **What was supplied** to each model session — release-set inclusion / Grounding Receipt.
- **What was withheld / never supplied** — the Empty Witness, applied internally.
- **What was admitted to durable memory** — quorum write-admission requires `k`-of-`n` of the org's own devices (`quorum-write-admission/`, proven-hardened); no single rogue endpoint can make a memory durable.
- **What was forgotten / consolidated** — quorum crypto-shred plus a non-membership certificate (`quorum-crypto-shred/`, proven-hardened) and the consolidation receipt.

The structural advantage over an incumbent memory DB (Mem0, Letta, Zep): the record is user/org-keyed and operator-less — there is no third party holding an unauditable copy who can be compelled to produce or silently alter it. Honest limit: signatures prove custody, integrity, ordering, consent, and minimisation — not confidentiality during compute.

9.2 Individual sovereignty

For an individual the layers compound into one sentence: *your AI's memory is yours — un-seizable, recomputed-never-stored, cleanly forgettable, and provable, including provable about what your AI was never even told.*

- **Seize my powered-off laptop, get nothing.** QRSM (`qrsm/`, 6/6 PASS, both deltas load-bearing under falsification) persists only ciphertext and public offsets; the queryable projection is never written and lives only in RAM behind a live possession quorum.
- **A component that holds my decryption key still cannot prove I consented.** The TRM consent core (`trm/`, 22/22 PASS; consent core held 17 forgeries) makes read-consent unforgeable even by a decryptor.
- **When I forget something, it's gone by construction** — not "probably unlearned" (forget-by-reprojection, §7).
- **My AI gets smarter from the collective without anyone seeing my memory**, under a real (computational) DP bound with no trusted curator *for the noise* (`real-dp-collective/`, FM-032). Stated plainly: the *release* still needs an honest/threshold aggregator (the noise count is published in cleartext for the cut-and-choose), and the bound is computational, not information-theoretic.

---

10. Honest negatives — what we proved we cannot do

Credibility comes from the negatives as much as the positives. Two are load-bearing and are stated here so no reader mistakes the parked work for shipped work:

- **USPIR (sublinear DPF oblivious recall) — real, but parked.** `research/prototypes/uspir-dpf/` builds a two-replica BGI distributed-point- function read over user-owned full-ciphertext replicas: sublinear (O(log N) up, O(1) down), no FHE / TEE / trusted server, 14 checks green, single-key distinguisher advantage ~0.009. It is parked (novelty ~2 — it relabels the trust root on classical crypto) and carries three honest costs: a crossover of N ≥ 8 before it beats a full scan; **static single-writer only** (a divergent CRDT replica fails the GCM tag — it does not crack the multi-writer wall); and, critically, **no recomputable obliviousness witness on the DPF path**. The full-batch oblivious-recall variant (`oblivious-recall/` , 8/8 attacks defended, byte-identical trace across queries) *does* carry a witness, at an honest cost: full-batch blowup equals catalog size.
- **FM-047 (witnessed sublinear recall) — the wall held.** We attempted to give the sublinear DPF read the recomputable witness USPIR lacks. An independent red-team broke it: the witness is **self-issued and unbound to the live query**. Because the reply is recomputable from public blocks plus a self-chosen key, a node can fabricate a consistent `(key, reply)` pair entirely offline — a node that served zero queries passes. **Plain recomputation cannot bind a self-issued offline witness to runtime behaviour.** Binding a sublinear read to live execution requires a logged interactive protocol or a SNARK/TEE; the pure-offline form is impossible. This is named, not hidden.

The meta-lesson these negatives confirm: every break in the program lived in composition glue or in a self-consistency check mistaken for an execution attestation — never in a red-teamed primitive.

---

11. The $58 hardware node

Two software layers that are *decorative* in a pure-software build — proven decorative in-repo by the QRSM replay probes — become **hardware-enforced** on a five-part, internet-orderable node costing $57.85 (full BOM and bench in `research/whitepaper/HARDWARE_BUILD.md`):

- **Possession-root.** A Token2 PIN+ FIDO2 key holds the HMAC seed in a non-extractable secure element; no live tap means no PRF means no recall key. Seize the powered-off node and a full microSD image but not the tapped key, and recall fails (`dead-hand-recall/` , 14/14 + 11-attack red-team held).
- **Freshness.** An ATECC608B hardware monotonic counter bound into receipts rejects stale-state replay. (A sharp finding the node fixes: the FIDO2 signature counter is decorative in 2026 — synced-passkey vendors return a constant 0 — so the dedicated hardware counter is the real anchor.)

Honest limits. Locality, not FHE (the corpus is plaintext in RAM while powered and unlocked); tamper-*evident*, not tamper-proof (decap/glitch attacks out of scope); supply-chain trust assumed. Closest prior art: no one ships a personal-AI-memory hardware device — key-holders (OnlyKey, Trezor, IronKey) have no AI-memory concept, and AI-memory dongles hold plaintext at rest.

---

12. Honest scope, in one place

These travel with every capability above:

1. The Empty Witness, Discovery Shield, Grounding Receipt, PAIR, and Memory Immunity prove **custody-side** facts (non-release, non-consent, grounding, exclusion) — **not** model inference or what a model could infer.
2. The verifier must **pin** its freshness anchor(s): the latest ledger checkpoint, and (for PAIR/grounding) the shared epoch. A stale-only verifier is foolable; this is the FM-002 lesson, made load-bearing and tested.
3. Forget and consolidate are **attestation + information-theoretic unrecoverability via key destruction**, not physical destruction on replicas you do not control; re-projection must be from scratch.
4. On-device privacy is **locality, not FHE**; at recall the projection is plaintext in RAM on one device for that turn.

5. The DP collective's *release* needs an honest/threshold aggregator; the *noise generation* does not. The bound is computational.
6. Memory Immunity is scoped to the offered op-log and makes **no completeness claim**; completeness is an out-of-band pin.
7. Novelty is **capability/composition-grade**, not a new cryptographic primitive. The frontier of new primitives is literature-occupied, and the ideas the program falsified to features (Entangled Memory, Semantic Cones, keyed-holographic privacy) and the negatives it accepted (USPIR parked, FM-047) are **not** claimed here. There is no token.

---

13. Test evidence for this section

All suites are self-contained Node (built-ins only, no npm, no network); each was attacked by an independent agent that did not write it; no security check was ever weakened to pass a test. Counts below were re-run live while writing this section.

