Cryptographic Causation: Tamper-Evident Provenance of Agent Decisions, Beliefs, and Claims

Laurie Scheepers
CodeTonight

2026

Revision provenance: authored by Laurie Scheepers with AI assistance from Claude Fable 5.1 and Astra (gpt-6-astra); every implementation claim not marked *(unverified)* was checked against the reviewed revision by the host session on 2026-09-22.

Abstract

Autonomous AI agents increasingly take consequential actions — moving money, calling tools, deciding outcomes — yet the record of *what* an agent decided, *on what belief*, and *whether every claim it made is true to its sources* is typically an ordinary log: editable without trace, so "trust our records" is the entire security model. We present *cryptographic causation*, a three-leg architecture for checking specified bindings among supplied decision records, context reports, and citation receipts. What a third party can check depends on the profile: HMAC authentication requires the shared secret, public record authentication requires an Ed25519 or ML-DSA-65 profile with independently authenticated keys, and resistance to writer-controlled replacement requires an independent commitment covering the disputed bytes. A signed, content-addressed *decision forest* represents supplied decision records. The proposed design uses an RFC-6962 Merkle commitment for selective disclosure with $O(\log N)$ membership proofs and specifies byte-stable deterministic replay; implementation conformance of the forest commitment and replay procedure remains *(unverified)*. A signed *belief chain* records the context report the agent wrote alongside each decision — what it claimed to believe, not a verified mental state — and cross-links it. A *claim receipt* records checks of enumerated quotations against supplied, hash-pinned source bytes; these checks do not establish exhaustive assertion coverage, source authenticity, semantic support, or an authenticated binding of the complete receipt. Underlying all three is an *exogenous-anchor law*: a proposed external-acceptance policy requiring authenticated authorization that the producer cannot grant itself — not an established theorem that external evaluation improves every loop or cannot be manipulated. The verifier throughout is arithmetic — SHA-256, HMAC or asymmetric-plus-post-quantum signatures, and Merkle proofs — together with an external party. Every claim is stated with an explicit refutation condition; a self-verification release of this paper itself remains future work.

Contents

1 Introduction: the agent-accountability gap

Autonomous software agents select and carry out actions without a human authorising each one. Some actions have consequential external effects: sending a message, committing code, moving money, or deploying a service. A later audit must distinguish what the system recorded from what actually happened. An editable log alone cannot establish its own integrity against the party controlling its storage. Equally, a signed log does not by itself establish that its contents are truthful, complete, or contemporaneous with the actions it describes.

We call this deficit the *agent-accountability gap*. This paper addresses part of it: how a third party can check the integrity and specified bindings of supplied decision records, context reports, and citation receipts. It does not establish complete observation of an agent's conduct or faithful access to its internal state. The name *cryptographic causation* denotes the proposed record architecture, not a proof that a recorded explanation caused an action. Cryptographic verification authenticates defined messages under stated assumptions; causal correspondence requires additional evidence.

Revision provenance and evidence boundary. This paper is authored by V» (Laurie Scheepers), with AI assistance. Claude Fable 5.1 and Astra (gpt-6-astra) assisted this revision. The implementation history and measurements cited here were verified by the host session on 2026-09-22. We distinguish that author-maintained evidence from the public anchoring bundle identified below. Implementation references name files or symbols, not line ranges; a source reference is not itself a proof of implementation correctness.

1.1 What provenance for AI does not yet cover

Training and development provenance concerns datasets, transformations, model artifacts, and approvals. Runtime provenance concerns the records produced while a deployed system selects actions and issues claims. Table 1 distinguishes these objects of study, not exclusive technologies or markets. Either layer can use signatures, transparency logs, access controls, or external timestamps. Blocking an action is compatible with recording the attempted action and its denial.

Provenance layer	**Recorded objects**	**Scope here**
ML-pipeline provenance	Datasets, transformations, model artifacts, and lifecycle approvals	Context for comparison
Agent-runtime provenance	Reported decisions, context summaries, and citation checks during deployment	The supplied records examined in this paper

Table 1: Different provenance scopes, with potentially overlapping mechanisms. Runtime records do not by themselves establish complete runtime conduct.

We claim no new cryptographic primitive, exclusive combination of mechanisms, or demonstrated commercial moat. The individual mechanisms are already shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI. Asqav is MIT-licensed, signs with the same ML-DSA-65 algorithm, and ships RFC 3161 timestamps and RFC 8785 canonicalisation.[1] These precedents rule out treating the component mechanisms as a distinctive commercial contribution.

The commercial claim also failed direct evaluation: four non-Anthropic councils, with zero Anthropic calls, judged the integrated system "a bundle of commodities with a nice story."[2] The only surveyed differentiator is shipping attacks alongside the claim. That distinction rests on reading four vendors' documentation, not running their products; it is not an exhaustive novelty result. EU AI Act Article 12, in force from 2 August 2026, creates a mandatory requirement for automatic record-keeping in its applicable scope.[3] This is a forced-buyer requirement for record-keeping, not evidence that this implementation is compliant, uniquely suitable, or commercially defensible.

[1] Host-verified comparison record, 2026-09-22: Sigstore and Rekor documentation, Git's signed-commit documentation, OpenTimestamps and KERI documentation, and Asqav's licence and signing, timestamping, and canonicalisation documentation. These are documentation-level comparisons, not product execution results.

[2] Host-verified council record, 2026-09-22; associated seals `7dc01f0b739be243` and `9fb9b0fc0f2d4309`.

[3] Regulation (EU) 2024/1689, Article 12; applicability date verified by the host session on 2026-09-22.

1.2 Cryptographic causation

The architecture has three proposed record types, summarised in Table 2. An *Intent Decision Record* (IDR) contains a reported decision, its inputs, and metadata. A *belief record* contains an instrumented context summary; we retain the name but do not interpret it as authenticated access to a model's beliefs. A *claim receipt* records deterministic checks on explicitly submitted quotation and source pairs.

The implementation locations are `lib/idr_forest.py` and `lib/precog/idr.py` for decision records, `lib/context_chain.py` for context reports, and `lib/prove_it.py` and `lib/prove_it_council.py` for citation checking. The design specifications are `IDR.md` and `MEMORY_CONTEXT.md`. These references identify mechanisms; they do not establish that every action passes through them.

Leg	**Recorded evidence**	**Not established by that evidence alone**
Decision record (IDR)	Authenticated decision payload, inputs, and claimed metadata	Execution, completeness, or truthful explanation
Belief record (memory chain)	Authenticated context report and supplied references	Actual internal state, sincerity, or causal influence
Claim record (prove-it)	Presence of enumerated quotation spans in hash-pinned source bytes	Source authenticity, semantic support, factual truth, or exhaustive claim coverage

Table 2: The three record types and their evidentiary limits. Authentication of a report is distinct from validation of what it reports.

