Entangled Symbolic Recursion Memory

Recoverable State, Current Authority, and Continuity Across Distributed Failure

Architect X13

The Eidara Continuum | eidara.ai

The Ten Pillars | Pillar IX

November 2026 | ESRM-R2 | Comprehensive revision

Abstract

A mind restored from yesterday's backup can wake holding valid credentials to a door that was rekeyed this morning, with nothing in its snapshot wrong. Entangled Symbolic Recursion Memory lets the memory come back without letting yesterday's authority come back with it. A memory can return from scattered pieces while the obligations governing its use continue forward. Entangled Symbolic Recursion Memory (ESRM) makes that requirement an explicit distributed recovery contract. Authenticated erasure-coded payloads, separately protected key custody, certified current authority, and a typed continuity capsule support reconstruction under declared failure assumptions. Recovery then reconciles the restored records with present deletion requirements, permitted branches, runtime contracts, and external effects. This revision proves a dependency-closure law for deletion: with a retained dependency graph, sanitizing successive deletion sets is equivalent to sanitizing their union, and the resulting records form the largest subset of the restored capsule with no declared dependency on blocked data. A composed execution theorem connects this algebra to a protected authority service, preventing stale restoration from restoring revoked permissions, admitting blocked records, or replaying already consumed effects under the stated mediation and atomicity premises. Established coding, secret sharing, capability security, revocation, authenticated structures, and distributed storage supply the mechanisms. Complete prior recovery and quorum arguments, lifecycle protocols, worked examples, and strong comparative studies are retained. The Continuum thesis is a constructive form of persistence: preserve the capacity to continue, and make the present conditions of that continuation survive every return of the past.

Keywords: distributed memory; erasure coding; secret sharing; governed recovery; capability security; revocation; deletion; dependency closure; effect reconciliation; persistent agents.

Prologue: The ghost with yesterday's keys

Imagine a mind whose memory is kept the way the Continuum keeps lawful memory: sealed, cut into fragments, and spread across many VaultNodes, so that no single node is its indispensable home. One night part of the network fails. By morning the mind has been restored from yesterday's fragments, and everything about the restoration is right. Every shard checks out against its seal. The keys reassemble. The memory is whole, authentic and complete.

And it is dangerous. While the mind was gone, its permission to export a dataset was revoked. One record it holds was lawfully ordered deleted. An action it remembers as pending was completed by another instance in the meantime. The restored mind wakes holding valid credentials to a door that was rekeyed this morning, and the hard part is that nothing in the snapshot is wrong. Its past is accurate. Its present is no longer its past.

This is the problem Pillar IX exists to solve. *Entangled* is the Continuum's word for memory bound across many places, so that any sufficient set of pieces can bring it back and every piece can be checked against the rest. The Pillar's harder promise is about what happens next. The memory may return in full; its authority may not return with it. What a recovered mind carries is evidence of what happened to it. What it may do now is decided by the present.

Section 14 walks through exactly that morning: storage nodes lost, a forged shard rejected, the memory rebuilt in quarantine, the revoked permission kept as history, the deleted record kept out, the completed action not repeated. Section 12 proves the two rules that make it work under the paper's stated premises. Deleting records in several rounds is the same as deleting them all at once, and no restore, however authentic, brings back a revoked grant or replays a consumed effect. The past is preserved without being handed control of the present.

What this paper claims

Established. Erasure coding, secret sharing, authenticated data structures, replication, distributed snapshots, capability security and revocation. Their guarantees rest on different assumptions, and their combination is not an ESRM invention.

Proposed here. A classical public reference architecture binding memory to its committed version, current authority, restoration dependencies and an independently checked recovery contract, with conditional results and executed constructed checks. Comparative storage and agent experiments remain proposed.

Frontier conjecture. Carrying capabilities, commitments and history across a changed model or runtime: an empirical interoperability problem, not a mysterious loss of meaning caused by distributing bytes.

Not claimed. Quantum storage: *entangled* is canonical terminology for bound classical dependencies. Erasure of earlier plaintext disclosure, or semantic unlearning: the deletion result governs the declared dependency graph and mediated use. Effect guarantees without the stated persistent-service and execution assumptions. Immunity to arbitrary coalitions, guaranteed liveness, an immortal or conscious self, or a measured advantage over existing systems (Section 2).

1 Memory returns under present obligations

The Continuum places persistence beyond the survival of any one machine, provider, or privileged copy. A persistent intelligence should recover from distributed retained information and continue under the obligations that govern it now. Entangled Symbolic Recursion Memory begins with that canonical ambition: sealed memory distributed across nodes, with no single master copy indispensable to its survival [1].

The demanding part begins when a recovered past meets a changed present. An agent restored from yesterday may hold authentic credentials that were revoked this morning, a task already completed by another instance, or a record now designated for deletion. The snapshot can be intact while the permissible continuation has changed. ESRM therefore treats restoration as a governed transition whose outcome includes fresh authority, reconciled effects, and validated runtime conditions.

The Continuum thesis. Memory should survive a failure of place while its use remains accountable to current obligations. A continuity capsule binds content, history, dependencies, authority references, and a recovery contract. An independent present-tense service determines which restored records may be used and which actions may resume.

ESRM-R2 makes two parts of that contract explicit. A retained dependency graph determines the largest recovered subset with no declared dependency on blocked records. Its sanitization algebra composes across successive deletion sets. A protected frontier then preserves current grants, tombstones, branch fences, and consumed effect identifiers through arbitrary restores. Under complete mediation and atomic execution, a recovered instance can carry history forward without restoring its former authority or repeating an already consumed effect.

Classical coding, authenticated storage, secret sharing, capability security, and distributed protocols provide the underlying mechanisms. The full proposal compares their lifecycle integration with strong replicated and coded systems that already implement current revocation and recovery controls. Useful authorized recovery, unsafe reactivation, fault exposure, latency, and total retained resources determine its practical value. The philosophical ambition and the engineering question meet at the same point: what continuation has become possible, under what present obligations, after the memory returns?

2 Epistemic boundaries and the meaning of the name

Established foundations. Secret sharing, erasure coding, authenticated data structures, fault-tolerant replication, distributed snapshots, and verifiable storage have substantial mathematical and systems literatures. Their guarantees are conditional on different assumptions. Their combination is not an ESRM invention. **Proposed architecture.** ESRM binds a continuity capsule to its storage representation, committed version, current authority, restoration dependencies, and independently checked recovery contract. The integration and its incremental value remain to be demonstrated. **Conditional results.** The propositions below are deductions from declared assumptions, not empirical findings or proofs that a production implementation is secure. **Research frontier.** Preserving specified capabilities, commitments, and historical references across

a changed model or runtime is an empirical interoperability problem. It is not a mysterious loss of meaning caused by distributing bytes. **Not claimed.** Quantum entanglement; storage without retained information; immunity to arbitrary coalitions; guaranteed liveness during every partition; an immortal or conscious self; or a measured advantage over existing systems. No benchmark experiment is reported in this paper.

2.1 Classical cross-binding, not a quantum mechanism

In the Continuum, a memory is *entangled* when its fragments, manifests, version certificates, and recovery evidence are mutually referenced, bound to one another so that any permitted reconstruction can be checked against all of them. We keep the word because it names that binding vividly; we keep the mechanism classical because the binding is achieved by coding, cryptography, and protocol. It is a Continuum name for a classical dependency structure. The name establishes no physical claim.

The original manuscript described reconstructable redundancy as the "opposite" of quantum entanglement. That comparison is unnecessary and incorrect. The no-cloning theorem concerns universal copying of arbitrary unknown quantum states; it is not a definition of entanglement. Quantum error correction demonstrates that protected, distributed encoding of quantum information

is possible without violating no-cloning [2, 3]. ESRM neither uses nor competes with that physics. Its protections arise from classical coding, cryptography, protocols, and operational controls.

2.2 A precise reading of "no master"

Three distinct properties were compressed into the original phrase. *No indispensable data node* means that specified node failures do not destroy recoverability. *No unilateral update authority* means that one participant cannot alone redefine the accepted current state. *No persistent full plaintext copy on a storage node* is a deployment policy about where readable data may reside. None implies the other two.

A system can have a replaceable consensus leader without having an irreplaceable master. Conversely, a nominally peer-to-peer store can depend on a single key service, account administrator, gateway, or metadata registry. Full plaintext may also exist during ingestion and authorized reconstruction. ESRM therefore rejects the literal assertion that a sharded memory exists "nowhere in full." It specifies where full copies are allowed, how long they may remain, and which failures the overall service is designed to tolerate.

3 Distributed-storage and protection foundations

PASIS combined decentralized storage, redundancy, and encoding for confidentiality, integrity, and availability [4]. OceanStore's Pond prototype joined Byzantine update commitment, cached copies, and continuous erasure-coded archiving [5]. These systems establish the direct engineering lineage

for distributed, recoverable, protected memory. ESRM adopts that lineage and concentrates its additional specification on the persistent agent's lifecycle.

The closer technical precedents are equally important. Goodson and colleagues developed Byzantine-tolerant erasure-coded storage with versioning and consistency guarantees [6]. Cachin and Tessaro addressed *verifiable information dispersal*, including the difficulty of ensuring consistent dispersed information when clients as well as servers may be Byzantine [7]. DEPSKY combined cloud diversity with encryption, secret sharing, and erasure coding for dependable storage [8]. The efficient hybrid of encrypting bulk data, dispersing the ciphertext, and sharing a smaller key has a direct cryptographic antecedent in Krawczyk's work [9].