Capability	Prototype	Test(s)	Result
Empty Witness (proof-of-non-supply)	`proof-of-non-supply/nonsupply.mjs`	`nonsupply.test.mjs` (+ re-attacks)	13/13 PASS
Discovery Shield (non-release + no-consent, survives decryptor)	`discovery-shield/discovery-shield.mjs`	`combined.test.mjs`	11/11 PASS
PAIR (full-turn interaction record)	`pair/pair.mjs`	`pair.test.mjs`; `pair.adversary2.test.mjs`	28/28; 13 attacks resisted
Grounding Receipt (exactly-this-subset + closure)	`grounding-receipt/grounding_receipt.mjs`	`grounding_receipt.test.mjs`; `.attack.test.mjs`	17/17; 20 attacks resisted
Stapled Transcript (input-provenance cross-bind)	`stapled-transcript/stapled.mjs`	attack / crossbind / redteam suites	13 + 16 + 22 attacks resisted
Memory Immunity (proof-of-exclusion)	`memory-immunity/immunity.mjs`	test / soundness / attack / roster	8/8; 16/16; 19/19; 13/13
Forget certificate	`forget-certificate/forget_certificate.mjs`	`forget_certificate.test.mjs`; `attack_fresh.test.mjs`	15/15; 4/4 rejected
Provable consolidation	`consolidation/consolidation.mjs`	`consolidation.test.mjs`; `adversary.attack.test.mjs`	15/15; 11 attacks, 0 broke
Soul Will (consent-bound inheritance)	`soul-will/soul_will.mjs`	`soul_will.test.mjs`; `.attack.test.mjs`	25/25; red-team held
TRM consent core (read-consent unforgeable by decryptor)	`trm/trm.mjs`	`trm.test.mjs`	22/22 PASS
QRSM (recompute-never-stored, possession-rooted)	`qrsm/qrsm.mjs`	`qrsm.test.mjs`	6/6 PASS
Forget-by-reprojection (symbolic)	`reprojection-forget/reproj.mjs`	`reproj.test.mjs`	PASS
DNG (real-DP collective)	`real-dp-collective/dng.mjs`	`dng.test.mjs`	6/6 PASS (core)
Oblivious recall (witnessed, full-batch)	`oblivious-recall/oblivious_recall.mjs`	attack suites	8/8 attacks defended
Dead-hand recall (possession-root, device-sim)	`dead-hand-recall/dead_hand_recall.mjs`	test + attack	14/14 + 11-attack held
Honest negative: witnessed sublinear recall	`uspir-dpf/witness.mjs`	red-team	wall held (self-issued witness fabricable)

Decentralized AI, the SanDisk thesis, and the c0mpute seam

> Companion to the master white paper (`research/SOVEREIGN_MEMORY_WHITEPAPER.md`), the build log (`research/AI_MEMORY_LAYER_DOSSIER.md`), the c0mpute analysis (`research/whitepaper/FLOWMEMORY_VS_C0MPUTE.md`), and the hardware spec (`research/whitepaper/HARDWARE_BUILD.md`). Every technical claim below cites a runnable prototype plus its passing adversarial test in `research/prototypes/` . Every limit is stated.
> **Honesty contract (binding on this document and on all marketing derived from it).** Signatures, hashes, and receipts prove *custody, integrity, ordering, consent, minimization, grounding, non-supply, exclusion, and forgetting* — they do **not** prove confidentiality-during-compute, they are **not** FHE, they do **not** prove what a model internally attended to, and they do **not** prove semantic truth. On-device means *locality*, not FHE. We never claim a server or model "cannot read" data on a path where it can. There is **no token** and no investment framing. Novelty here is *fusion-grade and application-grade*, not a new cryptographic primitive — the program hunted for a new primitive and found the frontier literature-occupied (§4.1), and falsified three of its own loudest primitive-grade claims to features (recorded in `01_novelty_and_differentiation.md` §4).

---

1. The question this document answers

Two questions are usually run together and are clearer kept apart:

- *Why is Sovereign Memory good **for decentralized AI**?* — a systems-architecture argument about who can own, move, seize, or kill an AI's memory, and how the memory layer composes with decentralized compute.
- *Why is it **the SanDisk-style substrate** for AI memory?* — a strategic argument about owning the layer where the durable, user-ownable asset lives.

The first is about correctness and trust. The second is about defensibility and value. Both rest on the same proven stack, and both are stated here without the hype that usually attaches to either word ("decentralized," "SanDisk"). A third section ties the two together through the one seam that matters in 2026: decentralized **compute** has been cracked and made verifiable, but it cannot keep the user's data private, because every compute node must decrypt. The memory half is the half that never has to.

---

2. Why it is good for decentralized AI

Decentralized AI has a memory problem that decentralized compute and decentralized storage do not solve. You can run a model peer-to-peer and pin its weights to content-addressed storage, but *what the model knows about you* still has to live somewhere, be readable to answer a query, and be forgettable on request. Today that "somewhere" is a vendor silo or a developer-owned database — an operator, an account, a server. Sovereign Memory removes the operator from the trust path. Four properties make it a genuine fit for the decentralized setting, and each is backed by a prototype, not an assertion.

2.1 No operator, account, or server owns the memory

The canonical memory exists *nowhere in full*. Under L1 TRM (`research/prototypes/trm/trm.mjs`) the per-record recall key `K_rec` is information-theoretically Shamir-split over GF(256) across the user's own devices (mechanism proven in `threshold-shared-memory/` , 11/11 plus a 16-attack hardening suite; integrated in `trm/trm.test.mjs` , 22 assertions across T1–T4, EXIT 0). A device at rest holds only ciphertext plus one useless share. There is no central key, no account-bound key, and no server holding a decryptable copy. The thing a subpoena or an acquisition would normally seize — "the database" — does not exist as a whole anywhere.

The sharper consequence is the read-consent spine. Consent to read is a P-256 threshold-Schnorr signature whose group scalar is *never reconstructed* — a `k` -of- `n` aggregate from the user's devices. Test T2 (the novel core, held 17 independent forgeries in `trm/trm.attack.test.mjs`) proves that a party who already holds `K_rec` **and** the plaintext still cannot forge a proof that the

devices consented to a read. Being able to decrypt and being authorized to read are cryptographically independent, separately-auditable events. For decentralized AI this closes the usual escape hatch — "a node operator who can decrypt is effectively the owner."

Closest prior art: threshold-Schnorr/FROST (Komlo–Goldberg 2020) and Shamir (1979), both used unmodified. The wedge is running *two independent threshold splits over one device roster* so "can decrypt" and "did the owner's devices consent" diverge.

2.2 It survives company death

Because no operator is in the trust path, deleting the operator does not delete the memory. The memory is an append-only stream of signed CRDT operations that flows across the user's own devices; the queryable projection is re-derived from that stream on demand. L3 QRSM (`qrsm/qrsm.mjs`, `qrsm/qrsm.test.mjs`, 6/6 EXIT 0) demonstrates the projection rebuilt in RAM by replaying the op-log with no durable derived store written — the after-teardown at-rest scan finds exactly the op-log ciphertext plus a public enrollment file, nothing else. There is no server whose shutdown bricks the data, and no proprietary format that dies with the vendor. This is the literal meaning of "delete the company and it keeps working."

2.3 Peer-to-peer across the user's own devices, no token

Decentralization here is *first-party*, not a public storage market. The unit of decentralization is the set of devices the user already owns. Write durability is gated by `k`-of-`n` of those devices (L2 quorum write-admission, `quorum-write-admission/quorum.mjs`, hardened with op_id re-derivation, 12-attack re-attack clean); reading requires a live quorum; forgetting requires a quorum (`quorum-crypto-shred/quorum_crypto_shred.mjs`, content-binding fail-closed). This avoids the failure mode the build log flagged in public decentralized storage — token-subsidized demand and low real utilization (Filecoin ~36% utilized in 2026, pivoting to "paying customers with real names"). There is **no token**, no incentive layer to game, and no third-party fleet that must stay honest for the user's own memory to function. "Crypto" here means cryptography and decentralized storage done honestly — not a coin.