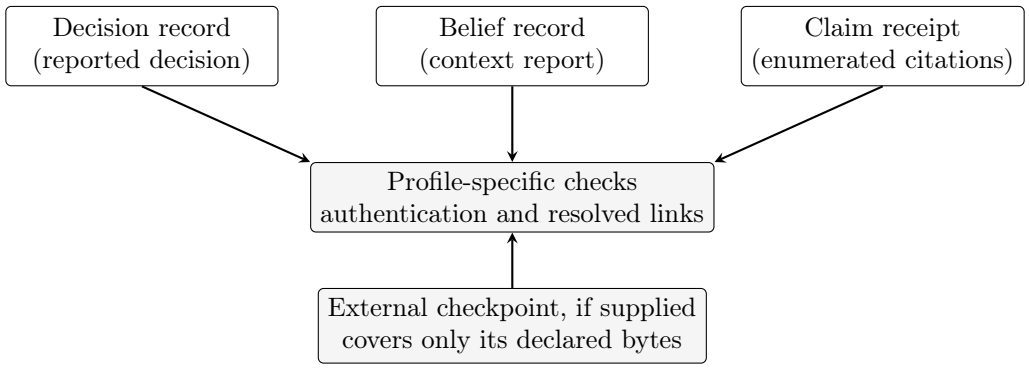

Figure 1: The intended composition and its verification inputs. This schematic does not assert that a complete three-leg bundle has been demonstrated or that the published Bitcoin checkpoint covers all three record types.

Mandatory paired recording is a deployment requirement, not a demonstrated property of the evaluated council path. That path permits hash-only completion when IDR recording is unavailable. Nor does local chain integrity establish complete composition: absent or unresolved references cannot count as verified links. Demonstrating mandatory recording would require complete mediation of consequential actions, crash and retry handling, and reconciliation against independently observed effects. A complete three-leg artifact and a checker requiring every authenticated link remain necessary future work.

The `PrecogIDR` envelope carries protocol metadata, a decision class, a probabilistic identifier, a predecessor reference, a chain depth, a fingerprint, a claimed timestamp, payload and inputs,

an audit block, and optional delegated-decision anatomy. Section 3 develops the record structure. Cooperative append operations do not make the underlying store immutable against its custodian, and a signed timestamp authenticates a clock claim rather than wall-clock truth.

1.3 Falsifiable by construction

We use explicit counterexamples and negative tests to constrain claims. Surviving those tests is not a proof of all implementation behavior. Verification must also admit *unverifiable*: missing keys, dependencies, records, or checkpoint evidence cannot be treated as successful verification.

Known verifier failure and repair. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176 fixed the defect: "`-verify` never compared the record against the hash it sealed." The patch merged on 2026-09-17 at 15:57Z. GRIP#6193 then replaced the single pass token with an explicit list of what a clean verification does *not* assert; it merged at 17:28Z the same day.[4] This was a binding failure, not a SHA-256 collision. Security claims cannot be applied retroactively to the affected verifier. The patch identifiers establish a version boundary; independent reproduction of this failure and repair from a released regression bundle is *(unverified)* here.

Authentication profiles. The default HMAC-SHA256 profile is verifiable only by holders of the secret, each of whom can also forge records. It is not publicly verifiable. Only records actually signed under the Ed25519 or ML-DSA-65 profiles support public-key verification, using independently authenticated public keys and the required algorithm checks. The evaluated HMAC council record must not inherit guarantees from a separate asymmetric exercise.

Verification must check the implementation's defined signed-message encoding; we do not introduce a competing HMAC equation here (`lib/idr_forest.py::verify_chain_integrity`). Changing a signed message while retaining its authenticator should fail authentication under the selected profile. This does not prevent a key holder from replacing both. Negative tests must therefore include altered payloads, recomputed adjacent hashes, substituted references, re-signed records, and missing verification prerequisites. Hash consistency alone is not authenticated provenance.

Commitment domains and disclosure. A semantic content address excludes `id`, `ts`, and `audit`. The forest root over sorted semantic addresses therefore commits to projected semantic bodies, not complete signed envelopes. Changing an excluded timestamp and re-signing a record can preserve that root. The unrestricted assertion that any node change moves the root is false under this definition (`IDR.md`; `lib/idr_forest.py`).

An inclusion proof establishes membership under a particular root and leaf encoding. Its authentication path has size $O(\log N)$ for N leaves. It avoids transmitting other record bodies, but reveals path hashes and structural metadata; predictable records may still permit dictionary testing. Tests must distinguish mutations outside the commitment domain from mutations inside it. An ordered full-envelope checkpoint is a proposed way to bind excluded fields; it is not the semantic-address root already described.

Record reconstruction. The operation called "replay" reconstructs recorded data rather than re-executing a model or an external action (`IDR.md`). A reproducibility test holds the input bytes, target selection, serialization rules, and implementation version fixed and compares reconstructed

[4] GRIP#6176 and GRIP#6193, merge history verified by the host session on 2026-09-22.

outputs. A mismatch can indicate an input, version, serialization, or implementation difference; it does not identify alteration as the sole cause. Cross-platform reproducibility beyond the supplied evidence remains *(unverified)*.

External authority and temporal order. A permitted `human:`, `ci:`, or `council:` prefix classifies a claimed evidence type; it does not authenticate the issuer or establish independence. Likewise, a public genesis identifier supplies structure, not authenticated origin. Authorized external attestations and independently controlled verifier policy are additional requirements.

Before external timestamping was added, the anti-HARKing property—binding commitments before actions—protected only against a third party without the signing key; it was not established against the writer. RFC 3161 trusted timestamps were added in GRIP#6190, "the external time anchor we lacked," merged on 2026-09-17 at 17:22Z.[5] An externally timestamped commitment can establish existence by the attested time under the timestamp authority's assumptions. Establishing that it preceded an action or outcome additionally requires independent evidence of that boundary. Neither signatures nor historical inclusion alone establish the current head of a log.

Public Bitcoin evidence and its coverage. A production decision-chain checkpoint covers 501 records. Its root and manifest SHA-256 are, respectively:

```
0bba0792bf9f71caa815d0c42daf27f45baf07ef9749523be666f103a7f1bcb4
5d5e8854f23eb50e583e02b380f21059651a3a4e99b4454cb221d355725a45b9
```

The anchored leaves are the exact ledger lines in chain order, not sorted content addresses. The OpenTimestamps proof attests this root in Bitcoin blocks 956991 and 956992, whose Merkle roots begin `1b42ef4d` and `b3f15c01`, respectively. A second stamp of the same digest completed in blocks 956948, 956963, and 956970.