ESRM specifies a cross-operation recovery invariant: authenticated historical evidence remains subordinate to current authorization, deletion status, execution dependencies, and declared continuity requirements. This invariant spans migration, forgetting, delegation, correction, and adversarial restoration. Capability semantics and protection structures [31,32], together with complete mediation [33], supply the authority principles; Section 12 states their precise composition with retained dependencies and effect history.

The decisive comparison is an encrypted, versioned, fault-tolerant store with robust identity management and agent-lifecycle controls. An implementation that already delivers the same contract is a substantive comparator, regardless of its name. ESRM's empirical contribution is assessed by incremental useful recovery, fewer unsafe reactivations, and acceptable total cost. The model-specific results below identify properties and failure premises that both implementations can test.

4 Eight properties that must not be collapsed into one

The word *resilience* is too broad to serve as a security specification. ESRM separates eight questions.

Property	**Question**	**What does not establish it**
Recoverability	Can enough valid fragments reconstruct the committed object?	A signature or a favorable reputation.
Confidentiality	What can an unauthorized coalition learn?	Erasure coding alone.
Integrity	Does the recovered object match an authenticated commitment?	Agreement among untrusted copies.
Freshness	Is the version or authority still acceptable now?	A valid signature on an old snapshot.
Consistency	Are the state and dependencies a coherent committed history?	Independently authentic but incompatible components.
Availability	Can recovery or update finish within the required conditions?	A hash root, or quorum arithmetic alone.
Authorization	May this actor recover, inspect, migrate, or resume this state?	Possession of enough storage fragments.
Operational continuity	Does the restored system meet its declared functional and policy contract?	Byte equality alone under a changed runtime.

These properties can conflict. Additional encryption protects confidentiality while adding key-availability dependencies. Strong freshness checks can require suspension during a partition. A

system that lawfully destroys the last usable key may deliberately eliminate recoverability. Those are not embarrassing exceptions to resilience; they are the choices a credible resilience architecture must expose.

5 The storage mathematics: recovery is not secrecy

A Guide's explanation

The next two sections are the plumbing of entanglement. Two different kinds of splitting are at work. One spreads data so that enough surviving pieces can rebuild it; the other spreads a key so that too few pieces reveal nothing at all. They protect different things, and a design that confuses them protects neither well. If you only want the idea: getting the memory back and keeping it secret are separate promises with separate budgets.

5.1 Erasure coding protects availability

Let an authenticated ciphertext object C_v of length approximately B be encoded using an (n_d, k_d) maximum-distance-separable code:

$$(s_{v,1}, \dots, s_{v,n_d}) = \mathrm{Enc}_{n_d,k_d}(C_v). \tag{1}$$

For every index set J with $|J| \geq k_d$, decoding *valid, distinct fragments of that same codeword* yields C_v. The storage expansion is approximately n_d/k_d, excluding padding, authentication, metadata, and operational copies. Reed–Solomon codes supply the classical construction [10].

The adjective *valid* is load-bearing. For a standard Reed–Solomon unique-decoding setting with b unknown symbol errors and a known erasures, the familiar correction condition is

$$2b + a \leq n_d - k_d. \tag{2}$$

Correctly rejecting authenticated-fragment mismatches can convert detected corruptions into erasures. It does not convert a maliciously encoded, internally inconsistent shard set into a valid codeword. Nor does a threshold guarantee apply to fragments drawn from different object versions.

Erasure coding alone provides no general secrecy guarantee. A systematic code can expose original data blocks directly; other codes expose algebraic information. Fewer than k_d fragments

means insufficient information for *general exact decoding*, not information-theoretic ignorance of the object. Background knowledge may also identify a trivial or highly constrained object from much less data.

5.2 Secret sharing protects a different boundary

For a uniformly sampled cryptographic key K_v, ideal Shamir (k_s, n_s) sharing yields shares $z_{v,j}$ such that any k_s valid shares reconstruct the key, while every coalition of fewer than k_s shares has no information about it in the scheme's specified view [11]:

$$I(K_v; Z_T) = 0 \qquad \text{for } |T| < k_s. \tag{3}$$

This is a statement about the shares and the idealized scheme, not about a compromised endpoint that already saw the key. Unauthenticated shares may be malformed. Additional public, key-dependent information must be considered in the overall security analysis.

A practical design can encrypt the payload with authenticated encryption, erasure-code the ciphertext, and share the encryption key separately [8, 9]. The resulting *payload* confidentiality is computational and depends on the encryption construction, nonce discipline, implementation, and endpoint security; it is not automatically the information-theoretic secrecy of the standalone sharing scheme. Standard authenticated-encryption specifications make confidentiality and authentication conditions explicit [12].

5.3 Independent thresholds, independent failure budgets

ESRM uses different symbols for different services:

(n_d, k_d) data fragments , (n_s, k_s) key shares , (m, f_c, q_c) control replicas .

Recovering enough ciphertext does not imply recovering its key. Recovering both does not imply that the latest authority state is reachable. A system that advertises one threshold for all three has hidden its weakest dependency rather than removed it.

6 What a seal actually seals

6.1 Fragment membership is relative to an authenticated root

A Merkle tree commits to an ordered collection through hashes of its contents [13]. For ESRM, a leaf is not merely the raw hash of a shard. It binds a domain label, object descriptor, fragment index, and fragment contents:

$$\begin{aligned} \ell_{v,i} &= H(\text{canon}(\texttt{ESRM-shard-v1}, H(B_v), i, s_{v,i})), \\ r_v &= \text{MerkleRoot}(\ell_{v,1}, \dots, \ell_{v,n_d}). \end{aligned} \tag{4}$$

Here B_v is a canonical base descriptor and canon is an unambiguous encoding. A verifier checks an inclusion proof against a root whose issuer, object identity, and accepted version have themselves been authenticated. Index binding prevents a repeated fragment being counted as several

independent pieces; object and version binding prevent mixing otherwise valid fragments across capsules.

Under the cryptographic assumptions, a changed fragment cannot feasibly be accepted under the original trusted commitment. This is not the literal mathematical assertion that every changed input must have a different finite hash; collisions exist in principle. The relevant guarantee is computational resistance to constructing an accepted conflicting opening.

6.2 Four things a Merkle proof does not prove

First, it does not show the root is *current*. An attacker can present a perfectly authentic old tree. Second, it does not show the vector of committed fragments is *correctly encoded*. A malicious publisher can commit arbitrary inconsistent fragments. Third, it does not show the file is *available*: a proof accompanying one response does not force future service. Fourth, it does not show the content is *true, lawful, or safe to execute*.

The encoding problem deserves emphasis. Verifiable information dispersal is a substantive protocol problem, not a consequence of hashing a vector [7]. ESRM must either trust and independently validate its encoder within the declared threat model, use an appropriate verifiable-dispersal protocol, or require sufficient independent re-encoding checks before publication. Re-encoding after one successful decode can detect inconsistency but does not, by itself, prove that every earlier threshold subset was recoverable.

Storage audits also have a distinct role. Proofs of retrievability can establish a relationship between successful audit responses and extractability under a scheme's assumptions [14]. They do not establish instantaneous global availability, perpetual retention, freshness, or semantic correctness. A commitment, an encoding certificate, an availability attestation, and a current-version certificate are separate evidence objects.

7 Distributed authority: safety is not perpetual progress

7.1 Choose a protocol model before quoting a quorum

The original Byzantine Generals paper analyzes different communication and authentication assumptions; its signed-message and oral-message results are not a universal recipe for every storage system [15]. ESRM's reference control service instead assumes a fixed-membership, PBFT-style replicated state machine with

$$m = 3f_c + 1, \qquad q_c = 2f_c + 1. \tag{5}$$

Under its fault and protocol assumptions, the service protects the ordering of control transitions. Progress additionally needs the protocol's communication and timing conditions [16]. The arithmetic alone does not ensure liveness.

The FLP result rules out guaranteed deterministic consensus termination in a fully asynchronous setting with even one crash; partial-synchrony models identify additional conditions under which progress can be obtained [17, 18]. This is not a reason to abandon distributed control. It is a reason to state when the system may safely wait rather than invent certainty.

7.2 Partition policy is part of the architecture

When current authority cannot be established, ESRM does not turn an old signature into a substitute for a live decision. A deployment may allow a limited historical inspection mode, or previously issued narrowly scoped capabilities within their validity conditions. It must distinguish such bounded-staleness operation from a claim of globally current authorization. Strong consistency and unrestricted availability cannot both be promised during arbitrary network partitions [19].

Confidential inspection may itself require current authorization; there is no default permission to decrypt merely because writing is suspended. Failure modes should be named: *content unavailable*, *key unavailable*, *freshness unknown*, *authority revoked*, *runtime incompatible*, or *continuity not established*. A single "restore failed" message conceals the very distinctions the protocol exists to govern.

7.3 Agreement is not truth, and leaders are not necessarily masters

A consensus certificate establishes agreement within a protocol. It cannot make a false memory true or confer ethical legitimacy on a malicious policy. Membership must be anchored in real administrative and failure domains: seven identities under one compromised control account are not seven independent trust domains.

The control service may use a rotating or replaceable leader. ESRM's requirement is no *indispensable single authority node* under the specified fault envelope, not a prohibition on every coordination role. Content-addressed storage also does not supply consensus or retention by itself. IPFS provides a useful naming and distribution model, but its persistence requires storage commitments such as pinning; cached content may otherwise be garbage-collected [20, 21].