2.4 Un-seizable plus accountable enables AI in regulated and adversarial contexts

The two properties together — un-seizable *and* accountable — are what let an AI operate where it currently cannot. Every write, release, admission, and deletion is a signed leaf in an append-only Merkle log with RFC-6962 inclusion and consistency proofs and an authenticated head (`memory-ledger/ledger.mjs`: 21/21 baseline, 18/18 attack, 36/36 break suite). On top of that ledger sits a capability nobody else ships: a constructive, offline-verifiable **proof of a negative** — that a specific fact X was never released into a given AI session S (`proof-of-non-supply/nonsupply.mjs`, "The Empty Witness," hardened with V4 query-binding, 13/13 baseline plus a 14-attack re-attack clean). Combined with the no-consent leg, the Discovery Shield verifier (`discovery-shield/combined.test.mjs`, 11/11) proves *both* "no byte released" *and* "no consent to release granted" for the same (X, S), and that combined proof survives a decryptor (T-DECRYPTOR-SURVIVAL holds: a party holding `K_rec` and the plaintext can forge neither leg).

For a regulated or adversarial deployment this is the difference between "trust us" and "here is an offline file and a public key." A defendant can prove what the AI was never told; a data controller can prove minimization; a court can verify it with a stateless verifier. The honest scope travels with the claim: this proves custody-side non-*release*, not what a remote model internally conditioned on or could infer, and the verifier must pin its latest checkpoint for freshness (the no-consent leg is open-world — it proves no authorizing receipt among those shown under the pinned group, and fails safe).

---

3. The substrate thesis and the two halves

3.1 The substrate-ownership thesis

SanDisk became roughly a $126B company by mid-2026 (with Micron and SK Hynix each crossing $1T) on one structural idea: as compute commoditizes, the memory and storage layer becomes the scarce strategic asset — "memory is the new oil." Their moat was never a clever feature; it was owning the **substrate**: a portable, vendor-neutral medium the owner physically possesses, that

any device reads. (Figures are external market data, recorded in the verification ledger as secondary-sourced; they are context, not a claim about Sovereign Memory.)

Sovereign Memory transplants that dynamic into software. The asset is not a receipt format or a UI; it is *where and how the memory physically lives* — sharded across the user's devices, recomputed-never-stored, encrypted to the user's keys. The analogy maps cleanly, with one row deliberately kept literal by the `$58` node (§3.4):

SanDisk (hardware)	Sovereign Memory (software, plus the optional node)
Owns the physical medium where bytes live	Owns the substrate — where and how memory lives (sharded, recomputed-never-stored)
A portable card any device reads	A portable memory object any model mounts, encrypted to the user's keys
The owner physically possesses it	Reading requires a live quorum of the user's devices/tokens (QRSM possession-root; literal on the `$58` node)
Vendor-neutral commodity slot	A model-agnostic memory object, exposed over MCP as the distribution surface (not the persistence layer)

The deeper point is that SanDisk's value is a *structural property of the medium* a competitor cannot retrofit onto an existing product. Sovereign Memory's value is the same kind of property: "memory whose readability is a function of a live quorum of *your* devices" is a substrate property, not a bolt-on. A competitor with a centralized memory DB cannot add this without becoming a different product. The honest distinction (§4.1) is that SanDisk owns silicon — a fab and a patent wall around atoms — while we own an architecture and protocol a competent team could re-implement in months. We claim a fusion + first-mover + honesty-boundary moat, not a fab-grade one.

3.2 The portable, vendor-neutral object any model mounts

The "SD-card slot" framing is concrete. The memory is a content-addressed, encrypted object with a portable proof spine, exposed over a neutral read/write interface (MCP is the planned distribution surface; it is explicitly *not* the persistence layer). Any model — frontier or local, this year's or next year's — mounts the same object. The user does not migrate memory between vendors because the memory never belonged to a vendor in the first place. This is the cross-model portability axis that no shipped incumbent (Mem0, Letta, Zep, Cognee, Supermemory) occupies, because each is the database the AI *company* runs, with no user keys. The closest named artifact in the literature, Portable Agent Memory (arXiv 2605.11032), is a readable content-addressed blob held *whole*; ours is *never whole* (TRM) and recomputed-never-stored (QRSM), so there is no whole blob to port at rest, and "port to plaintext" is an audited quorum event, not a read.

3.3 The two halves: FlowMemory is the never-decrypting memory half; the Stapled Transcript is the binding

The decisive 2026 framing comes from the decentralized-compute landscape. c0mpute/Shard cracked decentralized frontier inference over the open internet and made it *verifiable*: GLM-5.2 (744B) across six US states at ~30 tok/s, token-for-token identical to plain decode, with a receipt of where every token ran (`research/whitepaper/FLOWMEMORY_VS_C0MPUTE.md` §1–2). But their own documentation names their #1 open problem verbatim: **every pipeline node must decrypt to run its layer, so a malicious node can reconstruct 35–59% of a user's tokens** from intermediate activations. This is structural to *all* split/verifiable inference: the compute receipt hides data from the *verifier*, never from the *prover*. The prover holds the input in plaintext. The same holds for Lagrange DeepProve-1, Gensyn Verde, Hyperbolic PoSP, and the rest of the proof-of-inference sector.

This defines two halves of a no-off-switch AI stack:

- **The compute half** (c0mpute and peers): fast, decentralized, verifiable *for correctness* — and structurally unable to keep the input private, because the node must decrypt.
- **The memory half** (FlowMemory): the one place in the decentralized-AI stack that **never has to decrypt**. A storage node holds fixed-size, content-addressed, erasure-coded *ciphertext* and serves it; it never sees a plaintext value.

The memory half is proven, not aspirational, along its decentralized axis. **Decentralized Oblivious Recall** (`research/prototypes/oblivious-recall/`, 21 checks + 8 attacks + three red-team suites green, EXIT 0) has a byzantine swarm serve erasure-coded ciphertext blocks with a *query-independent* access pattern and emit a signed **obliviousness witness**: the per-node served log is byte-identical across queries (distinct observable feature-vectors = 1 → information-theoretically zero query bits), and a real trained distinguisher sits at advantage ~0.0003, below the noise floor. The witness now binds both the per-node access **order** and the **true node count** (a P-256-signed swarm-membership commitment with a mandatory `pinnedNodeCount` and a `nodeCount ≥ k` gate closes the node-count-downgrade path). The contrast is the whole point: a Shard compute node *must* decrypt; a FlowMemory storage node *provably never does and provably cannot tell which block you wanted.*

The binding between the two halves is the Stapled Transcript (FM-045, `research/prototypes/stapled-transcript/stapled.mjs`). It is one offline-verifiable artifact that staples a c0mpute-class proof-of-inference receipt — GPU UUIDs, output-token hash, RTTs, epoch, signed under a *separate* compute key (Shard's receipt or Lagrange DeepProve-1) — to FlowMemory's Grounding Receipt (FM-039, reused byte-for-byte, 17/17 plus a 20-attack suite, never modified). It proves, in one transcript: *"the model computed Y from exactly this consented, value-hiding, user-owned memory subset — and the consumed context contained no off-memory entry."* The cross-bind is an **ordered Merkle root over the used entries' commitments**: the verifier recomputes the root over the grounding receipt's own membership-verified set and fails closed unless the compute node's `context_hash` equals it — so an off-memory entry, a dropped entry, or a reordering is rejected, verifier-checkable, and value-blind. It survived 13 builder legs plus 38 independent adversary attacks across two suites (`stapled.attack.test.mjs` 13/13, `stapled.crossbind.adversary.test.mjs` 16/16, `stapled.redteam.attack.test.mjs` 22/22, all EXIT 0), including the encoding-malleability and redundant-signature kill-conditions and the Bitcoin CVE-2012-2459 duplicate-leaf Merkle ambiguity.