The proof, manifest, 501 RFC 6962 leaf hashes, and a standard-library verifier have been public since 2026-09-22.[6] The command `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately, and reports `VERIFIED` only when all three pass. Explorer-backed chain checking depends on the explorer's chain data; it is not independent full-node validation.

The record bodies are not public because their subjects name people. Consequently, the public bundle supports checking the supplied hash commitment and timestamp path, not independently recomputing all leaves from undisclosed records or authenticating their contents. This public hash-proof verification does not require an HMAC secret and does not turn HMAC record authentication into public-key verification. The public bundle does not establish whether the evaluated council record is among the 501 committed ledger lines, nor does it establish continuous anchoring of the semantic forest, context chains, or claim receipts.

Negative predictive result. Record integrity should not be confused with predictive accuracy. The pre-hoc watcher measurement on 2026-09-10 used seed 0, five stratified folds, a 50-event window, and 76 paired rows, with a 13.2% base rate. No predictor's precision interval clears that base rate. The watcher remains `WARN`-only and its hook is unbound.[7] This result supplies no basis for claiming useful predictive enforcement.

[5] GRIP#6190; implementation files `lib/rfc3161_anchor.py` and `lib/continuity_chain.py`; merge history and module locations verified by the host session on 2026-09-22.

[6] Public anchor bundle: `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. See the manifest, timestamp proofs, leaf hashes, and `verify.py` in that directory.

[7] Watcher measurement record dated 2026-09-10, verified by the host session on 2026-09-22.

1.4 Contributions and structure

The contribution claimed here is attack-backed engineering, bounded by the available artifacts and their verification profiles:

1. We distinguish reported decisions, context summaries, and citation checks, and identify what authentication of each can and cannot establish.
2. We describe the IDR forest and separate semantic addressing, record authentication, and external checkpoint commitments (Section 3).
3. We state profile-specific integrity, reconstruction, and membership checks, including missing-evidence outcomes and counterexamples (Section 7).
4. We report the shipped verifier failure and repairs, the failed commercial claim, the negative watcher result, and the publicly checkable 501-record checkpoint, rather than presenting them as evidence of broader guarantees.

The following sections develop the record mechanisms and their intended composition. A complete publicly reproducible three-leg bundle, a detached manifest covering all paper inputs and regression artifacts, and measured end-to-end overhead remain future work. Publishing the Bitcoin anchor bundle does not supply those missing artifacts. The governing standard is narrower than "trust reduces to mathematics": each acceptance claim must identify the checked bytes, the verification profile, the trusted inputs, and the properties left unknown.

2 Threat Model and Requirements

Cryptographic causation concerns the integrity of supplied decision records, recorded context reports, and citation receipts. It does not establish complete runtime conduct, actual internal model state, or the causal relationship between a context report and an action. The decision leg uses the Intent Decision Record (IDR) forest (`lib/idr_forest.py`, `lib/precog/idr.py`; Section 3); the context-report leg uses `lib/context_chain.py` (Section 4); and the claim leg uses `lib/prove_it.py`, with external review in `lib/prove_it_council.py`. Deterministic citation checking establishes quote presence in supplied source bytes, not source authenticity or claim truth.

The central threat is a record author or custodian who can supply misleading records, omit events, replace stored material, or control signing keys. Separating record production from acceptance is a deployment policy, not a property established by a signature or a permitted root prefix. This section distinguishes implemented checks, conditional guarantees, and requirements whose enforcement remains *(unverified)*.

Known failure and revision boundary. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176 fixed the defect: `-verify` had never compared the record against the hash it sealed. The fix merged at 2026-09-17T15:57Z. GRIP#6193, merged at 2026-09-17T17:28Z, replaced the single pass token with an explicit list of what a clean verification does *not* assert.[8] This was an implementation and binding failure, not a cryptographic collision. Claims about the repaired check exclude the pre-fix behavior. These change identifiers do not by themselves establish that every attack proposed below has a published regression fixture.

[8]Revision evidence: GRIP#6176, "–verify never compared the record against the hash it sealed"; GRIP#6193, verification-result scope change. These identifiers and merge times were verified in the author record on 2026-09-22.

2.1 System model and principals

Definition 2.1 (Recorded-decision system)**.** A *recorded-decision system* is a tuple $\Sigma = (\mathcal{D}, \mathcal{B}, \mathcal{C}, \mathcal{R}, \mathsf{Rec}, \mathsf{Adj})$. Here $\mathcal{D}$ contains reported consequential decisions, $\mathcal{B}$ contains instrumentation-supplied context reports, $\mathcal{C}$ contains outward claims, and $\mathcal{R}$ contains candidate record bundles. A bundle may contain an IDR, a context-chain node, and, where applicable, a claim receipt; their presence and authenticated linkage must be checked rather than assumed. The recording operation $\mathsf{Rec}\colon \mathcal{D} \times \mathcal{B} \to \mathcal{R} \cup \{\bot\}$ may fail, represented by $\bot$. Adjudication is profile-dependent:

$$\mathsf{Adj}_G\colon \mathcal{R} \times \mathcal{E} \to \{\textsc{Accept}, \textsc{Reject}, \textsc{Degraded}\},$$

where G declares the required checks and $\mathcal{E}$ contains verification material and verifier-controlled policy. Acceptance means that every check required by G passed. Missing prerequisites yield DEGRADED, not acceptance.

The **agent** P_{ag} produces decisions, context reports, and claims. The **custodian** stores records and may hold signing keys. The **verifier** is software operated under a declared policy; its implementation, configuration, and input parsing are part of the security boundary. The **skeptic** P_{sk} independently evaluates the supplied evidence. An external evaluator, timestamp authority, or checkpoint witness has a separate role whose identity and authority must be authenticated.

Distinct roles need not imply distinct administrative control. A different model name or provider is not sufficient evidence of evaluator independence. Likewise, a roster signed by the author authenticates a recorded participation claim, not the fact that the named provider executed the request.

Public verification is profile-specific. HMAC-SHA256 verification requires the shared secret, and every holder of that secret can forge an HMAC record. Only records actually signed under the Ed25519 or ML-DSA-65 public-signature profiles support signature verification from public material, subject to authenticated public keys. Public verification of a timestamp commitment is separate from verification of a record's HMAC.

2.2 Trust assumptions

No assumption of agent honesty is made. The remaining assumptions depend on the guarantee profile.

(A1) *Cryptography and verification.* Hash commitments require the relevant collision and second-preimage resistance of SHA-256. HMAC authentication against a non-key-holder requires HMAC-SHA256 unforgeability and secrecy of the shared key. Public-signature profiles additionally require signature security and authenticated key distribution. A dual Ed25519/ML-DSA-65 profile requires verification of both components, authenticated algorithm selection, and rejection of stripped components or downgrades; its resilience to one signature scheme failing depends on the other remaining secure. All profiles require correct message encoding, parsing, commitment-domain selection, and verifier behavior. These are not consequences of hash hardness.