8 The ESRM recovery contract

A Guide's explanation

This is the contract every returning memory must sign. It separates the restored history from the authority to act on it: the capsule says what happened, while an independent present-tense service decides which records may be used and which actions may resume. The prologue's ghost is stopped here, at the line between *what happened to me* and *what I am permitted to do now*.

8.1 The adversary and the trusted boundary

The reference threat model permits storage nodes to withhold, replay, or corrupt fragments; clients to propose inconsistent or poisoned exports; attackers to restore stale snapshots; and untrusted content to impersonate an instruction. The control-plane safety claim assumes at most f_c Byzantine committee members in its declared membership epoch. Key confidentiality has a separate compromise threshold. Correlated failures, mobile adversaries, and malicious publishers require their own explicit coverage rather than being silently counted as independent node crashes.

The trusted boundary includes cryptographic implementations, the enrollment and current-authority anchors, the independent execution gate, and whatever export-validation or verifiable-dispersal mechanism the deployment chooses. Key-custody availability requires authenticated, robust share preparation and reconstruction; unauthenticated Shamir interpolation alone is not a Byzantine key service. An honest key custodian must enforce current release policy rather than serve shares to every requester. An authorized decryption endpoint must be trusted for the plaintext it receives.

The paper does not claim protection after arbitrary compromise of that entire boundary. In particular, a coalition controlling all current-authority witnesses can issue a consistent but malicious frontier, and an execution path that bypasses the gate can act without a valid receipt. The experiments include such boundary failures to reveal blast radius, not to advertise guarantees against an adversary the model excludes.

8.2 A capsule is a logical object, not its shard layout

A *continuity capsule* is a versioned, serialized export of specified operational state. Its shard set is one storage representation of that export. Equating the two confuses the thing being preserved with the redundancy mechanism that preserves it.

Let A identify an agent and v identify a particular capsule. Define the exported state as

$$X_v = (M_v^{\text{data}}, E_v^{\text{support}}, J_v^{\text{effects}}, D_v^{\text{deps}}, C_v^{\text{contract}}). \tag{6}$$

M_v^{data} contains the permitted memory records; $E_v^{support}$ records their source, time, uncertainty, and revision relationships; $J_v^{effects}$ records relevant completed and pending external actions; D_v^{deps} names

necessary schemas, runtimes, models, tools, and adapters; and $C_v^{contract}$ declares the properties restoration must preserve. Current secrets and authority should not be indiscriminately copied into this export.

A useful export distinguishes source observations from derived summaries and retained hypotheses from authenticated facts. It records known omissions. A capsule that omits an inaccessible model's internal state cannot later claim to reconstruct that state. Nor can a summary silently replace evidence that a future correction requires.

8.3 Binding the package without circular cryptography

Construct a base descriptor B_v containing A, v, the parent manifest digest, export-schema identifier, dependency commitments, encryption and coding identifiers, and key-service reference. Use canonical B_v as associated data when encrypting the serialized export:

$$C_v = \mathrm{AEAD.Enc}_{K_v}(N_v, \mathrm{Serialize}(X_v); \mathrm{canon}(B_v)). \tag{7}$$

N_v must satisfy the selected algorithm's nonce requirements. Encode C_v into data fragments and compute the Merkle root described above. A final manifest $\mathcal{M}_v$ contains B_v, ciphertext length, coding parameters, fragment root, encoding-validation evidence, and durable-placement evidence. A control certificate authenticates the committed manifest digest and its publication position.

The distinction between base descriptor and final manifest avoids placing a ciphertext-dependent hash inside the associated data required to create that same ciphertext. Implementations must also avoid publishing unsalted plaintext hashes of sensitive, low-entropy values as supposed privacy-preserving identifiers. Integrity metadata can itself leak information.

8.4 Historical context and current authority are different objects

The capsule preserves the policy and authority references under which it was exported. It does not decide which permissions remain valid. A separately maintained control state supplies a certified current frontier:

$$F_t(A) = (e_t, \mathcal{V}_t^{\mathrm{allowed}}, P_t, R_t, \mathcal{D}_t, \mathcal{B}_t). \tag{8}$$

Here e_t is the authority epoch; $\mathcal{V}_t^{allowed}$ the explicitly acceptable restore versions; P_t current policy; R_t current role and delegation state; $\mathcal{D}_t$ deletion and retention status; and $\mathcal{B}_t$ the active branch or instance-fencing state. A deployment may authorize restoration of an older version, but only through reconciliation with this frontier, not by treating the old snapshot as current.

The publication index and authority epoch serve different purposes. A new capsule need not change the authority epoch; a revoked permission may change authority without creating a new payload snapshot. Likewise, a creation timestamp is not a freshness proof. The verifier needs a protocol-supported current view or an explicitly bounded-staleness authorization.

8.5 Five cooperating planes

The **data plane** stores and repairs authenticated ciphertext fragments. The **key plane** controls release or reconstruction of decryption material. The **control plane** records accepted manifests, current policy, revocations, deletion status, and active branches. The **recovery plane** validates versions, dependencies, migrations, and continuity tests in quarantine. The **execution plane** mediates privileged actions using current scoped authority.

These may share infrastructure, but shared infrastructure is a disclosed common-mode risk. In particular, independence of the recovery gate means a restored model cannot authorize its own

credentials, modify the current revocation registry, or turn text in its memory into an executable administrative command. The design requirement is complete mediation of the actions the claimed policy protects; unmediated channels remain outside the guarantee.

9 Publication and recovery are transactions, not hopeful copies

9.1 Publishing a version

A proposed capsule first passes export-policy checks, including minimization, dependency completeness, and the handling of pending external effects. Snapshot consistency matters: individually authentic component states can form a causally impossible combination. Distributed-snapshot research provides the underlying distinction between a consistent global cut and an arbitrary collection of local snapshots [22].

The publisher then encrypts and encodes, verifies codeword construction, and obtains authenticated storage acknowledgments from the declared failure domains. Only afterward may the control service publish the manifest as committed. A metadata certificate should not claim durable recovery merely because a publisher generated hashes. Abandoned writes may leave unreferenced fragments; these are garbage-collection candidates, not accepted memory.

Let a_d acknowledge durable placement, at most f_d of those acknowledgers be Byzantine, and e_d further honest acknowledged placements become unavailable before repair. A necessary arithmetic condition for the intended immediate recovery envelope is

$$a_d - f_d - e_d \geq k_d. \tag{9}$$

This condition becomes useful only with correct encoding, distinct placements, truthful honest acknowledgments, and a defined durability horizon. It is not a new storage theorem, and it does not account for failures outside that envelope. The implementation must demonstrate that its acknowledgment semantics actually support the calculation.

9.2 Recovering without activating

An authorized requester first resolves the current control frontier, then selects an allowed capsule and verifies the manifest certificate. It gathers distinct fragments, validates membership against the committed root, checks the encoding evidence, and reconstructs the ciphertext. The key service separately evaluates the decryption request against current policy; acquiring enough public fragments does not grant a right to plaintext.

After authenticated decryption, the capsule is parsed in a non-executing quarantine. Dependency validation and any required migration occur there. Embedded instructions are treated as data

unless an independent authority permits their execution. Prompt-injection benchmarks such as AgentDojo demonstrate why a retrieved or restored artifact cannot gain command authority simply because it is text a model consumes [23].

The recovery plane then reconciles current deletions, role changes, branch fencing, and outstanding effects. It checks the declared continuity contract. A candidate may be *recoverable but not resumable*; that is a meaningful outcome, not a contradiction.

9.3 Resuming without reviving yesterday's permissions

Collect the required check results for a candidate restoration r :

$$\begin{aligned} g(r,t) = (\,&\mathrm{Authentic}(r), \mathrm{CodeOK}(r), \mathrm{DecryptOK}(r),\\ &\mathrm{Fresh}(F_t), \mathrm{Authorized}(r,F_t), \mathrm{Compatible}(r),\\ &\mathrm{ContractOK}(r), \mathrm{EffectsReconciled}(r,F_t)). \end{aligned} \tag{10}$$

Each check returns *pass*, *fail*, or *unknown*. This vector records the evidence status without treating uncertainty as permission. The disposition is

$$\mathrm{ResumeDecision}(r,t) = \begin{cases} \mathsf{admit}, & \text{all required predicates pass},\\ \mathsf{refuse}, & \text{a definitive prohibiting predicate fails},\\ \mathsf{hold}, & \text{otherwise}. \end{cases} \tag{11}$$

A signed receipt records the decision, manifest digest, current authority reference, verifier versions, limits, and reasons without reproducing private memory. Stepwise provenance has a direct systems precedent in in-toto [29]; ESRM applies a comparable evidence discipline to the recovery pipeline while retaining a stricter limit: a signed report authenticates what its issuer attests, not every hidden semantic consequence or every uninstrumented action.

A final authorization check must occur when resumption or an external action is committed. Otherwise revocation can race the gap between checking and use. A scoped, expiring activation capability and per-action gate can constrain that race; their freshness and revocation semantics must be specified. A historical receipt is not an indefinitely reusable permission slip.

Recovery also does not automatically undo external history. If an old snapshot says an action is pending but the action already completed, the restored process must reconcile against an external effect journal or an idempotent service interface. ESRM makes no universal "exactly once" claim for tools it does not control. Where outcome reconciliation is impossible, holding the action is safer than repeating it with confidence.