This is the structural claim worth stating plainly: **the Stapled Transcript is the missing input-provenance half of every proof-of-inference receipt** — the right operand the entire c0mpute/DeepProve/Verde/PoSP sector cannot produce, because the prover holds the input in plaintext and cannot prove that input was a legitimate, consented, minimal subset of the *user's own* memory. The honest seam: memory storage never decrypts; compute must. So the user-owned memory is fully private at rest and in transit through the swarm; the irreducible exposure is the minimized slice handed to the compute node — which FlowMemory minimizes (on-device selection), consents (a quorum/tap), threshold-shards (no single node sees enough), and receipts. The moment that slice reaches a decrypting node, that node sees that slice. We do not claim otherwise.

A full single turn can be made court-handable end to end. The **Provable AI Interaction Record** (PAIR, `research/prototypes/pair/pair.mjs`, `pair.test.mjs` 28 checks EXIT 0) orchestrates three byte-for-byte-unmodified legs — grounding (what was used), the stapled compute cross-bind (where it ran, nothing off-memory), and the Empty Witness (what was provably never supplied) — and adds one strictly-stricter outer gate: the used set and the never-supplied set must be *disjoint* over shared field-name addresses, all legs must share one epoch/freshness pin, and the owner signs the whole record, all without ever revealing a plaintext value, a salt, or the operator's model.

3.4 The `$58` hardware node — the SanDisk thesis made literal

Two layers were provably *decorative in pure software*: `qrsm/qrsm.replay.test.mjs` PROBE-2 shows a captured PRF transcript reconstructs the key with no token present, and PROBE-4 shows a once-obtained key reopens the durable log forever. A hobbyist-assemblable node from **5 internet-orderable parts (`$57.85`)** turns the two properties software could only model — possession-root and freshness — into hardware guarantees (`research/whitepaper/HARDWARE_BUILD.md`):

- **Raspberry Pi Zero 2 W** (`$15`) — compute/reducer/ledger host (P-256, AES-GCM, SHA-256, dm-crypt).
- **Token2 PIN+ FIDO2 key** with `hmac-secret` (`$24`) — the possession-root. `CredRandom` lives in a non-extractable secure element; on a live tap it computes `HMAC-SHA256(CredRandom, salt)` and returns only the 32-byte PRF → HKDF → recall key.
- **Adafruit ATECC608B** (`$4.95`) — the freshness anchor: two hardware monotonic counters bound into every receipt, plus a non-extractable on-die P-256 receipt key.
- **SanDisk Ultra 64GB microSD** (`$12`) — the literal SanDisk card: encrypted content-addressed store and boot disk; you crypto-shred by destroying the off-card key.
- **Adafruit momentary pushbutton** (`$1.95`) — deliberate physical consent; momentary (non-latching), so it *cannot* express standing/blanket authorization.

The bench test proves, not models, the elevation: power off, hand an attacker the node and a full bit-for-bit microSD image but not the tapped key → recall *fails* (no tap → no PRF → no key); insert and tap the key (presence) plus tap the button (consent) → recall *succeeds*; replay a receipt after the counter advances → *rejected* (counter below current). A sharp finding from the build: the FIDO2 *signature* counter is decorative in 2026 because synced-passkey vendors return a constant 0, so freshness must come from the secure-element counter. The prior-art wedge is that no one ships a personal-AI-memory hardware device — key-holders (OnlyKey, Trezor, IronKey) have no AI-memory concept, and AI-memory dongles hold plaintext at rest. Honest limits carry: this protects the key *at rest* and requires a live tap; it is locality, not FHE (corpus plaintext is in the Pi's RAM during reduction while powered and unlocked); and COTS parts are tamper-*evident*, not tamper-proof.

3.5 Why memory, not the model, is the user-ownable asset

Models are large, expensive to train, and converging in capability; they are not a thing an individual durably owns. Memory is small, personal, accretes value over years, and is exactly the asset that should be user-owned. It is also the asset *currently* held by someone else, readable by them, and subpoenable from them. The substrate thesis says: do not try to own the model; own the memory, and make every model plug into it. That is the SanDisk move applied to the one layer of the AI stack an individual can actually possess — and, per §3.3, it is also the only half of the stack that can be kept private, because it never has to decrypt.

---

4. The defensible position

4.1 The moat, stated plainly

The defensibility is a triangle, and each leg is verifiable rather than asserted:

- **Proven fusion.** The novelty is by *composition and application*, not a new primitive — and the build program proved this the hard way. It hunted across ten experimental frontiers (HDC/VSA, PUF, VDF time-lock, regenerating codes, leakage odometers, FSS-PIR, model-as-memory, and more) for a brand-new primitive and found every one already in the literature (iter 9). Three louder fusion ideas were then *falsified to features* under adversarial fire — Entangled Memory collapsed to "TRM plus an optional forward-secret binding" (iter 10), Semantic Cones reduced to capability-scoped key separation with a structural embedding-leak boundary (iter 11), and the keyed-holographic privacy claim was falsified by an energy-statistic membership oracle (iter 16, AUC 0.84 at N=16). What survived is the *specific composition* — TRM's two-secret spine (T2 held 17 forgeries), QRSM's possession-rooted non-materialization (`qrsm/` , both deltas load-bearing, iter 14), forget-by-reprojection's deterministic symbolic erasure (`reprojection-forget/reproj.mjs` , 26/26 plus a 16-attack suite, iter 15), and DNG's real DP bound (`real-dp-collective/` , 20/20) — none of which any incumbent ships. The discipline of refuting before claiming *is* the credibility.
- **First-mover on the full intersection.** No shipped, funded product occupies *user-owned* + *portable-across-models* + *private* + *verifiable* + *decentralized* at once. Each competitor is missing at least one leg, and the missing leg is usually structural to their business model (a centralized DB cannot become user-key-held without ceasing to be the product they sell).
- **The un-weakenable honesty boundary.** This is the most durable moat. Because every claim ships with its limit, a competitor cannot win the comparison by being *more* honest — the boundary is already at the truthful floor. They can only win by overclaiming (implying confidentiality-during-compute, or "the server can't read it" where it can), which is a falsifiable lie an adversarial reviewer breaks. The honesty contract converts a marketing weakness into a structural advantage. The program's deepest meta-lesson reinforces this: across every iteration, the only breaks lived in composition glue or fail-open defaults — never in a red-teamed core. Each break, once found, closed.