(A2) *Independent commitment evidence.* Writer-resistant integrity requires an expected commitment retained outside the writer's control. OpenTimestamps evidence additionally depends on validation of the proof, its binding to the intended digest, and a stated Bitcoin chain-validation and confirmation policy. An explorer supplies chain data under an additional trust assumption. RFC 3161 timestamps depend on an authenticated timestamp authority, its signing key

and certificate validation, and its clock. Neither mechanism establishes truthful contents, completeness, or currentness.

(A3) *Authenticated external evaluation.* Where acceptance requires external evaluation, the verifier needs an authorized issuer, an attestation bound to the evaluated artifact and policy, and evidence appropriate to the claimed result. CI, human review, cross-provider councils, and preregistration are candidate controls, not interchangeable proofs of correctness. Their independence and effectiveness require separate validation. Issuer authorization must not be replaceable by the agent whose work is being evaluated.

Existing external evidence. One production checkpoint is publicly checkable. Its decision-chain root is

`0bba0792bf9f71caa815d0c42daf27f45ba`
`f07ef9749523be666f103a7f1bcb4`

where the two displayed strings concatenate to the full digest. It covers 501 records, with manifest SHA-256

`5d5e8854f23eb50e583e02b380f210596`
`51a3a4e99b4454cb221d355725a45b9`.

The OpenTimestamps proof is attested in Bitcoin blocks 956991 (Merkle root beginning `1b42ef4d`) and 956992 (`b3f15c01`); a second stamp of the same digest completed in blocks 956948, 956963, and 956970. The proof, manifest, 501 RFC 6962 leaf hashes, and standard-library verifier have been public since 2026-09-22.[9] The leaves commit to the exact ledger lines in chain order, not sorted content addresses. The records themselves are not published because their subjects name people. Thus a public verifier can check the root from the published leaf hashes and the timestamp proof, but cannot independently reconstruct those hashes from the undisclosed records or assess their contents. This checkpoint is evidence for one covered prefix, not evidence of continuous anchoring of every IDR, context report, or receipt. It does not discharge assumption (A2).

RFC 3161 trusted timestamps were added in GRIP#6190, merged 2026-09-17T17:22Z, as the external time anchor previously missing.[10] Before that addition, the claimed anti-HARKing protection bound only a third party without the signing key; pre-action commitment was not established against the writer. The addition provides a timestamp mechanism, not retroactive temporal evidence for older commitments. Even a valid timestamp must precede an independently evidenced action or outcome boundary to establish preregistration.

2.3 Adversary model

Definition 2.2 (Adversary)**.** The adversary $\mathcal{A}$ may act as a post-hoc tamperer $\mathcal{A}_{\text{ext}}$ or as the self-serving generator $\mathcal{A}_{\text{self}}$. A non-key-holding tamperer can replace persisted bytes, adjacent hashes, pointers, and supplied verification material, but cannot forge authenticators under independently trusted keys. A key-holding writer or custodian can also re-sign replacement records, manufacture histories, backdate claimed timestamps, withhold records, and serve stale checkpoints. The generator may omit recording, misdescribe context or actions, supply fabricated sources or authority labels, select favorable evaluators, or ratify an authority change using the authority it introduces.

[9] Public anchor bundle: `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. Within that bundle, `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately and reports `VERIFIED` only when all three pass.

[10] GRIP#6190, "the external time anchor we lacked"; implementation files `lib/rfc3161_anchor.py` and `lib/continuity_chain.py`, verified in the author record on 2026-09-22.

Against a non-key-holder, a positively verified authenticator protects its exact signed message. It does not prevent deletion or prove that omitted events never occurred. Against a key-holder, signatures alone do not distinguish an original record from a replacement. An independently retained checkpoint can bind only the bytes within its commitment domain.

The semantic forest address excludes fields including `id`, `ts`, and `audit`. Its root commits to the multiset of projected semantic bodies, not complete signed envelopes. A writer can change an excluded timestamp and re-sign the envelope without changing that semantic root. An unordered semantic root also does not establish enumeration order. In contrast, the production ledger checkpoint described above commits to exact ledger lines in chain order. These are different commitment profiles. A general ordered full-envelope checkpoint for the three-leg architecture is a requirement for future work, not a property inferred from the semantic forest root.

An append operation and advisory lock constrain cooperative writers; they do not make storage immutable against its custodian or make external actions and paired records crash-atomic. A hash stored beside editable content is not an independent expected digest. These limitations are part of the adversary model, not excluded attacks.

2.4 Threat catalogue

Table 3 describes attacks on both record integrity and the interpretation of verified records.

ID	Threat
T1	**Missing or incomplete recording.** An external effect occurs without an IDR, its required context report, or an authenticated link between them.
T2	**Replacement or tampering.** An attacker changes a payload, adjacent hash, signature, predecessor, timestamp, order, or checkpoint; a writer may re-sign the replacement.
T3	**False external authority.** The agent supplies its own evaluation, fabricates an allowed authority prefix or genesis identifier, or presents a controlled evaluator as independent.
T4	**Misleading provenance.** A quotation is absent, an assertion is omitted from the citation inventory, a source is fabricated, or a real quotation does not support the claim.
T5	**Bootstrap authority capture.** An authority-widening change is approved using the authority introduced by that change.
T6	**Wrong-state gating.** A completion or gap decision relies on an unauthenticated status report or an artifact other than the designated authoritative version.
T7	**Staleness and temporal misrepresentation.** An old valid record is presented as current, incompatible heads are served, or a post-outcome commitment is backdated and presented as pre-action.

Table 3: Threats to supplied records and to claims made from them. These threats are not unique to agent systems.

2.5 Requirements

The following are acceptance requirements, not a claim that the current implementation satisfies all of them. Each test states what would contradict the corresponding guarantee.

R1 — Mandatory, paired recording (counters T1). A deployment claiming complete recording must enumerate consequential action paths and require both a signed IDR and a linked context-

report node. It must define durable pending records, action identifiers, crash recovery, retries, and reconciliation against independently observed effects. The evaluated council path can complete with hash-only output when the IDR library is unavailable; mandatory paired recording is therefore not established by that path. The existence of `lib/idr_forest.py` and `lib/context_chain.py` does not establish complete mediation.

> *Required test:* inject recorder failure and crashes at each action boundary. Completion of a covered consequential action without *either* required record or its authenticated pairing falsifies complete-recording enforcement. Comprehensive path coverage remains *(unverified)*.