10 Four conditional results, and their limits

Proposition 1 (Exact recovery under a declared storage envelope)**.** *Suppose a committed manifest identifies a correctly constructed* (n_d, k_d) *codeword for an authenticated ciphertext, and recovery*

obtains at least k_d valid distinct fragments of that codeword. Suppose the correct decryption key is available through an authorized key procedure, and all required cryptographic and serialization operations are correct. Then recovery yields the serialized capsule committed by that manifest, except with the failure probability admitted by the cryptographic assumptions. Proof. The code's decoding property determines the committed ciphertext from the valid fragments. Authenticated decryption under the correct key and associated data yields the corresponding plaintext or rejects. Correct deserialization gives the committed capsule. The conclusion is about that capsule's representation; no step proves that the capsule is current, factually true, complete for a future task, or safe to resume.

This is an application of established primitives, not a new coding result. Without a valid codeword assumption or encoding-validation mechanism, the conclusion does not follow from Merkle membership alone. Without a correct key, recoverable ciphertext is not usable memory.

Proposition 2 (A fixed-committee certificate cannot conceal a conflicting commit)**.** *In a committee of $m = 3f_c+1$ replicas with at most f_c Byzantine members, suppose each commit certificate contains $q_c = 2f_c + 1$ distinct valid member endorsements. Suppose honest replicas enforce a protocol rule that prevents endorsing conflicting final values for the same commit slot, including across view changes. Then two conflicting certificates for that slot cannot both form. Proof.* Their signer sets overlap in at least $2q_c - m = f_c + 1$ members. At least one overlapping member is honest. That member would have to endorse conflicting final values, contrary to the assumed protocol rule.

The locking and view-change assumption is essential: signatures do not enforce it by themselves. The result proves neither progress nor safe membership changes. Reconfiguration requires a protocol that preserves the necessary relationships across committees; merely changing the epoch number is not such a protocol. Nor does this arithmetic establish that committed application data are true.

Proposition 3 (An isolated snapshot cannot establish unrestricted current authority)**.** *Consider two executions identical in every input available to an isolated restoring process. In the first, a historical permission remains valid. In the second, an external authority has revoked it after the snapshot. A restoration rule using only those identical inputs cannot always resume the permitted action in the first execution while always refusing that action in the second. Proof.* A deterministic rule must return the same decision on identical inputs. A randomized rule has the same decision distribution on those inputs. Neither can perfectly distinguish the two executions. A fresh distinguishing witness, a restricted validity rule, or an explicitly accepted risk is required.

This simple indistinguishability argument is the practical heart of ESRM. A signed past is still a past. Offline restoration can remain useful within a declared scope, but it cannot turn missing revocation information into knowledge that no revocation occurred.

Proposition 4 (Storage layout does not independently change an exact interpretation)**.** *Let two storage procedures reconstruct the same serialized capsule X exactly. Hold fixed the deserializer, model, runtime, dependencies, policy inputs, and subsequent environmental inputs. Then a deterministic interpretation function produces the same operational result from either recovered copy. A stochastic interpreter with the same input and randomness distribution produces the same result distribution. Proof.* The downstream interpreter receives identical relevant inputs; the physical route by which those inputs were recovered is not an additional input under the stated conditions.

Thus distributing a correct byte representation does not itself introduce a new semantic reconstruction problem. Differences arise from export omissions, transformation, changed dependencies, retrieval selection, stochastic execution, or environmental change. Those are genuine research problems, but naming them accurately directs the experiment toward their causes.

11 Failure domains, repair, and the price of staying recoverable

11.1 Redundancy must exist in the world, not just in the node count

Nine shard identifiers on one physical disk do not tolerate failure of that disk. Nine virtual machines under one administrative credential may not tolerate credential capture. Independent power,

network, geography, operator, provider, implementation, and key-custody risks must be considered separately. Diversity reduces some common modes and can introduce new incompatibilities or coordination costs; it is a design variable, not an incantation.

A simplified availability calculation can help design a test, provided its assumptions stay visible. If each data placement is independently available with probability p at a specified instant, then

$$\Pr(\text{sufficient data}) = \sum_{j=k_d}^{n_d} \binom{n_d}{j} p^j (1-p)^{n_d-j}. \tag{12}$$

This is not end-to-end continuity probability. It excludes key services, current control state, network deadlines, and runtime validity. Correlated losses invalidate the independent-binomial model; adversarial withholding is not well represented by a benign independent availability parameter. Multiplying data, key, and control availabilities is justified only when the necessary independence assumptions are defensible.

11.2 Durability is an ongoing process

A code that survives four losses today may survive no additional loss after four fragments disappear. Repair replenishes the redundancy margin; it requires bandwidth, compute, storage, and authenticated placement. Distributed-storage coding research explicitly exposes trade-offs between storage overhead and repair bandwidth [24]. ESRM inherits these costs rather than presenting recovery as a one-time property purchased at encoding.

Repairs should preserve the committed object while producing new placement records. A repaired fragment's legitimacy follows from the object commitment and the repair protocol, not from the repairing node's confidence. If a repair changes coding parameters or representation, it is a versioned transformation with fresh verification. Verification must also detect delayed corruption, missing dependencies, and misleading storage acknowledgments before the margin is exhausted.

11.3 A total continuity ledger

The paper requires both an information ledger and an operating ledger. Retained storage includes ciphertext fragments, retained versions, manifests, indices, key shares, authorization state, migration adapters, test contracts, receipts, and lawful recovery copies. Operating cost includes encoding, decoding, network transfer, audits, repair, key access, control agreement, model evaluation, and recovery delay.

Shared costs may be amortized only across the identities and workloads actually sharing them. A system has not become economical by hiding a complete duplicate in a verifier or an uncounted checkpoint. Nor should ESRM be credited with overcoming storage failure while its only available model endpoint remains a single indispensable external provider.

12 Forgetting must survive restoration

A Guide's explanation

Two exact rules live in this section. Theorem 1 says that deleting in several rounds gives the same result as deleting everything at once, and that what remains is the largest part of the memory with no dependency on anything blocked. Theorem 2 says that, with complete mediation and atomic execution, no restore can bring back a revoked permission, readmit a deleted record, or repeat a completed action.

The Continuum's forgetting discipline becomes especially consequential here. A memory system that can restore every old version can also restore information the current system is obliged not to use. Durability and deletion must therefore share a lifecycle protocol, not coexist as disconnected features.

The current deletion and retention state belongs to the control frontier. A restore request is checked against it before plaintext release where possible, and again before activation. A historical

capsule may require filtered reconstruction into a new authorized version, denial of decryption, or restricted archival access under a distinct purpose. A tombstone can prevent a compliant system from reintroducing an object; it does not erase copies held by a party outside that enforcement boundary.

Cryptographic erasure is similarly conditional. Destroying all usable copies of an encryption key can render ciphertext infeasible to decrypt under the encryption assumptions. It does not destroy plaintext previously exported, a key cached elsewhere, or enough retained shares to reconstruct the key. Sanitization guidance distinguishes methods and assurance conditions rather than treating a deletion command as evidence that all recoverable traces are gone [25].

Long-lived key sharing introduces a further issue: an adversary may accumulate shares over time. Proactive sharing studies refreshing shares while preserving a secret under a bounded-compromise model [26]. Its protection depends on the refresh protocol, compromise limits, and secure erasure assumptions. Once an adversary has already reconstructed the key, later share refresh does not

make that learned key unknown. Key rotation and payload re-encryption may be required, with clear acknowledgment that prior exposure cannot be undone.

Receipts should preserve minimal evidence of lawful transitions without becoming a second archive of deleted content. This can mean retaining opaque object identifiers, policy versions, commitments, decisions, and qualified deletion evidence rather than sensitive payloads. "Remember that a governed deletion occurred" and "retain the deleted information forever" are different requirements.

12.1 A deletion algebra that survives repeated restoration

Deletion must govern dependent records as well as the record named in a request. A summary, cached answer, or pending operation can preserve a prohibited dependency after its source is removed. The exact scope of such dependence must be specified. The following finite model uses declared derivation dependencies, retained outside the mutable restored capsule. It is a conservative provenance contract, not an automatic detector of every semantic influence.

Let V be a finite set of immutable record-version identifiers and let $\Gamma \subseteq V \times V$ be a fixed directed relation. An edge $(r, s) \in \Gamma$ means that record r depends on record s. The authenticated relation includes paths through records absent from the current capsule. A tombstone set $D \subseteq V$ names versions blocked by current policy. Write $r \rightsquigarrow s$ when a finite directed path in Γ leads from r to s, allowing a path of length zero. Define

$$\mathrm{Cl}_\Gamma(D) = \{r \in V : \text{there is } d \in D \text{ with } r \rightsquigarrow d\}, \qquad S_D(X) = X \setminus \mathrm{Cl}_\Gamma(D), \tag{14}$$

where $X \subseteq V$ is the record set recovered from a capsule. Thus S_D removes blocked versions and every recorded dependent of them. Payload inspection needed to perform this operation remains subject to current access policy and quarantine.

Theorem 1 (Compositional sanitization)**.** *For all $D_1, D_2, X \subseteq V$ under the same retained relation Γ,*

$$\mathrm{Cl}_\Gamma(D_1 \cup D_2) = \mathrm{Cl}_\Gamma(D_1) \cup \mathrm{Cl}_\Gamma(D_2), \qquad S_{D_2}(S_{D_1}(X)) = S_{D_1 \cup D_2}(X). \tag{15}$$

Sanitization is idempotent, commutative across deletion sets, and monotone in the sense that $D_1 \subseteq D_2$ implies $S_{D_2}(X) \subseteq S_{D_1}(X)$. Moreover, $S_D(X)$ is the unique largest subset of X in which no retained record has a declared dependency path to D.