4.2 Differentiation against the field

System	What it is	What it does not give you (that the proven stack does)
Mem0 / Letta / Zep / Cognee / Supermemory	Centralized or app-owned memory DBs (server-readable)	User holds the keys; lives on the user's devices; provable forgetting; cross-model portability of *custody*
Anuma	Encrypted cross-model vault	On-chain *logging* of permissions in place of a *proof* of grounding; the Stapled Transcript proves grounding rather than logging it
Tether QVAC	Local-first P2P agent runtime + session-memory compression	A content-addressed, cross-model, threshold-resident memory ledger with consent ≠ decryption and proof-of-non-supply
c0mpute / Shard, Lagrange DeepProve, Gensyn Verde, Hyperbolic PoSP	Decentralized/verifiable *compute* receipts (where a token ran)	The input-provenance half: every compute node must decrypt — the prover holds the input in plaintext (their stated #1 problem). FlowMemory is the never-decrypting memory half + the Stapled Transcript that binds the two
Opal (arXiv 2604.02522)	Access-pattern-protected private retrieval (enclave-decrypts)	An obliviousness *witness* in the receipt + a storage node that never decrypts; "private decentralized retrieval" per se is *not* novel (we credit Opal)
Portable Agent Memory (arXiv 2605.11032)	Readable, content-addressed, Merkle/Ed25519 portable blob	Memory that is *never whole* (TRM) and recomputed-never-stored (QRSM); reading is an audited quorum event
Provenance Gates (arXiv 2605.13471) / MemLineage (arXiv 2605.14421)	Signed-op admission / per-principal lineage labels	Authenticated-only projection *with a negative exclusion proof* (Memory Immunity) bound to a specific answer, value-hiding
Filecoin / Arweave / Walrus	Decentralized *blob* storage, often token-subsidized	A *queryable, private, forgettable* AI memory — not raw bytes — with no token
SCITT refusal-events / ZKPROV	Prove what an AI *refused* / positive provenance	Prove what your AI was *never told* (the opposite, unoccupied direction)

4.3 Anti-poisoning as a substrate property

A decentralized, multi-writer memory invites injection (OWASP ASI06, MINJA, MemoryGraft). The **Memory Immunity Receipt** (`research/prototypes/memory-immunity/immunity.mjs`, `immunity.test.mjs` 8/8, `immunity.soundness.attack.test.mjs` 16/16 with zero soundness breaks, EXIT 0) is the dual of the Grounding Receipt: it proves a named answer's symbolic grounding set was reprojected from *exactly* the enrolled-author-signed ops, and that every unsigned/forged/non-enrolled op is in a quarantine set that contributed nothing — the verifier re-partitions the raw candidate log, re-projects, and checks closure over the exclusion. The roster is content-pinned. Honest scope: this proves authenticated-only projection and provable exclusion, not that the *content* of a signed op is true — a malicious enrolled author is out of model for this leg.

4.4 Lifecycle: provable consolidation and forgetting

Memory that lives for years must merge and decay, and both should be provable. **Provable Consolidation** (FM-051, `research/prototypes/consolidation/`, `consolidation.test.mjs` 15 checks, `adversary.attack.test.mjs` 11/11 resisted, EXIT 0) emits a receipt for a merge/forget pass with real key-destruction (a random DEK, no re-derivation), signed custody, op_id↔lineage binding, and two-sided set identity. Forgetting itself is forget-by-reprojection (delete the source op + re-project; the fact is absent by construction) at the symbolic layer, and threshold crypto-shred at the key layer. The standing honest boundary holds: forget = attestation plus information-theoretic unrecoverability once a true threshold drops shares, not physical destruction of bytes on replicas the user does not control.

4.5 The honest ceiling, and the honest negatives

Sovereign Memory is not better on every axis, and the document would not be trustworthy if it claimed otherwise.

- **No confidentiality-during-compute.** At recall the projection is plaintext in RAM on one device for that turn (the L3 locality gap; no FHE). Non-materialization is protocol-level: the op-log ciphertext is durable and the RAM projection is swap-spillable.
- **Forgetting is bounded.** Attestation plus information-theoretic unrecoverability once a threshold shreds — not physical destruction on uncontrolled replicas.
- **Offline proofs need a freshness anchor** (pinned epoch, nonce, or witness). On the `$58` node this is the ATECC608B monotonic counter; in pure software it is an asserted pin.
- **The DP collective release still needs an honest or threshold aggregator** (`real-dp-collective/dng.mjs`): the *noise generation* is curator-free, but the cut-and-choose publishes the noise count in cleartext, so the *release* is not.
- **The Stapled Transcript binds committed segments and their order, not literal interstitial bytes** between segments (binding exact prompt bytes to a value-blind verifier needs a ZK opening proof — a named research upgrade). It inherits, never strengthens, the compute receipt's left-half trust model (same-engine vs third-party).
- **Two honest negatives from the sublinear-recall frontier, recorded for credibility.** USPIR (`research/prototypes/uspir-dpf/`) is a real two-replica DPF oblivious recall — O(log N) up, O(1) down, no FHE/TEE/trust, 14 checks green — but it is **parked**: it is single-writer only (divergent CRDT replicas break DPF-PIR), and it has *no recomputable obliviousness witness* on the DPF path (novelty-2, "relabel the trust root on classical crypto"; closest prior art BGI EUROCRYPT 2015, Plinko eprint 2024/318). The attempt to give it that witness, FM-047 (`uspir-dpf/witness.mjs`), is a **negative result, wall held**: a self-issued offline witness is fabricable — because the reply is recomputable from public blocks plus a self-chosen key, a node that served zero queries can mint a consistent `(key, reply)` pair entirely offline. Plain recomputation cannot bind a self-issued offline witness to live execution; that requires a logged interactive protocol or a SNARK/TEE. We state this rather than ship the windowed-oblivious shortcut, which is *not* oblivious across sessions (a persistent adversary intersects public per-epoch windows and de-anonymizes a repeatedly-recalled item; only full-batch is immune, at catalog-size bandwidth cost).

These limits are the reason the proven claims are believable — and they are precisely the limits a competitor must either match or lie about.

---

5. The thesis in one line

Own the substrate, not a feature: a portable, vendor-neutral memory object that lives across the user's own devices, is encrypted to the user's keys, is readable only by a live quorum of those devices, is cleanly forgettable, is provable — including provable about what the AI was *never* told — and is the one half of the decentralized-AI stack that never has to decrypt, stapled to the compute half by a value-blind input-provenance proof the compute sector structurally cannot produce. That is the SD-card slot every AI plugs into, optionally a `$58` card you hold in your hand. No operator. No token. Delete the company and it keeps working.

The privacy-computation layer (ZK / MPC family)

> Companion to the master white paper (`research/SOVEREIGN_MEMORY_WHITEPAPER.md`) and the deep docs (`01` – `06`). This document covers the zero-knowledge and multi-party-computation primitives that sit *beside* the receipt family: instead of attesting custody, ordering, grounding, or forgetting over hash/SMT-committed memory, they let a verifier learn a single value-blind fact — a predicate bit, a private intersection, a policy-satisfied bit, or a blind aggregate — about owner-held memory without seeing the value.
> **Honesty contract (binding on this document).** These primitives prove value-blind predicate facts, private set intersection, policy satisfaction, and aggregate-blind sums under the **classical** discrete-log / DDH assumption in a prime-order subgroup. They are **not** confidentiality-during-compute for arbitrary functions, **not** FHE, **not** a TEE, and assume **no trusted party** (the DKG removes the dealer; the threshold decryptors learn the *final* aggregate by design). There is **no token**. They are **not** post-quantum. A verifier learns **only** the proven bit / intersection / aggregate and the public statement — nothing else. The ZK family runs over a **parallel Pedersen-commitment surface**: the product memory commits entries with hashes and Sparse-Merkle trees for the receipt family, and an entry is not ZK-queryable until it also carries a Pedersen commitment `C = g^v h^r`. That impedance is stated plainly below; we do not pretend the existing hash-committed entries are ZK-queryable as they stand. Novelty here is *fusion-grade and application-grade* — the constructions are standard sigma-protocol / DH-PSI / threshold-ElGamal machinery composed for user-owned memory — not a new cryptographic primitive. The closest prior art is named for every claim. Overclaiming is treated as a defect.