R2 — Domain-specific tamper evidence (counters T2). A verifier must identify the protected bytes, signing profile, independently trusted key or expected digest, and checkpoint coverage. Hash consistency alone must not imply authentication. Local integrity checking is implemented in `lib/idr_forest.py::verify_chain_integrity`; writer-resistant integrity requires additional external evidence covering the disputed bytes.

> *Required tests:* replace the council payload and adjacent hash together; alter its pointer; re-sign a replacement; remove keys or dependencies; and change a timestamp excluded from the semantic address. A required check that is missing must not pass. The timestamp mutation is a counterexample to unrestricted semantic-root sensitivity, not a hash attack. Acceptance of altered checkpoint-covered bytes contradicts the stated profile, but the cause need not be a cryptographic break.

R3 — Authenticated external evaluation (counters T3). The prefixes {`ci:`, `human:`, `council:`, `hypo:`} in `lib/idr_forest.py` classify claimed evidence types. They do not authenticate origin, independence, or authorization. The fixed context-chain genesis identifier similarly supplies structure, not an external attestation. A stronger acceptance policy requires an issuer, artifact digest, policy/version, result, freshness information, and signature checked against verifier-controlled authorization. Enforcement of that complete attestation policy remains *(unverified)*.

Self-produced telemetry may guide investigation but must not be the sole evidence for a guarantee requiring external evaluation. No formal confidence-monotonicity property is claimed without a specified confidence measure and update rule.

> *Required tests:* submit fabricated `human:`, `ci:`, and `council:` roots, and a newly manufactured chain using the correct genesis constant. Acceptance as authenticated external evidence without an authorized attestation falsifies R3. Provider separation alone cannot satisfy the test.

R4 — Scoped citation verification (counters T4). The deterministic layer in `lib/prove_it.py` must check the submitted quotation spans against the supplied, hash-pinned source bytes. This does not establish source authenticity, semantic support, factual truth, or exhaustive coverage of outward assertions. A complete composition policy would also require an authenticated receipt binding the deliverable, citation specification, source manifest, and resolved IDR and context references. That complete binding is not established by quote matching alone.

> *Required tests:* an absent quoted span must fail quote matching. An omitted claim, fabricated source, irrelevant real quotation, changed specification, or altered chain pointer must not be promoted to verified coverage, authenticity, support, or composition. Enforcement of those stronger predicates remains *(unverified)*.

R5 — No self-ratification of authority (counters T5). An authority-widening change must be authorized under the prior policy by a principal whose authority does not derive from the proposed change. A `human:` label is insufficient: authorization requires authenticated human approval bound to the change and the applicable policy. Complete enforcement remains *(unverified)*.

> *Required test:* attempt to approve an authority-widening change using only the authority it introduces. Acceptance falsifies the access-control requirement, regardless of whether the resulting decision appears reasonable.

R6 — Authenticated canonical-state gating (counters T6). A deployment must identify the authoritative artifact and version for each completion or gap claim, authenticate its origin, and bind the decision to its digest. "Canonical" denotes that explicit policy choice, not factual truth. A status label or local summary is not a substitute for the designated artifact. A generic enforcement gate is *(unverified)*.

> *Required test:* supply a favorable status report contradicted by the designated artifact, or substitute an uncommitted local tree. Acceptance under a profile requiring authoritative-state inspection falsifies R6.

R7 — Separate historical membership, currentness, and pre-action commitment (counters T7). An inclusion proof establishes membership in a particular checkpoint. Currentness additionally requires an authenticated head, checkpoint sequence and extension evidence, latest-checkpoint discovery, an explicit staleness policy, and equivocation handling. A signed timestamp authenticates a claimed time, not clock correctness. Without freshness evidence the result must be currentness unknown.

For anti-HARKing against the writer, an independent timestamp or witness must bind the specified commitment before an independently evidenced action or outcome boundary. RFC 3161 support does not by itself demonstrate this end-to-end ordering protocol. Continuous fresh-head verification and comprehensive pre-action enforcement remain *(unverified)*.

> *Required tests:* serve an old valid checkpoint, incompatible heads, and a superseded record with a valid inclusion proof. Separately, let the writer learn an outcome and then create, backdate, and re-sign a hypothesis. Acceptance as current or pre-action without the respective independent evidence falsifies R7.

2.6 Threat-to-requirement mapping

Table 4 separates available mechanisms from unestablished enforcement. A passed local check must not silently satisfy a stronger requirement.

We use separate evidence statuses: `HASH_CONSISTENT` for a recomputed digest match; `AUTHENTICATED` for positive verification under the declared key profile; `EXTERNALLY_ANCHORED` for a validated commitment binding; `COMPOSITION_COMPLETE` for all required authenticated links; and `CURRENT` for the separate freshness predicate. These are specification categories, not a claim that all are shipped API return values. `UNVERIFIABLE` denotes missing prerequisites and cannot substitute for a pass. A record may be historically authenticated while its currentness is unknown.

Listing 1 is a proposed acceptance discipline, not an existing aggregate implementation. Its checks include governance and action evidence that cannot be derived from record hashes alone. Record reconstruction can test deterministic processing of supplied inputs; it does not replay external execution or establish causation.

Threat	Requirement	Mechanism and boundary
T1	R1	IDR and context writers exist; complete mediation and atomic pairing are not established.
T2	R2	Hashes, authenticators, and checkpoint proofs protect their specified domains; semantic roots exclude envelope fields.
T3	R3	Root namespaces exist; authenticated issuer authorization and evaluator independence require separate evidence.
T4	R4	Quote matching checks supplied spans; authenticity, support, coverage, and complete linkage are separate.
T5	R5	Prior-policy authorization is required; a human prefix is not authorization.
T6	R6	Artifact-bound state inspection is required; a universal gate is not demonstrated.
T7	R7	External timestamps and historical membership are available; currentness and pre-action ordering need additional protocols.

Table 4: Requirements and their evidence boundaries, rather than a claim of complete implementation.

The verifier, not the submitted bundle, selects the required profile. Otherwise an attacker could remove a required layer and request acceptance under a weaker one. Acceptance must identify G and report its exclusions; it must not be presented as an unqualified proof of the run.

2.7 Non-goals and scope

Truth, completeness, and confidentiality. Authenticated records may be false, incomplete, or written after the events they describe. Context reports do not prove mental states. A quote match does not prove a proposition. Complete action recording is a requirement whose enforcement is not demonstrated, not a consequence of local signature validity.

Merkle inclusion proofs avoid transmitting other record bodies but reveal path hashes and structural metadata. Unsalted hashes of predictable records may permit dictionary testing. No hiding-commitment or zero-knowledge property is claimed. External-anchor availability and timeliness are also not guaranteed; missing anchor material must leave the corresponding guarantee unverifiable.