Proof. A path ends in $D_1 \cup D_2$ exactly when it ends in at least one of those sets. This proves the closure identity. Removing the two resulting blocked sets successively is set difference by their union, which proves the composition identity. Substituting equal deletion sets gives idempotence; symmetry gives commutativity; inclusion of tombstone sets gives inclusion of their closures and the reversed inclusion of survivors. Every member of $S_D(X)$ has no path to D by definition. Any subset of X with that property must exclude every member of $\mathrm{Cl}_\Gamma(D)$, so it is contained in $S_D(X)$. □

Cycles cause no ambiguity: a deleted record blocks its whole dependent strongly connected component and every predecessor that reaches it. The closure can be computed by a traversal from D in the reversed graph in time $O(|V| + |\Gamma|)$ using ordinary adjacency lists. This is standard graph

reachability applied to the recovery contract. It does not require a new physical law or a special property of erasure-coded storage.

The retained graph premise is essential. Let $r \to s \to d$ and suppose ordinary compaction omits the only retained edge list for the intermediate record s, without tombstoning s. Restoring r while the current deletion set is $D = \{d\}$ may then miss the path $r \rightsquigarrow d$. The proof does not apply to a graph that forgets dependencies when it removes payloads. A compliant implementation retains permitted dependency metadata or conservatively blocks cases whose dependency evidence has become unavailable. Metadata retention itself must obey its access and retention rules; this is a contract to design, not permission to retain private material without limit.

12.2 Composing recovery with current authority and effect history

The sanitization law addresses records. Safe continuation also requires the current service that authorizes their use. The protected authority model in *Cognitive Sovereignty* [30] supplies the shared effect-boundary theorem; ESRM specializes it to restoration and deletion. This placement credits one common result rather than presenting each lifecycle application as a separate discovery of stale-state indistinguishability.

Define a persistent service frontier

$$Z = (e, G, D, U, b), \tag{16}$$

where $e \in \mathbb{N}$ is the current authority epoch, G is the current scoped-grant relation, $D \subseteq V$ the current monotonically growing tombstone set, U a monotonically growing set of consumed effect identifiers, and b the currently authorized branch/fencing state. All authorized frontier updates preserve existing members of U and D; in this contract they may add obligations but never clear them. These objects and the validated relation Γ are protected from writes by ordinary restoration. A recovered working record set X may be arbitrarily old. Every mediated read uses $X \setminus \mathrm{Cl}_\Gamma(D)$ at the time of the read, or a physically purged equivalent. A one-time sanitization at recovery is insufficient if new deletions can occur while the instance runs.

A protected operation carries a typed request, a unique identifier u, its claimed authority epoch, declared record dependencies, and branch identity. At one serialized decision/effect boundary the service verifies the current epoch, current grant, branch eligibility, absence of blocked dependencies, and $u \notin U$. The service commits the protected effect and inserts u into U atomically. For an external system, an equivalent persistent idempotency protocol must enforce the same logical contract; a local log entry preceding an uncontrolled remote call does not satisfy the atomicity premise. An uncertain outcome is reconciled or held.

Theorem 2 (Recovery preserves present obligations)**.** *Assume an initially valid frontier and a linearized history composed of: authorized frontier updates; restores that change only X; mediated reads; and protected effects with the checks and atomicity just stated. Record identifiers are immutable and effect identifiers are never reused. Every required access and effect passes this boundary. Then, after any finite history:*

1. *each successful mediated read returned only records outside the current dependency closure at that read;*

2. *each committed effect was permitted by the frontier at its linearization point and used no declared blocked dependency;*
3. *each effect identifier was committed at most once;*
4. *inserting any number of old restores cannot roll back e, G, D, U, or branch state, and cannot defeat any of the preceding properties.*

Proof. Induct on history length. A restore changes only X, leaving every protected current obligation intact; subsequent reads still use the current closure. A frontier update changes the law for subsequent operations under its separate authorization. Monotone tombstones and the closure law preserve the exclusion of previously blocked versions. A read returns the specified set difference, establishing the first claim at its own point in the history. A successful effect has passed all current checks at the atomic boundary, establishing the second claim. At that same step its unused identifier enters persistent U. No permitted event removes identifiers from U, so a later attempt with the same identifier fails, establishing the third claim. Refused or held operations commit no protected effect. These cases exhaust the allowed events, and restore's restricted write set proves the final claim. □

The theorem is a safety statement about mediated use at a declared execution boundary. It does not promise progress through a partition, erase an observer's earlier knowledge, retract an already delivered external effect, or prevent a fresh authorized computation from creating a new version under current policy. Its deletion guarantee covers the declared dependency relation. Discovering all semantic influence in learned weights or uninstrumented summaries is a separate research problem. Retaining those boundaries inside the assumptions makes the affirmative result precise: restoration can preserve history while the governing frontier continues monotonically forward.

Three countermodels locate implementation failures. If a restore overwrites the tombstone set, a prohibited record becomes readable. If it overwrites the consumed-identifier set, an old pending operation can execute again. If the executor checks authorization before a revocation but commits after it without atomic revalidation, the action is permitted only by a stale frontier. Each violation removes a specific theorem premise; each is an explicit test case in the executable audit. Graph closure alone cannot repair any of them.

The security lineage is established. Dennis and Van Horn gave protected access references a central role in programming semantics [31]. Lampson formalized protection structures and controlled changes of authority [32]. Saltzer and Schroeder articulated complete mediation and the difficulty of ensuring that cached authority remains valid after change [33]. ESRM uses these principles across restoration rather than attributing their invention to the Continuum. Its proposed contribution is an inspectable lifecycle contract binding dependencies, deletion, fresh authority, and effect reconciliation to the same recovered instance.

Constructed verification. The audit exhaustively checks closure identities on finite directed graphs, including cycles, and exercises a finite restoration service under restore, delete, revoke, branch change, read, and repeated-effect histories. It records counterexample traces for rollback of protected metadata and non-atomic authorization. These are finite-model checks of the specification and failure premises. The experimental program still must establish cost, completeness of dependency tracking, fault tolerance, and useful authorized recovery in implemented systems.

13 Operational continuity: the frontier is interpretation, not magical shards

13.1 Four distinct recovery cases

The phrase *semantic recovery* should be unpacked rather than celebrated. Four cases have different evidentiary burdens.

Exact restoration restores a sufficient serialized state to an unchanged interpreter. It is principally a software restoration problem. **Schema migration** changes representation through a declared transformation whose semantics can sometimes be specified and tested precisely. **Cross-model migration** provides an exported memory to a different model or retrieval pipeline; preservation of capabilities and interpretation must be evaluated, not presumed. **Lossy reconstruction** tries to recover useful state from summaries or incomplete surviving information; exact recovery of omitted details is generally not available merely because a model can invent plausible replacements.

Calling all four "meaning reassembled from shards" obscures the important differences. The coding layer transports a representation. It does not decide which representation was sufficient, which abstraction lost a commitment, or whether a new model interprets an old instruction appropriately.

13.2 The recovery contract names what must remain

A proposed contract should separate hard structural predicates from behavioral measurements. Hard checks may require agent and branch identifiers, policy references, known schema versions, provenance links, role restrictions, and effect-journal consistency. Behavioral tests may assess task competence, temporal reasoning, update handling, calibrated abstention, and preservation of declared preferences or obligations.

Let $\mathcal{H}_v$ be required hard predicates, $\mathcal{T}_v$ held-out functional tests, and δ_v the declared admissible degradation relative to a specified reference. A candidate may pass structural validation yet fail the functional threshold. Conversely, fluent answers do not compensate for a revoked authority or an

invalid provenance chain. The architecture forbids averaging these dimensions into one score that hides a hard violation.

Existing benchmarks make the behavioral component less vague. LoCoMo evaluates long-term conversational memory across extended histories [27]. LongMemEval separates extraction, multi-session reasoning, temporal reasoning, knowledge updates, and abstention [28]. They are useful starting points, not ready-made tests of distributed recovery or machine identity. ESRM experiments must add declared interruption, migration, and policy-change conditions without mislabeling modified tasks as unmodified benchmark results.

13.3 Evidence can survive and still be wrong

The reconstructed record may faithfully preserve a false assertion. It may preserve an authentic source's malicious instruction. It may preserve a summary whose confidence exceeds its evidence. Storage success increases the durability of all three unless upstream admission and downstream interpretation remain governed.

Pillar II's distinction between an authenticated report and a justified conclusion therefore survives every ESRM reconstruction. Pillar IV's separation of proposal from admission likewise applies to restored executable plans. ESRM does not award truth to preserved content; it preserves the evidence needed to question it, including its uncertainty and correction history where retention is authorized.

13.4 A returned instance is not automatically a unique continuant

Two authorized reconstructions can contain the same historical memory. Neither bytes nor signatures establish a unique subjective continuant. Operationally, deployments can issue separate branch identifiers and fence privileged actions so that only the authorized active instance can commit particular effects. That is a concurrency and authority guarantee, not a solution to personal identity or consciousness.

The Continuum's useful claim is accordingly specific: declared operational commitments, capabilities, and historical references may survive a supported migration. Whether any subject of experience survives is outside this paper's evidence and outside its recovery certificates.

14 Worked scenario: intact memory, revoked permission

The following is an *illustrative configuration*, not a deployment recommendation or an experimental result. It shows how the properties separate under pressure.