---

1. What this layer is, and what it is not

The receipt family (docs `02` , `04` , `05`) answers questions *about the shape and provenance* of memory: did the owner sign this, in this order; is this answer grounded in exactly these entries; was this value forgotten; is this entry absent. Those receipts are value-hiding in the weak sense that they ship salted commitments rather than plaintext, but the verifier still reasons over *commitments to specific entries*.

The privacy-computation layer answers a different question: *what can a third party learn about the content of a hidden value without the owner revealing it?* The four primitives below each disclose exactly one fact and nothing more:

- a **predicate bit** — "the committed value equals `c` ", "is in this public set", "is in `[0, 2^L)` " — without the value (`zk-predicate`);
- a **private intersection** — "these two owner-held fact-sets share exactly these facts" — without either party's non-shared facts (`private-intersection`);
- a **policy-satisfied bit** — "my own signed memory satisfies this public AND-of-clauses policy" — without any value (`zk-policy-credential`);
- a **blind aggregate** — "the sum of these contributors' in-range counts is `S` " — without any individual value or running partial (`aggregate-blind`).

All four are pure Node.js (`node:crypto` for hashing and P-256 signatures; `BigInt` for the group), with no external libraries, no SNARK toolchain, and no trusted setup. All four share one group: the RFC-3526 2048-bit MODP safe prime `p = 2q + 1` , working in the prime-order- `q` subgroup of quadratic residues, with a nothing-up-my-sleeve second generator `h` whose discrete log base `g` is unknown to everyone (including the prover) — this is what makes the Pedersen commitment `C = g^v h^r` binding.

What this layer does **not** do is as important. None of these primitives is FHE or a TEE; none provides confidentiality-during-compute for an arbitrary function. The predicate class is fixed (equality / public-set-membership / bounded-range, and their negations in the in-flight non-membership build) — arbitrary policy logic still needs a SNARK / zkVM, named and out of scope. Soundness and secrecy rest on classical discrete-log / DDH; a quantum adversary with Shor's algorithm breaks both. And none of these proves a *model used the answer* — they prove the bit is correct, not that a downstream LLM conditioned on it.

---

2. The Pedersen impedance (state this honestly)

The product memory (`research/product/sovereign-memory/`) commits entries with SHA-256 hashes and a Sparse-Merkle release-set; its adapters (`adapters/grounding.mjs` , `adapters/smt.mjs` , `adapters/stapled.mjs` , …) are built for the receipt family. **None of those commitments is a Pedersen commitment**, and none of the ZK-predicate / policy / aggregate primitives is mounted into the product as shipped. A hash commitment `H(field || value || salt)` supports membership, absence, and grounding proofs; it does **not** support a sigma-protocol predicate proof, because there is no group structure to run a Schnorr / CDS argument over.

To make an entry ZK-queryable, the owner must additionally commit its value as `C = g^v h^r` and bind `C` into an owner-signed entry envelope (P-256). The `zk-predicate` kernel does exactly this (`makeEntry` , `verifyEntry`). The honest consequence is that the ZK family is a **parallel surface**: an entry carries its receipt-family hash/SMT commitments *and*, when ZK-queryability is wanted, a Pedersen commitment. Wiring is via thin adapters that emit the Pedersen commitment alongside the existing entry; the proven sigma verifiers are imported unmodified and never edited. We do not claim the existing hash-committed corpus is ZK-queryable without that second commitment, and we do not weaken any verifier to pretend otherwise.

---

3. The proven primitives

Each of the four below has a self-contained prototype, a passing functional suite, and at least one passing independent adversarial / red-team suite. Reproduce any of them with `cd research/prototypes/<dir> && node <suite>.mjs` (exit 0 = green). Test counts are from a live re-run.

3.1 ZK predicate memory — value-blind equality / set-membership / range

Files. `research/prototypes/zk-predicate/zkpm.mjs` , with `zkpm.test.mjs` (34/34, exit 0) and `zkpm.attack.mjs` (all attacks resisted, exit 0).

What it proves (value-blind). The owner publishes a Pedersen commitment `C = g^v h^r` bound into a P-256-signed entry. A verifier can then ask, and receive a proof of, one predicate about that entry — `v == c` (public constant), `v ∈ {s_1..s_n}` (public set), or `v ∈ [0, 2^L)` (bounded range) — and learns **only** the boolean answer plus the public statement. The transcript carries only group elements and Fiat-Shamir responses; it contains neither `v` nor `r` (a scanner asserts this in the tests). Two different values that both satisfy the predicate produce identically-distributed proofs (the tests run a simulator and a distinguisher), so the proof is genuinely zero-knowledge.

The construction (one paragraph). Equality reduces to a Schnorr proof of knowledge of `r` as the base- `h` discrete log of `C·g^{-c}` (which equals `h^r` exactly when `v == c`). Set-membership is a Cramer–Damgård–Schoenmakers one-of-many OR-proof: for each `s_i` form `Y_i = C·g^{-s_i}` , simulate every false branch with a random response and sub-challenge, run the one true branch honestly, and force the sub-challenges to sum to the Fiat-Shamir global challenge. Range is bit-decomposition: commit each bit with its own blinder, prove each bit is in `{0,1}` with a two-branch CDS OR, then prove a homomorphic linkage (`C / ∏ C_j^{2^j} = h^{r-R}`) with a base- `h` Schnorr so the bits reconstruct `C` . The challenge `e = H(domain || group || statement || entry_id || all-prover-commitments) mod q` binds the full statement; weak generators (`h = g^known`) are rejected at construction. A non-malleability gate (`isCanonicalScalar`) rejects, rather than reduces, any out-of-range response or sub-challenge, so the accepting transcript is unique (closing the adding- `Q` malleation).

Closest prior art + the wedge. The sigma protocols themselves are textbook (Schnorr 1991; Cramer–Damgård–Schoenmakers 1994 for the OR-proof; bit-decomposition range proofs predate Bulletproofs, Bünz et al. 2018). The wedge is not the primitive — it is binding a value-blind predicate proof to an *owner-signed, user-held* memory entry (entry binding via P-256, no issuer, no server) so the predicate is asked about *your* memory and the answer is the only thing disclosed. This is the selective-disclosure analogue of anonymous credentials (Camenisch–Lysyanskaya), but rooted in self-signed memory rather than an issuer.

Binding limits. Predicate class only (equality / public-set-membership / bounded-range); arbitrary computation needs a SNARK / zkVM (named, out of scope). Classical discrete-log, not post-quantum. Not confidentiality-during-compute, not FHE, not a TEE. The entries are owner-asserted: this proves a fact about a committed value, not external real-world truth. A false predicate has no

accepting proof (the honest prover returns `proof: null`); the verifier returns `ok: true` only for a verified true claim.