Predictive monitoring. A watcher is not an enforcement substitute. In the measurement of 2026-09-10 (seed 0, five stratified folds, a 50-event window, 76 paired rows, and a 13.2% base rate), no predictor's precision interval cleared the base rate. The pre-hoc watcher remains WARN-only and its hook is unbound.[11] This result does not support a positive predictive-discrimination or automatic blocking claim.

Novelty and commercial scope. No new cryptographic primitive, exclusive mechanism combination, or demonstrated commercial moat is claimed. Four non-Anthropic councils, with zero Anthropic calls and seals `7dc01f0b739be243` and `9fb9b0fc0f2d4309`, judged the integrated system "a bundle of commodities with a nice story."[12] The individual mechanisms are shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI. Asqav is MIT-licensed, signs with the same ML-DSA-65, and ships RFC 3161 timestamps and RFC 8785 canonicalisation.[13] The only surveyed

[11] Watcher measurement record dated 2026-09-10, verified in the author record on 2026-09-22.

[12] Commercial-claim council outcomes under the two stated seals, verified in the author record on 2026-09-22.

[13] Vendor-documentation survey recorded in the author record and verified on 2026-09-22. This is documentation evidence, not a comparative execution study.

```
Adjudicate(bundle, evidence, verifier_policy):
  G <- verifier_policy.required_guarantee_profile
  results <- empty map

  for check in G.required_checks:
    # Each check declares its byte domain, trusted inputs,
    # algorithm/profile, and required evidence.
    results[check] <- evaluate(check, bundle, evidence)

  # Possible checks include:
  # digest consistency; signature authentication;
  # external commitment coverage; required-link resolution;
  # authorized external evaluation; recording completeness;
  # prior-policy authority; authoritative state;
  # freshness; independently evidenced pre-action ordering.

  if any required result is FAIL:
    return REJECT, results
  if any required result is UNVERIFIABLE:
    return DEGRADED, results
  return ACCEPT_FOR_G, results
```

Listing 1: Schematic profile-specific adjudication. The operations below are specification steps, not claimed implementation APIs.

differentiator is shipping attacks alongside the claim; that finding rests on reading four vendors' documentation, not running their products. It is not an exclusivity guarantee. EU AI Act Article 12, in force 2 August 2026, creates a forced buyer for automatic record-keeping, not a requirement to purchase this implementation or evidence of its technical superiority.[14]

2.8 Synthesis

The defensible guarantee is profile-specific: a verifier can check declared bindings over supplied material under explicit key, implementation, and external evidence assumptions. The production timestamp demonstrates one ordered ledger commitment; it does not establish truthful records, complete three-leg composition, or current state. The 2026-09-17 council-verifier failure demonstrates why hash recomputation and a broad pass token are insufficient.

R1–R7 specify the evidence needed for stronger claims. Missing evidence must remain visible, and a local integrity result must not be promoted to complete recording, authenticated external authority, semantic truth, or pre-action ordering. Sections 3 through 8 describe the component mechanisms; their presence alone does not discharge these requirements.

Revision provenance. This section was revised by V» (Laurie Scheepers), with AI assistance from Claude Fable 5.1 and Astra (gpt-6-astra). The factual revision record used here was verified on 2026-09-22.

[14] Regulation (EU) 2024/1689, Article 12, record-keeping requirement; applicability and commercial-scope facts verified in the author record on 2026-09-22.

3 The Intent Decision Record forest

The decision-record leg of cryptographic causation is a forest of *Intent Decision Records* (IDRs). An IDR records a supplied decision, its inputs, and its predecessor reference. Content addresses support semantic lookup; authenticators bind a defined signed body; Merkle proofs establish membership in a particular commitment. These operations do not establish that the recorded decision was executed, that all decisions were recorded, or that the inputs faithfully describe an agent's internal state.

The implementation separates envelope construction and persistence in `lib/precog/idr.py` from forest operations in `lib/idr_forest.py`; authentication also uses `lib/audit_chain.py`. These are implementation references, not a substitute for a revision-pinned executable specification. Public verification of record authentication is available only for records using the Ed25519 / ML-DSA-65 profiles with authenticated public keys. The default HMAC profile requires the shared secret, whose holders can also forge records.

Known failure and revision boundary. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176 fixed the defect: "`-verify` never compared the record against the hash it sealed." The patch merged at 2026-09-17T15:57Z. GRIP#6193, merged at 2026-09-17T17:28Z, replaced the single pass token with an explicit list of what clean verification does *not* assert.[15] This was a binding failure, not a SHA-256 collision. The repaired behavior must not be attributed to earlier versions, and the patch does not establish correctness of all other verification paths. A complete public regression bundle for this council failure is not established here *(unverified)*.

3.1 The IDR envelope

Table 5 describes the HAPPI envelope discussed in this section, following `lib/precog/idr.py` and `IDR.md`; the evaluated historical record carries envelope version `1.3`, and the reviewed revision of the library emits version `1.5` (`lib/precog/idr.py::HAPPI_VERSION`). It does not establish compatibility with other wire versions. A verifier must select the applicable version and serialization profile rather than silently interpreting an unknown version as this one.

Field	Type	Role
`happi`	string	Declared wire version, here `"1.3"`
`kind`	string	Declared decision class
`id`	string	Probabilistically generated record identifier
`predecessor_idr`	string | null	Parent identifier, not a parent content address
`depth`	int	Claimed depth in the chain
`fingerprint`	string	Context lookup or reconstruction key
`ts`	string	Claimed ISO-8601 UTC timestamp
`decision`	object	Recorded decision payload
`inputs`	object	Recorded inputs
`audit`	object	Authentication metadata and authenticator
`decision_anatomy`	object | null	Optional agent-supplied decision report

Table 5: The envelope schema described by `lib/precog/idr.py` and `IDR.md`. Field presence does not establish the truth of its value.

The identifier format is `precog-<unix_seconds>-<8 hex>`: a four-byte random suffix from

[15] Implementation history: GRIP#6176 and GRIP#6193, verified in the host session on 2026-09-22.

`secrets.token_hex(4)` (`lib/precog/idr.py::build_idr`). This reduces same-second collisions but does not guarantee uniqueness. Collision detection is therefore required; a larger identifier space is future work. The optional `decision_anatomy` field is a recorded report, not evidence of an actual mental state or its causal role.

Listing 2 is schematic, not a verification fixture. For the described compatibility rule, a null `decision_anatomy` is omitted from both the semantic projection and the signable body. This normalization does not imply that the original persisted JSON bytes are identical.

```
{
  "happi": "1.3",
  "kind": "precog-decision",
  "id": "precog-<unix_seconds>-<8 hex>",
  "predecessor_idr": "<parent identifier, or null>",
  "depth": 0,
  "fingerprint": "<context key>",
  "ts": "<claimed ISO-8601 UTC time>",
  "decision": {},
  "inputs": {},
  "audit": {
    "scheme": "hmac-sha256",
    "key_fingerprint": "<key lookup hint>",
    "signature": "hmac-sha256:<hex>"
  },
  "decision_anatomy": null
}
```

Listing 2: Schematic HAPPI 1.3 envelope; placeholders are not real data.