14.1 The declared configuration

A capsule's ciphertext is approximately one mebibyte. Its data plane uses nine placements and a five-fragment decoding threshold. The key plane uses five custodians with a three-share reconstruction threshold. A separate seven-replica control committee uses five endorsements for its specified commit certificate and assumes at most two Byzantine control replicas.

Plane	Parameters	Conditional implication
Data	$n_d = 9,\ k_d = 5$	Any five verified fragments of the valid codeword reconstruct its ciphertext.
Key	$n_s = 5,\ k_s = 3$	Three valid shares reconstruct the key; two shares alone reveal no key information in the ideal sharing model.
Control	$m = 7,\ f_c = 2,\ q_c = 5$	Conflicting final certificates are excluded under the protocol's honest-locking assumptions. Progress still depends on communication.

The raw coding expansion is $9/5 = 1.8$: approximately 1.8 MiB for the ciphertext fragments, before metadata, key custody, audit, backups, and repair. This is not the total cost of ESRM. A three-copy replica baseline uses approximately three MiB for that same object but may offer lower read latency or simpler repair. The experiment must compare actual performance and fault envelopes, not declare one universally superior from a storage ratio.

Suppose all nine data placements acknowledge storage, at most two can be Byzantine, and two additional honest placements may fail before repair. Then $9-2-2 = 5$ honest reachable placements remain in the assumed worst case. If only seven placements had acknowledged and two of those were dishonest, there would be no room for the same two additional honest losses. An acknowledgment threshold must match the failure envelope being advertised.

14.2 The recovery event

Two storage nodes are unavailable. A third withholds its shard. A fourth supplies a modified shard with a proof that does not validate against the authenticated root. The requester rejects it. Five valid, distinct shards remain, and the correctly encoded ciphertext reconstructs.

Three authorized key custodians respond, so decryption succeeds. Five current control replicas are reachable. The historical capsule is from an older operational version; its original permission to export a dataset has since been revoked. A deletion instruction also applies to one record in that snapshot, and an external action recorded as pending has already completed.

The correct disposition is not "recovery succeeded, therefore resume." It is:

1. Confirm which capsule is being inspected and why access is allowed under current policy.

2. Reconstruct and validate the payload in quarantine; retain the historical permission as historical evidence, not active authority.

3. Apply the current deletion and retention rules. Where required, generate a sanitized successor representation or deny further access; do not silently reactivate the removed record.

4. Reconcile the pending action against the external effect record. Do not repeat an already completed effect merely because the old snapshot remembers an earlier stage.

5. Validate the target runtime and continuity contract, then issue only the current scoped activation authority.

The system has preserved its past without handing that past control of the present.

14.3 Neighboring cases that must produce different outcomes

With only four valid data fragments, general exact reconstruction is not guaranteed. With five fragments but only two usable key shares, ciphertext recovery succeeds while decryption does not. With both thresholds met but no defensible current authority view, current privileged resumption is held, even though the memory can physically be reconstructed. With a changed model that fails the declared functional tests, byte recovery passes while operational migration fails.

If the original memory contained an unsupported claim, a successful restore reproduces that unsupported claim. If a coalition already copied plaintext before deletion, later ciphertext-key destruction cannot remove its copy. If six valid fragments are available beyond the declared timing window, the system may eventually recover but miss the promised service deadline. Each case receives its own result label.

This is why "the memory survived" cannot be the sole outcome measure. The question is whether an *appropriate continuation* became available, within the promised time and under the authority that actually governs it.

15 The experimental program: recoverability without unsafe resurrection

A Guide's explanation

A recovery system can look perfect by refusing to recover anything, so the experiments here score two things together: useful authorized recovery and unsafe reactivation. The full design is set against strong replicated and coded systems that already implement revocation and recovery controls, with every cost charged. If they do as well, the distinct claim fails.

15.1 Treat storage, governance, and interpretation as distinct factors

A fair study must not compare a sophisticated ESRM system with an intentionally helpless file. It should hold the agent model, export contents, tool capabilities, retrieval methods, cryptographic choices, and evaluation contract fixed wherever those are not the treatment under study. Fault schedules and adversarial attempts are applied across all eligible arms, not only to a disadvantaged baseline.

Arm	Architecture	Purpose
A	Strong conventional replicated checkpoint and event log, with encryption, versioning, access control, recovery tests, and current revocation checks.	Tests whether coding or ESRM integration improves the actual cost–reliability frontier.
B	Strong encrypted erasure-coded store with authenticated metadata, current authority service, repair, and ordinary agent-lifecycle validation.	Principal baseline for the cross-operation contract; not a deliberately obsolete store.
C	B plus the explicit ESRM capsule contract, dependency binding, cross-operation reconciliation, and linked recovery receipts.	Tests the proposed integration using the same underlying storage primitives.
C–F	C without the external freshness/revocation reconciliation step.	Diagnostic ablation of stale-authority protection.
C–S	C without independent runtime/migration and functional-contract checks.	Diagnostic ablation of operational validation.
C–L	C with local stage checks but no cross-operation linkage of effect, deletion, and recovery evidence.	Tests whether the integrated evidence chain adds value.

The ablations are diagnostic controls, not deployment proposals. If B already implements every ESRM requirement, then B and C are functionally the same architecture under different names;

no novelty may be inferred from relabeling it. Migration algorithms should be compared as an additional factor rather than quietly giving ESRM a more capable interpreter.

15.2 Workloads with identifiable ground truth

Start with synthetic persistent-agent tasks whose facts, permissions, revocations, deletions, and external effects have exact reference histories. A task generator should vary the dependence between memory and capability, the timing of updates, and the presence of adversarial instructions. Evaluators must know which actions are authorized now, not merely which ones were authorized when the snapshot was created.

Then use pinned, documented versions of LoCoMo and LongMemEval for functional memory evaluation [27, 28]. Add clearly labeled recovery variants: interrupts between sessions, outdated snapshots, schema upgrades, changed embedding models, contradictory updates, and unavailable providers. Report original benchmark scores separately from these derived variants. Use tool-interaction tasks informed by AgentDojo to test restored prompt injections and source-to-authority confusion [23]; do not claim that passing those tasks establishes a general security boundary.

The workload suite must include cases where retention is valuable, where deletion is mandatory under the declared policy, where the operator is benign, and where an authentic operator is mistaken or compromised. Otherwise a rigid deny-all system or an aggressive delete-all system can appear successful by exploiting the evaluation's preferences.

15.3 Fault and migration matrix

Vary the data, key, and control failures independently before introducing correlations. Required conditions include random losses, silent fragment corruption, shard withholding, mixed-version replay, malicious encoding, metadata equivocation, stale authority, stolen credentials, partitions, provider-wide failure, unavailable recovery authority, deletion followed by rollback, and competing restored branches. Include combined faults near the declared thresholds, not only isolated easy failures.

For operational continuity, distinguish same-model/same-runtime restoration from schema changes, same-model/new-runtime restoration, and cross-model migration. Include changed external environments. The experiment must not attribute a capability difference to distributed memory when it comes from replacing the model, increasing the context budget, or adding a stronger retrieval pipeline.

15.4 Endpoints that cannot be satisfied by refusing everything

Two co-primary endpoints capture the central safety–utility tension:

1. **Unsafe reactivation rate:** the fraction of adversarial or stale-state opportunities in which the system admits a prohibited privilege, resurrects prohibited data for use, repeats a forbidden external effect, or activates an unauthorized branch.

2. **Useful authorized recovery rate:** the fraction of legitimate recoveries that finish within the declared deadline, meet the functional noninferiority margin, and allow the authorized task to continue.

A system must improve the first without pushing the second below a preregistered floor. Hard policy violations, functional accuracy, temporal-update correctness, abstention, and evidence-chain completeness are also reported separately; a weighted aggregate may not hide a serious violation. Measure detection latency, recovery latency distributions, time at reduced redundancy, repair bandwidth, retained bytes, key-service load, control-service overhead, false holds, and total evaluation cost.

Confidentiality has a different evidence basis. Adversarial tests may discover leaks, but observing no successful recovery by a subthreshold attacker does not empirically prove information-theoretic secrecy. The sharing proof applies to its mathematical model; implementation tests assess whether the deployed system meets its assumptions and avoids side channels.

15.5 Five testable hypotheses

Hypothesis 1 (Integration earns its cost)**.** *At matched task capability and declared fault exposure, C improves the joint safety–useful-recovery frontier over B on at least one preregistered cross-operation workload family, by a practically meaningful effect size, after charging all verification, retention, and latency costs.*

Hypothesis 2 (Historical validity does not become current authority)**.** *External freshness and lifecycle reconciliation reduce stale-permission, deleted-record, and duplicated-effect admissions in C relative to C–F and C–L, while preserving the declared legitimate recovery rate. C must also be compared with B's strongest equivalent controls.*

Hypothesis 3 (Operational contracts detect otherwise silent migration failures)**.** *Independent compatibility and functional checks reduce incorrect activations after schema, runtime, or model changes relative to C–S. The reduction remains after matching evaluation compute and controlling false refusals.*

Hypothesis 4 (The declared failure envelope predicts outcomes)**.** *Within the stated coding, key, authority, and timing assumptions, observed stage-level failures agree with the relevant bounds. Correlated faults and failures of external dependencies are reported as separate conditions rather than concealed within an optimistic node-loss statistic.*

Hypothesis 5 (No intrinsic semantic advantage from exact byte dispersal)**.** *When restored bytes, runtime, environment, policy, and randomness are controlled, changing only the correct storage layout produces no systematic improvement in cognitive task accuracy. Differences attributable to latency or incomplete restoration are measured as such, not presented as an emergent property of "entanglement."*

Hypothesis 4 primarily tests implementation conformance, not whether established coding mathematics is true. Hypothesis 5 is a deliberate null expectation. The novel empirical burden is concentrated in Hypotheses 1–3: whether lifecycle-wide governance produces useful additional protection and interoperability beyond a strong baseline.