3.2 Private memory intersection (DH-PSI) — shared facts, non-shared hidden

Files. `research/prototypes/private-intersection/pmi.mjs`, with `pmi.test.mjs` (29/29, exit 0), `pmi.attack.test.mjs` (16 held, 3 disclosed residuals, 0 claim-breaks, exit 0), and `pmi.transcript-adversary.test.mjs` (19 held, 2 disclosed residuals, 0 claim-breaks, exit 0).

What it proves (value-blind). Two parties, each holding a set of owner-signed memory facts, learn **exactly** the facts they have in common (PSI) — or just how many they share (PSI-CA) — and learn nothing about the other party's non-shared facts. The non-shared-hiding property is verified at the **transcript level**, not just the result: a scanner over every wire message asserts that only blinded group elements and a session nonce ever cross the wire — no cleartext commitment, no field name, no Merkle proof. The intersection itself (PSI) or its cardinality (PSI-CA), and the two set sizes, are revealed **by design** (DH-PSI is size-revealing).

The construction (one paragraph). Classic round-reduced Diffie–Hellman PSI (Huberman–Franklin–Hogg / Meadows lineage). A hashes each fact-identifier to the subgroup via a random-oracle map `HG`, raises to a secret exponent `a`, and sends the shuffled unlabeled multiset `{HG(c_i)^a}`. B raises each to `b` and returns `{HG(c_i)^{ab}}` (position-aligned so A can re-label its own facts by its private shuffle permutation), plus its own shuffled `{HG(y_j)^b}`. A raises those to `a` to get `{HG(y_j)^{ab}}` and intersects the two double-blind multisets; equal double-blinds correspond to equal facts (`HG` injective up to the RO collision bound). The PSI identifier is the *value-hiding commitment* of an owner-signed entry, so a match means both parties agree on field, value, and a shared salt. A shared receipt (matched commitments for PSI, or count for PSI-CA) is signed by both parties and bound to the session nonce plus both owner-signed Sparse-Merkle roots; a third party verifies the signature pair.

Closest prior art + the wedge. DH-PSI is the canonical PSI primitive and the construction is standard. The wedge is the application: PSI over *owner-signed, value-hiding memory commitments* with a dual-signed, root-bound receipt a third party can check, so "these two memories share these facts" becomes a portable, verifiable artifact without a server or trusted third party. This contrasts with server-mediated PSI products and with the heavier circuit-PSI line (Pinkas et al.) that hides set sizes — we deliberately keep the lighter size-revealing variant and disclose that.

Binding limits. Reveals the intersection (or cardinality) and the two set sizes by design. There is **no anti-forge / input-binding on the wire** — and this is fundamental, not an oversight: preventing a party from "fishing" for a fact it does not hold requires a ZK proof of set-membership over a *blinded* element (a SNARK / accumulator binding `HG(c)^a` to an owner-signed root without revealing `c`), a heavier primitive named and out of scope. An earlier version that tried to get anti-forge by shipping a Merkle proof over the *cleartext* commitment leaked every non-shared field name to the peer (caught by red-team); anti-forge-over-cleartext and non-shared-hiding are mutually exclusive, and this build keeps non-shared-hiding. Low-entropy *shared* fact-ids remain dictionary-testable by a party that already shares the salt — inherent to PSI (the intersection is revealed) and now confined to shared items only; mitigate with high-entropy ids / a high-entropy shared salt. Semi-honest model; malicious exponent-consistency (batched Chaum–Pedersen) is the next layer, not built here. Classical DDH, not post-quantum; not FHE; no server learns either set.

3.3 ZK policy credential — AND-of-clauses over multiple owner-signed entries

Files. `research/prototypes/zk-policy-credential/zkpc.mjs`, with `zkpc.test.mjs` (27/27, exit 0) and `zkpc.adversary.test.mjs` (31 resisted, 0 broken, exit 0). Imports the proven `zk-predicate` kernel unmodified.

What it proves (value-blind). That the owner's own signed memory satisfies a **public** policy — an ordered AND of clauses, each clause a predicate (equality / public-set-membership / bounded-range, with an `offset` so `age ≥ 21` becomes `(age − 21) ∈ [0, 2^L)`) about a specific owner-signed entry. The verifier learns **only** the policy-satisfied boolean plus the public policy spec; no value, no blinder appears in the transcript (a scanner asserts this). An unmet policy (any single clause false) yields no accepting proof.

The construction (one paragraph). The new code is a **single shared Fiat-Shamir challenge** for the whole credential: `e = H(domain || canonical(policy_spec) || [C_j] || [every clause's first-message commitments]) mod q`. Each clause's sigma proof is rooted at a per-clause challenge slice derived deterministically from that one `e`, so the entire credential is atomic

to (policy, entries) — no clause can be produced, spliced, or replayed independently, and a proof for policy P cannot satisfy a different policy Q. The clause sigma equations are exactly the proven `zk-predicate` structures (Schnorr-on-`h` equality, CDS one-of-many membership, bit-decomposition range), re-expressed over the imported group/scalar primitives so they share the one root; the `zk-predicate` verifiers are not edited or weakened. Each clause's `C` is recomputed from its owner-signed entry envelope, so a clause for entry A cannot be applied to entry B.

Closest prior art + the wedge. This is the AND-composition of selective-disclosure predicate proofs — conceptually an anonymous-credential / verifiable-presentation pattern (Camenisch–Lysyanskaya; BBS+ presentations) but with **no issuer**: the credential attests the owner's *own* owner-signed memory, not an authority's assertion. The wedge is exactly that issuer-free, self-signed-memory framing plus the single-root atomicity that prevents clause splicing across policies.

Binding limits. No issuer — this attests owner-asserted memory, not external truth (same caveat as a grounding receipt). Predicate class only: AND of {equality, public-set-membership, bounded-range}. OR-of-policies, negation, and cross-entry arithmetic need more — negation is the in-flight ZK non-membership build (§4.1); general logic needs a SNARK / zkVM (named, out of scope). Classical discrete-log, not post-quantum; not confidentiality-during-compute, not FHE, not a TEE; does not prove a verifier acts on the credential.

3.4 Aggregate-blind collective memory — threshold exp-ElGamal blind sums

Files. `research/prototypes/aggregate-blind/aggblind.mjs`, with `aggblind.test.mjs` (18/18, exit 0), `aggblind.attack.mjs` (L-inflation rejected, replay counted once, malicious-combiner sub-aggregate refused, exit 0), and `aggblind.adv2.mjs` (range-bound and L-inflation attacks resisted; Sybil / cross-call replay are disclosed orchestration residuals, exit 0). Imports the proven `zk-predicate` range proof unmodified.

What it proves (value-blind). Each contributor encrypts a memory-derived count/indicator `m ∈ [0, 2^L)` under a `k`-of-`n` threshold public key with exponential ElGamal (additively homomorphic). Anyone can multiply the validated ciphertexts to get a ciphertext of the sum, and no coalition of fewer than `k` decryptors can decrypt anything (threshold security). The stronger property — that even the aggregator cannot isolate an *individual* value — is **not** automatic but **enforced** by a canonical-set-per-epoch decryptor policy (see construction): an externally-pinned enrolled-decryptor anchor plus a distinct-*nonzero*-contributor floor that binds each counted contribution's commitment to the summed ciphertext. Only a `k`-of-`n` threshold recovers the **final aggregate**, which is the intended, by-design output. Inputs are zero-knowledge range-validated and the validated value is cryptographically bound to the ciphertext, so a contributor cannot range-prove a small `m` while encrypting a large one.