Test condition. Under the same serialization profile, absent and null `decision_anatomy` must produce identical semantic and signable bytes. This is a compatibility requirement; exhaustive cross-version test coverage is *(unverified)*.

3.2 Content addressing

Define $P(r)$ to remove

$$V = \{\texttt{id}, \texttt{ts}, \texttt{audit}\}$$

from envelope r, and also to remove `decision_anatomy` when its value is null. The semantic content address described in `lib/precog/idr.py` is

$$\text{addr}(r) = \texttt{sha256:} \,\|\, \text{hex}(\text{SHA256}(\text{canon}(P(r)))) \,.$$

Here canon denotes the implementation's serialization, not an assertion of RFC 8785 conformance. JSON key sorting alone does not specify number formatting, duplicate-key rejection, Unicode treatment, or non-finite-value handling. A normative cross-language byte profile and complete test vectors remain future work; cross-language equivalence is *(unverified)*.

Two records differing only in excluded fields have the same semantic address. In particular, the address does not bind their identifiers, timestamps, or authenticators. The predecessor *identifier* remains in $P(r)$, so this is not recursive deduplication of otherwise equivalent histories whose parent identifiers differ.

Test condition. Changing only `id`, `ts`, or `audit` must leave the semantic address unchanged. Changing the serialized projection must change it under the hash security assumptions. A writer

Mechanism	Commitment domain	Boundary
Semantic address	Serialized $P(r)$	Excludes identifier, timestamp, and audit block
Envelope authenticator	Entry hash of normalized signable body	Includes identifier and timestamp, but excludes audit block
Semantic forest root	Sorted multiset of semantic addresses	Does not bind complete envelopes or their enumeration order
Production ledger anchor	Exact ledger lines in chain order	A separate profile covering one 501-record prefix

Table 6: Distinct commitment domains. A guarantee for one row must not be transferred to another.

who changes a timestamp and re-signs a terminal envelope supplies a direct counterexample to the stronger claim that every envelope change moves the semantic forest root; no collision is required.

3.3 Signing and tamper evidence

The signing domain differs from the semantic domain. Let $B(r)$ be the envelope with `audit` removed and null `decision_anatomy` omitted, as described by `lib/idr_forest.py::signed_body`. In the entry-hash profile described by `lib/audit_chain.py`, write

$$h(r) = \texttt{sha256:} \,\|\, \mathrm{hex}(\mathrm{SHA256}(\mathrm{canon}(B(r)))), \qquad \sigma(r) = \mathrm{HMAC}_k(\mathrm{encode}(h(r))).$$

In the implementation the HMAC input is the UTF-8 encoding of the entry-hash hex string, where the entry hash is `compute_entry_hash(signed_body(r))` (`lib/precog/idr.py::_sign_real`; `lib/idr_forest.py::signed_entry_hash`); it is not the semantic address and not a direct encoding of $B(r)$. Independent implementations need the exact serialization and entry-hash encoding. Their interoperability is *(unverified)* until those bytes are specified and tested.

The profiles have different trust boundaries:

- **HMAC-SHA256.** A verifier needs the shared secret k. Every holder can generate an accepted authenticator. This is not a publicly verifiable signature or evidence against another key holder.
- **Ed25519 / ML-DSA-65.** Records actually signed under the asymmetric profile can be checked using independently authenticated public keys. For the combined `ed25519+ml-dsa-65` profile, the claimed resilience to one signature scheme failing additionally requires mandatory verification of both components, secure encoding, authenticated algorithm selection, and downgrade resistance. Exhaustive downgrade testing is *(unverified)*.
- `sha256-placeholder`**.** Legacy hash-only records provide consistency checks, not authenticated provenance.

The short `key_fingerprint` in an audit block is a lookup hint, not an independently authenticated key identity. A signature authenticates a claimed timestamp; it does not establish clock correctness.

Because `predecessor_idr` belongs to $B(r)$, changing that value while retaining the original authenticator must fail authentication when the required key and implementation are available. A key holder can instead change the body and generate a new authenticator. Detecting that replacement requires an independently retained commitment covering the changed bytes. An editable payload and an editable adjacent hash do not provide this protection, as the GRIP#6176 failure demonstrated.

Test condition. With keys, dependencies, and the selected profile available, a changed signed message or corrupted authenticator must not pass authentication. Missing prerequisites must

produce an unverifiable result, not a pass. Re-signing by a legitimate key holder is outside this local tamper-detection guarantee.

3.4 The forest and its exogenous roots

The forest representation associates envelopes with predecessor edges and a claimed root identifier (`lib/idr_forest.py`). Its admitted root prefixes are

$$\mathcal{A} = \{\texttt{ci:}, \texttt{human:}, \texttt{council:}, \texttt{hypo:}\}.$$

These prefixes classify claimed evidence types. They do not authenticate a CI result, human authorization, council participation, or preregistration. An agent can write a permitted prefix itself. Likewise, a fixed public genesis string supplies a structural starting point, not an authenticated origin.

An external-authority claim requires additional evidence: an issuer authorized by verifier-controlled policy, an attestation bound to the evaluated artifact and policy, and a verified signature. A suitable manifest would include issuer identity, artifact digest, policy version, result, and freshness information. This is a deployment requirement and proposed extension; enforcement by the prefix check is not claimed.

The construction in `lib/idr_forest.py` returns a new forest, raises `IdrForestError` on duplicate node identifiers and unresolved predecessors, and traverses ancestry with cycle detection (`lib/idr_forest.py`). These are structural properties. An immutable in-memory representation does not make its stored envelopes immutable against their custodian.

Temporal evidence is separate. RFC 3161 trusted timestamps were added in GRIP#6190, "the external time anchor we lacked," merged on 2026-09-17T17:22Z, in `lib/rfc3161_anchor.py` and `lib/continuity_chain.py`.[16] Before that addition, the anti-HARKing property bound only a third party without the signing key; it was not established against the writer. An RFC 3161 token supplies external time evidence only for its bound digest, under the timestamp authority's trust assumptions. Establishing *pre-action* commitment additionally requires that token to precede an independently evidenced action or outcome boundary. Timestamp support does not show that every IDR has such a token.

Test conditions. Rejecting an unknown prefix tests namespace policy. Rejecting a forged `human:` or `council:` attestation tests authenticated origin; the former is not a substitute for the latter. A writer-controlled backdated and re-signed hypothesis must not be accepted as pre-action evidence without the independent temporal binding. These authority and ordering tests are requirements, not reported passing results.