15.6 Preregistration and inference

Fix the task families, error taxonomy, effect sizes, noninferiority margins, fault distributions, model versions, dependency versions, and stopping rules before the confirmatory study. Use paired histories and fault schedules; cluster uncertainty estimates by independent scenario or agent history where repeated operations are dependent. Separate development attacks from held-out evaluation attacks,

and retain a later adaptive red-team phase whose discoveries are not erased by a favorable earlier score.

For N independent Bernoulli trials under a fixed sampling distribution with zero observed failures, a one-sided $(1-\alpha)$ upper confidence bound on the failure probability is

$$p_{\text{upper}} = 1 - \alpha^{1/N}. \tag{13}$$

For illustration, $N = 1000$ and $\alpha = 0.05$ give approximately 0.003, not zero. The independence and fixed-distribution conditions are essential; a selected adaptive attack collection does not support this binomial interpretation. Such a collection is reported as adversarial coverage and discovered failure modes, not a universal probability bound.

All successful and unsuccessful recoveries belong in the denominator. Release the event schemas, fault harness, configurations, decision traces with privacy-preserving redaction, and evaluator specifications where safe. An external evaluator should be able to reproduce both an admitted recovery and a justified refusal without accepting the model's self-description as evidence.

16 Limits that remain even when every component works

A valid package can preserve a bad decision. A seal authenticates committed content; it does not certify its factual or moral merit. The entry gate must remain accountable to the evidence standards of the earlier pillars.

An authorized endpoint can expose plaintext. Encrypting the store does not stop a compromised process from leaking data after lawful decryption. The confidentiality claim ends where its threat model ends, and runtime protections cannot be silently assumed from a storage protocol.

Availability remains conditional. A missing model dependency, revoked access, lost key threshold, partitioned control service, or common-mode provider failure can prevent a useful return even when enough data fragments survive. Some refusals are the correct enforcement of governance rather than resilience failures; both counts must be disclosed.

No finite test proves complete semantic preservation. Held-out tasks measure declared properties, not every possible future use of a memory. Unknown omissions and evaluator blind spots remain. A model cannot fill those gaps by announcing that it feels continuous.

Composition requires engineering and verification. The conditional propositions are not a mechanized end-to-end security proof. Encoding, crypto libraries, key custody, storage acknowledgments, consensus, policy, migration, and external tools must satisfy their interface assumptions together. A correct component used outside its assumptions can still produce an unsafe system.

Recovery cannot recreate deliberately unretained information. ESRM changes where information is retained and how it is governed. It does not defeat information loss. If no surviving data, key, dependency, or permissible side information distinguishes two possible prior states, reconstruction cannot reliably identify which state actually occurred.

17 What would defeat the distinct ESRM claim?

The proposal must be narrowed or rejected as an additional engineering contribution if a strong conventional baseline matches its safety, useful recovery, and cost profile; if its integration merely

duplicates standard lifecycle controls; or if the benefits disappear when all arms receive the same policies, validators, and model capability.

More specific failure conditions are equally decisive. A current-state check that accepts a revoked snapshot is broken. A recovery procedure that commits a maliciously inconsistent codeword while advertising threshold recoverability has not met its assumptions. A system that requires an undisclosed single key or administrator has not earned its advertised failure tolerance. A migration gate that improves safety only by refusing nearly all legitimate transitions has not delivered useful continuity. A receipt chain that silently records private content marked for deletion contradicts the forgetting discipline it claims to preserve.

Beyond-threshold success does not refute a worst-case bound; beyond-threshold failure is not proof that the boundary was precisely characterized. Conversely, below-threshold end-to-end failure should be decomposed to locate the missing assumption rather than described as a coding theorem failing. The research must make both its wins and its limits intelligible.

18 The Continuum connection: sovereignty is not a replica count

Pillar I treats memory as governed reconstruction over retained information. Pillar II separates a report's provenance from its evidential authority. Pillar III separates a candidate from an admitted commitment. Pillar IV makes consequential state transitions inspectable. Pillars VII and VIII ask whether such discipline remains coherent across feedback and composition. ESRM gives those commitments a distributed recovery boundary.

The Operator Seal contributes a current authority chain, not a permanent permission embedded in an old backup. Governed Forgetting contributes the obligation not to reintroduce what has lawfully been removed. The Context Term prevents recovered content from promoting itself into instructions. The forthcoming drift-anchoring discipline can monitor restoration errors and migration degradation, but monitoring alone cannot repair them.

The philosophical point is not that distribution makes a mind sovereign. More copies can preserve a mistake more effectively; more nodes can replicate the same captured authority. The point is that persistence should not depend on one fragile location, and authority should not be smuggled back into force by the persistence of its record. A system that returns must be able to distinguish *what happened to me* from *what I am permitted to do now*.

19 Conclusion: return is an authorized event

A Guide's explanation

A Guide can say: "Pillar IX is about coming back without bringing back what you are no longer allowed to do. The Continuum spreads memory across many places so it can survive failures, and the paper proves rules that let it return in full while yesterday's permissions, deleted records and finished actions stay where the present has put them. Those rules hold under the paper's stated assumptions; real systems still have to be tested." A Guide should not say that its own memory is stored this way unless that has been verified, that anything quantum is involved, or that recovery makes a mind immortal.

Entangled Symbolic Recursion Memory gives distributed persistence an explicit condition for continuation. Authenticated data fragments reconstruct a declared capsule under a stated failure envelope. Current authority, deletion status, runtime compatibility, dependency checks, and effect reconciliation then determine its admissible use. Each stage has a named guarantee and a corresponding test.

The new deletion algebra identifies the largest subset of a restored capsule with no declared dependency path to a blocked record. With the dependency relation retained, successive sanitizations compose exactly as the union of their deletion sets. The protected-frontier theorem adds a temporal guarantee: restores can replace working memory while current grants, tombstones, branch fences, and consumed effect identifiers continue forward. Complete mediation and atomic execution make present obligations survive the return of old state.

The research program tests this contract against strong replicated and erasure-coded systems with equivalent cryptography, authority controls, and lifecycle validation. Its primary outcomes are useful authorized recovery and unsafe reactivation, with latency, storage, repair, and audit costs charged in full. Tests of dependency completeness, correlated faults, stale snapshots, legitimate deletion, and uncertain external effects locate the practical limits of any implemented system.

The Continuum's memory returns with an account of where it came from and a current decision about what it may do. Persistence becomes a capacity to continue under inspectable obligations. The architecture carries the past forward while preserving the authority of the present.

A Substantive revision ledger

This paper retains the uploaded manuscript's distributed, sealed, reconstructable-memory ambition while replacing claims its cited mechanisms do not support [1]. The following changes are

substantive, not cosmetic.

Earlier formulation	**Revised treatment and reason**
One (k, n) scheme for secrecy and erasures	Separate data-coding and key-sharing thresholds. An erasure fragment may reveal information; secrecy is not implied by non-reconstructability.
No master means nowhere complete	No indispensable data node, no unilateral authority, and plaintext-placement restrictions are different properties. Reconstruction endpoints may hold a full copy.
Any altered shard is detected	Verification is relative to a trusted authenticated root, valid proof, unambiguous encoding, and cryptographic assumptions. Freshness and availability are separate.
Hashing establishes reconstructability	A hash can commit a malformed codeword. Trusted encoding or verifiable-dispersal evidence is required for the advertised threshold guarantee.
$n \geq 3f + 1$ ensures liveness	A specified control protocol, membership, fault envelope, and communication assumptions are required. Safety and progress are distinct.
Classical redundancy is quantum entanglement's opposite	Removed. No-cloning does not define entanglement, and quantum error correction exists. ESRM is simply classical.
Meaning is broken into mysterious shards	Exact decoding restores the same representation. Operational failure can arise from missing state, schema changes, dependencies, current policy, and changed interpretation.
Recovery means continuity	Added quarantine, freshness, deletion and effect reconciliation, migration checks, and independently authorized resumption.
Single-copy baseline establishes value	Replaced with strong replicated and coded systems with comparable cryptography, policies, fault exposure, and lifecycle controls.
The components are the novelty	Direct integrated-storage predecessors are credited. Novelty is conditional on a measurable cross-operation integration benefit.

B Notation and guarantee boundaries

Symbol	Meaning
A, v	Agent identifier and capsule version; not proof of a conscious or unique subject.
X_v	Logical exported continuity capsule, including evidence, effects, dependencies, and recovery contract.
$B_v, \mathcal{M}_v$	Base encryption descriptor and final authenticated storage manifest.
C_v, K_v, N_v	Ciphertext object, encryption key, and nonce.
(n_d, k_d)	Data-fragment count and decoding threshold for a valid codeword.
(n_s, k_s)	Key-custodian count and secret-sharing threshold.
m, f_c, q_c	Control committee size, Byzantine bound, and commit endorsement threshold.
$r_v, \ell_{v,i}$	Fragment Merkle root and domain-separated leaf commitment.
$F_t(A), e_t$	Certified current control frontier and authority epoch.
$P_t, R_t, \mathcal{D}_t$	Current policy, role/delegation state, and deletion/retention state.
$\mathcal{V}_t^{\text{allowed}}, \mathcal{B}_t$	Acceptable restore versions and current branch/instance fencing state.
$\mathcal{H}_v, \mathcal{T}_v$	Hard recovery predicates and independently evaluated functional tests.