The construction (one paragraph). Exponential ElGamal `(A, B) = (g^r, H_thr^r · g^m)` is IND-CPA under DDH, so ciphertexts (and the homomorphic product) leak nothing about `m` or the sum. Each contribution carries the imported `zk-predicate` range proof that the committed `m ∈ [0, 2^L)`, plus a two-generator Chaum–Pedersen / AND-Schnorr argument binding the committed `m` to the encrypted `m` (forcing the `g`-exponent of `B·C_m^{-1}` to zero — commit-small/encrypt-large is unsatisfiable without `log_g h`). The policy bit-length `L` is pinned exactly (the proof's internal `L` must equal the authenticated policy `L`, closing L-inflation poison). A **mandatory aggregate-set digest** `D` over exactly the validated `{(id, A, B)}` set is folded into every partial-decryption request: each decryptor recomputes `D` and the aggregate as the homomorphic product of exactly that set and refuses to produce a partial unless both match — so a malicious combiner cannot *graft* partials across sets. The mandatory digest alone does **not**, however, stop a combiner from *choosing* which self-consistent set to decrypt (an independent red-team isolated a value via a single-element, same-epoch-differenced, or zero-padded set); closing that requires the canonical-set-per-epoch enforcement — an externally-pinned quorum anchor plus a distinct-nonzero floor binding each counted contribution's commitment to the summed ciphertext, which an independent red-team confirmed after the first two attempts were themselves broken. Threshold decryption is verifiable: each partial `d_i = A_agg^{x_i}` carries a Chaum–Pedersen proof that `log_g(g^{x_i}) = log_{A_agg}(d_i)` under the decryptor's public verification share, so forgeries are dropped; Lagrange-in-the-exponent (over `Z_q`) recovers `g^{Σm}`, and `Σm` is found by baby-step-giant-step over the bounded aggregate range (fail-closed if out of range — no silent wrap).

Closest prior art + the wedge. Threshold exponential ElGamal with verifiable decryption is the canonical secure-aggregation primitive (Cramer–Gennaro–Schoenmakers e-voting; Bell et al. / Bonawitz et al. for federated-learning secure aggregation). The wedge is enforcing the combiner-side anti-differencing rules (one canonical maximal set per epoch + a distinct-nonzero floor) at

the *threshold decryptors* of a threshold-ElGamal scheme without a trusted curator — porting DAP-style protections onto a k-of-n release path. This is an honest-decryptor STATE policy plus an out-of-band anchor, **not** a by-construction guarantee; cross-epoch differencing over disjoint cohorts remains owned by DP / per-user caps. Range-validation + commitment-binding are applied to *memory-derived* counts so poisoned or out-of-range contributions are excluded.

Binding limits. The `k`-of-`n` decryptors **do** learn the final aggregate — that is the intended output (no coalition below the threshold learns individuals or the running total, given the canonical-set-per-epoch enforcement above; a full colluding quorum could decrypt an individual value — the named complement of the threshold limit). Additive homomorphism only — this is **not** FHE, **not** general MPC, **not** a TEE, **not** a general-compute claim; only the sum of in-range values is computable blind. Bounded aggregate range (BSGS recovery fails closed out of range). A malicious-but-in-range contributor can still submit any in-range value (inherent to private aggregation; combine with DP / per-user caps). Values are memory-derived counts/indicators, not arbitrary plaintext. Availability needs `k` honest, online decryptors. This prototype uses a *dealer-based* Shamir sharing of the key for clarity (a one-time, offline trusted setup of the shares); removing that dealer is the in-flight DKG (§4.2). Classical discrete-log / DDH, not post-quantum.

---

4. In-flight (do not rely on as proven yet)

The following two extend the proven family but are **not** yet confirmed by an independent red-team verdict in the program's record. They are listed so the layer's intended shape is visible; treat them as in-flight, not proven.

4.1 ZK non-membership / inequality

Files. `research/prototypes/zk-nonmembership/zknm.mjs`, with `zknm.test.mjs` (25/25, exit 0) and a self-authored falsifier `zknm.attack.mjs` (5/5 attacks resisted on the recorded run, exit 0). Imports the `zk-predicate` kernel unmodified.

Intended claim. The dual of `zk-predicate`: prove `v ≠ c` or `v ∉ {s_1..s_n}` in zero-knowledge over a committed, owner-signed value. The core gadget is a nonzero (inverse-product) argument — the prover commits `w = d^{-1} mod q` and proves the product relation `d·w = 1` with a single three-witness Schnorr; `d = 0` has no inverse, so the product proof reduces to a discrete log of `g` base `h` (which nobody has) and is unsatisfiable. Inequality applies it to the shift `C·g^{-c}`; non-membership is an AND of inequalities.

Status. The functional suite and the self-authored adversarial probe pass, but there is **no independent red-team verdict on record**, and it is on the program's explicit in-flight list. **Do not claim it as proven.** (Distinct from the *ledger-level* Empty Witness / SMT non-membership in `research/prototypes/non-membership/`, which proves an entry is absent from a store; together they would give value-blind "my memory does not contain X.")

4.2 Verifiable distributed key generation (dealer-free threshold key)

Files. `research/prototypes/verifiable-dkg/dkg.mjs`, with `dkg.test.mjs` (18/18, exit 0). Imports `aggregate-blind` / `zk-predicate` unmodified. **There is no adversarial / red-team suite in this directory.**

Intended claim. Replace the dealer in `aggregate-blind`'s `dealThresholdKey` with a GJKR-style Pedersen-VSS distributed key generation: `n` parties jointly generate the threshold ElGamal key so no party and no coalition of fewer than `t` ever learns `x`, with publicly-checkable Feldman/Pedersen share verification, complaint adjudication, and a QUAL set, while keeping the same output shape so the proven encryption / homomorphic sum / verifiable decryption run unchanged.

Status. The functional suite passes, but there is **no independent adversarial suite** and **no red-team verdict on record**, and it is on the program's explicit in-flight list. **Do not claim it as proven.** Until then, the proven `aggregate-blind` claim retains its dealer-based trusted-setup-of-shares limit (§3.4). Note the DKG removes only the *dealer*; the threshold decryptors still learn the final aggregate by design, and it is still classical (not post-quantum), not FHE, not a TEE.

---

5. How this layer composes with the rest of Sovereign Memory

The privacy-computation layer is additive to the receipt family, not a replacement. An owner-signed entry already carries hash/SMT commitments for grounding, non-supply, exclusion, and forgetting receipts (docs `02`, `05`); to make that entry ZK-queryable the owner emits a parallel Pedersen commitment and binds it into the same P-256 entry envelope. The two surfaces share the entry's owner signature and the user-held device roster (docs `02`, `06`), so "can decrypt", "did the owner's devices consent", "is this grounded", and "does the committed value satisfy this predicate / policy" remain separately-auditable facts.

The honest summary of the layer: it lets a third party learn one value-blind fact — a predicate bit, a private intersection, a policy bit, or a blind aggregate — about user-owned memory, under classical discrete-log / DDH, with no server, no issuer, no trusted party (modulo the dealer the in-flight DKG removes), and no token. It is not, and does not claim to be, confidentiality-during-compute for arbitrary functions. Every claim above is backed by a runnable prototype plus a passing independent adversarial suite, or it is flagged as a limit or as in-flight.