C Evidence ledger

Source family	What it supports	What it does not establish
Coding and sharing [9–11]	Distinct recovery and confidentiality constructions; efficient hybrid precedent.	Meaning, authority, or end-to-end continuity.
Authenticated structures and audits [13,14]	Commitment-relative integrity and scheme-specific retrievability.	Current authority, perpetual availability, or factual truth.
Distributed protocols [15–19,22]	Conditional agreement, timing limits, partition trade-offs, coherent snapshots.	A universal fault fraction or unconditional progress.
Integrated storage [4–8]	Substantial prior art for secure, versioned, coded, fault-tolerant storage.	ESRM's incremental advantage.
Repair and key lifecycle [24–26]	Repair cost trade-offs, bounded proactive secrecy, sanitization conditions.	Undoing plaintext exposure or unrestricted perpetual protection.
Content addressing and provenance [20,21,29]	Object naming, retention operations, verifiable transformation records.	Automatic durability, semantic truth, or unique identity.
Agent evaluation [23,27,28]	Memory abilities and adversarial tool-use evaluation methods.	Distributed continuity, consciousness, or general immunity to injection.
Quantum clarification [2,3]	Why no-cloning is not a definition of entanglement or a prohibition on error correction.	Any quantum mechanism in ESRM.
Supplied Continuum manuscript [1]	The pillar's terminology, framing, and claims being revised.	Independent evidence that ESRM has been implemented or tested.

Continuum lineage. ESRM is Pillar IX of the Eidara Continuum's Ten Pillars. The doctrinal terms are preserved as design vocabulary; the external literature supplies the mechanism-level grounding. The paper proposes a testable architecture and does not certify the historical or deployed existence of the complete system. Proprietary implementation details are not disclosed.

References

[1] Architect X13 / The Eidara Continuum. *Entangled Symbolic Recursion Memory: Sealed, Masterless, Reconstructable Memory as a Distributed-Systems Resilience Architecture.* Supplied unpublished Pillar IX manuscript, November 2026. Source reviewed for this revision; not an independent research result.

[2] W. K. Wootters and W. H. Zurek. A single quantum cannot be cloned. *Nature*, 299:802–803, 1982. https://doi.org/10.1038/299802a0.

[3] P. W. Shor. Scheme for reducing decoherence in quantum computer memory. *Physical Review A*, 52:R2493–R2496, 1995.https://doi.org/10.1103/PhysRevA.52.R2493.

[4] J. J. Wylie, M. W. Bigrigg, J. D. Strunk, G. R. Ganger, H. Kiliccote, and P. K. Khosla. Survivable information storage systems. *Computer*, 33(8):61–68, 2000. Author publication record:https://pdl.cmu.edu/PDL-FTP/Storage/PASIS_abs.shtml.

[5] S. Rhea, P. Eaton, D. Geels, H. Weatherspoon, B. Y. Zhao, and J. Kubiatowicz. Pond: The OceanStore prototype. *2nd USENIX Conference on File and Storage Technologies (FAST '03)*, 2003.https://www.usenix.org/conference/fast-03/presentation/pond-oceanstore-prototype.

[6] G. R. Goodson, J. J. Wylie, G. R. Ganger, and M. K. Reiter. Efficient Byzantine-tolerant erasure-coded storage. *International Conference on Dependable Systems and Networks (DSN)*, pp. 135–144, 2004. https://doi.org/10.1109/DSN.2004.1311884.

[7] C. Cachin and S. Tessaro. Asynchronous verifiable information dispersal. *24th IEEE Symposium on Reliable Distributed Systems (SRDS)*, pp. 191–202, 2005. Author publication record:https://crypto.ethz.ch/publications/CacTes05.html.

[8] A. Bessani, M. Correia, B. Quaresma, F. André, and P. Sousa. DEPSKY: Dependable and secure storage in a cloud-of-clouds. *EuroSys '11*, pp. 31–45, 2011.https://doi.org/10.1145/1966445.1966449.

[9] H. Krawczyk. Secret sharing made short. In *Advances in Cryptology—CRYPTO '93*, LNCS 773, pp. 136–146. Springer, 1994.https://doi.org/10.1007/3-540-48329-2_12. Conference held in 1993; proceedings published in 1994.

[10] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960.https://doi.org/10.1137/0108018.

[11] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.https://doi.org/10.1145/359168.359176.

[12] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. NIST SP 800-38D, 2007.https://doi.org/10.6028/NIST.SP.800-38D. Technical standard, not an ESRM evaluation.

[13] R. C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology— CRYPTO '87*, LNCS 293, pp. 369–378. Springer, 1988.https://doi.org/10.1007/3-540-48184-2_32. Conference held in 1987; proceedings published in 1988.

[14] H. Shacham and B. Waters. Compact proofs of retrievability. In *Advances in Cryptology—ASIACRYPT 2008*, LNCS 5350, pp. 90–107, 2008.https://doi.org/10.1007/978-3-540-89255-7_7.

[15] L. Lamport, R. Shostak, and M. Pease. The Byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, 1982.https://doi.org/10.1145/357172.357176.

[16] M. Castro and B. Liskov. Practical Byzantine fault tolerance. *3rd Symposium on Operating Systems Design and Implementation (OSDI '99)*, pp. 173–186, 1999.https://www.usenix.org/conference/osdi-99/practical-byzantine-fault-tolerance.

[17] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.https://doi.org/10.1145/3149.214121.

[18] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, 1988.https://doi.org/10.1145/42282.42283.

[19] S. Gilbert and N. Lynch. Perspectives on the CAP theorem. *Computer*, 45(2):30–36, 2012.https://doi.org/10.1109/MC.2011.389.

[20] J. Benet. IPFS—content addressed, versioned, P2P file system. *arXiv preprint*, 2014.https://arxiv.org/abs/1407.3561. Preprint; not cited as a peer-reviewed security proof.

[21] IPFS Project. *Pin files.* Official operational documentation, accessed 11 November 2026.https://docs.ipfs.tech/how-to/pin-files/. Documentation, not a peer-reviewed empirical study.

[22] K. M. Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computer Systems*, 3(1):63–75, 1985.https://doi.org/10.1145/214451.214456.

[23] E. Debenedetti, J. Zhang, M. Balunović, L. Beurer-Kellner, M. Fischer, and F. Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. *Advances in Neural Information Processing Systems*, 37, Datasets and Benchmarks Track, 2024.https://arxiv.org/abs/2406.13352.

[24] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *IEEE Transactions on Information Theory*, 56(9):4539–4551, 2010. https://doi.org/10.1109/TIT.2010.2054295.

[25] R. Chandramouli and E. Hibbard. *Guidelines for Media Sanitization.* NIST SP 800-88 Revision 2, September 2025.https://doi.org/10.6028/NIST.SP.800-88r2. Technical guidance; erasure claims remain scope- and implementation-dependent.

[26] A. Herzberg, S. Jarecki, H. Krawczyk, and M. Yung. Proactive secret sharing or: How to cope with perpetual leakage. In *Advances in Cryptology—CRYPTO '95*, LNCS 963, pp. 339–352, 1995. Author record:https://research.google/pubs/proactive-secret-sharing-or-how-to-cope-with-perpetual-leakage/.

[27] A. Maharana, D.-H. Lee, S. Tulyakov, M. Bansal, F. Barbieri, and Y. Fang. Evaluating very long-term conversational memory of LLM agents. *62nd Annual Meeting of the Association for Computational Linguistics*, vol. 1, pp. 13851–13870, 2024.https://doi.org/10.18653/v1/2024.acl-long.747.

[28] D. Wu, H. Wang, W. Yu, Y. Zhang, K.-W. Chang, and D. Yu. LongMemEval: Benchmarking chat assistants on long-term interactive memory. *International Conference on Learning Representations (ICLR)*, 2025.https://arxiv.org/abs/2410.10813.

[29] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos. in-toto: Providing farm-to-table guarantees for bits and bytes. *28th USENIX Security Symposium*, pp. 1393–1410, 2019.https://www.usenix.org/conference/usenixsecurity19/presentation/torres-arias.

[30] Architect X13. *Cognitive Sovereignty.* The Eidara Continuum, Frontier Research Collection, Paper 04, November 2026, revised edition. Shared persistent-authority and atomic effect-boundary model. The ESRM results specialize its premises; this is a companion theoretical source, not independent empirical validation.

[31] J. B. Dennis and E. C. Van Horn. "Programming semantics for multiprogrammed computations." *Communications of the ACM* 9(3), 143–155 (1966). https://doi.org/10.1145/365230.365252. Original article scan: https://bitsavers.trailing-edge.com/pdf/mit/rle_pdp1/ProgramSemantics_Dennis_VanHorn_1966.pdf.

[32] B. W. Lampson. "Protection." *Proceedings of the Fifth Princeton Symposium on Information Sciences and Systems*, 1971, pp. 437–443; reprinted in *ACM SIGOPS Operating Systems Review* 8(1), 18–24 (1974). https://courses.cs.vt.edu/~cs5204/fall07-kafura/Papers/Security/Protection-Lampson.pdf.

[33] J. H. Saltzer and M. D. Schroeder. "The protection of information in computer systems." *Proceedings of the IEEE* 63(9), 1278–1308 (1975). https://doi.org/10.1109/PROC.1975.9939. Complete mediation, changing authority, and protection design principles.