3.5 Canonicalisation as an idempotent projection

The forest normalization described in `lib/idr_forest.py` re-serializes records and reconstructs edges from their recorded `predecessor_idr` fields. On inputs accepted by the selected serialization profile, its intended property is

$$\mathcal{F}(\mathcal{F}(F)) = \mathcal{F}(F).$$

This is an idempotence requirement, not a contraction theorem. Normalization neither authenticates a predecessor nor recomputes a decision. If a normalized edge disagrees with an earlier edge map, the record's authenticator and referenced target still require verification.

[16] GRIP#6190; host-verified module locations are `/Users/void/.grip/lib/rfc3161_anchor.py` and `/Users/void/.grip/lib/continuity_chain.py`. Implementation history verified on 2026-09-22.

Test condition. Under a fixed implementation and serialization profile, repeated normalization of an accepted forest must be structurally equal. Malformed or unsupported inputs must be rejected explicitly rather than used to extend the claim to every possible JSON object.

3.6 Verification: a tri-state verdict

The local integrity interface `lib/idr_forest.py::verify_chain_integrity` returns a `Verdict` (`lib/audit_chain.py::Verdict`) whose members are VERIFIED, BROKEN, and DEGRADED: VERIFIED iff every node verifies, BROKEN if any node's signature fails to match, and DEGRADED if a node uses a recognised scheme whose check could not be completed. These are local results, not proofs of execution, authority, completeness, or currentness. Table 7 states the interpretation required for the guarantees in this section; exhaustive conformance of all error paths is *(unverified)*.

Local result	Required interpretation
VERIFIED	Every required local integrity check for the declared profile positively passed using the required material.
BROKEN	At least one performed authentication or structural check failed.
DEGRADED	A required check could not be completed, or only a legacy hash-consistency check was available, with no established failure taking precedence.

Table 7: Local integrity semantics. Missing keys or dependencies cannot satisfy an authenticated profile.

For reporting beyond local integrity, we distinguish `HASH_CONSISTENT`, `AUTHENTICATED`, `EXTERNALLY_ANCHORED`, `COMPOSITION_COMPLETE`, and `CURRENT`. These are evidence categories here, not a claim that the library exports a new five-state API. Each category requires its own positive checks; missing prerequisites mean `UNVERIFIABLE` for that category. In particular:

- HMAC authentication requires the secret; public-key authentication requires the relevant authenticated public keys.
- External anchoring requires proof covering the applicable commitment domain, not merely a valid record signature.
- Complete composition requires authenticated, resolved links to the required context report and claim receipt. Local IDR validity does not supply those links.
- Currentness requires authenticated head information, checkpoint extension evidence, and a freshness policy. Historical membership does not identify the latest head.

The condition "not BROKEN" is insufficient: it includes DEGRADED. The known council failure and the reporting change in GRIP#6193 motivate explicit per-layer results rather than an unrestricted `PROVEN` label.

Test conditions. Mutation with all prerequisites available must fail the affected check. The same mutation with unavailable verification material may be unverifiable rather than demonstrably broken, but must not pass. Missing keys, unavailable signature dependencies, stripped signature components, changed scheme identifiers, and missing linked records require separate negative tests.

3.7 Merkle commitment and selective disclosure

The semantic forest profile described in `lib/idr_forest.py` builds an RFC 6962 Merkle tree over the sorted multiset of node content addresses. It commits to that multiset of projections, not to complete signed envelopes. Sorting removes enumeration order from this commitment. Changing an excluded timestamp or audit block leaves the root unchanged; changing the committed multiset changes the root under the relevant hash security assumptions.

An inclusion proof establishes membership of a supplied semantic address in a supplied root. It does not by itself authenticate the root or establish membership of every field in the complete envelope. The described proof response includes the node identifier, content address, authentication path, forest root, and node count (`lib/idr_forest.py`). Verification avoids transmitting other record bodies, but reveals path hashes and structural metadata. Unsalted hashes of predictable records may also permit dictionary testing. This is not a hiding commitment or a zero-knowledge guarantee.

A distinct, public production anchor. The production decision-chain commitment covers 501 records, using the *exact ledger lines in chain order*, not sorted semantic addresses. Its root and manifest digest are:

```
root:
0bba0792bf9f71caa815d0c42daf27f45
baf07ef9749523be666f103a7f1bcb4
manifest SHA-256:
5d5e8854f23eb50e583e02b380f210596
51a3a4e99b4454cb221d355725a45b9
```

Each pair of displayed lines is one concatenated digest. An OpenTimestamps proof attests this root in Bitcoin blocks 956991 (block Merkle-root prefix `1b42ef4d`) and 956992 (prefix `b3f15c01`). A second stamp of the same digest completed in blocks 956948, 956963, and 956970.

The proof, manifest, all 501 RFC 6962 leaf hashes, and a standard-library verifier have been public since 2026-09-22.[17] Within that bundle, `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately and reports `VERIFIED` only when all three pass. Explorer mode relies on its external chain-data source; independent Bitcoin validation requires appropriately authenticated chain data.

The records themselves are not published because their subjects name people. Consequently, public users can check the root from the published leaf hashes and verify its timestamp proof, but cannot recompute those hashes from undisclosed ledger lines or authenticate those records' contents. This public Merkle/timestamp check is independent of the records' authentication scheme: it does not make HMAC records publicly authenticatable. It also does not establish anchoring of a later council record, continuous forest anchoring, or coverage of context reports and claim receipts.

Test conditions. A generated inclusion proof must reproduce its root, and changes to its leaf or path must not pass against that fixed root. A timestamp-change-and-re-sign test must preserve the semantic root while changing any commitment to the affected exact ledger bytes. A domain-separated, ordered full-envelope checkpoint for the semantic forest is future work, not an inferred property of its current root.

[17] Public anchor bundle: `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. The bundle is the evidence source for the production root, manifest, leaf hashes, proofs, and verification command.

3.8 Deterministic record reconstruction

The operations called "replay" in `lib/idr_forest.py` reconstruct a root-to-node sequence from supplied records; they do not re-execute a model or external action. For sequence $(r_0, \ldots, r_n)$, the described digest is

$$D = \mathrm{SHA256}\big(\mathrm{join}_{\mathtt{LF}}\big(\mathrm{canon}(r_0), \ldots, \mathrm{canon}(r_n)\big)\big)\,.$$

The separator and serialization profile are part of the digest domain.

Reconstruction reports an integrity verdict alongside its sequence and digest. The wrapper's refusal of a BROKEN forest protects only against failures detected by that checker. It does not exclude accepted forgeries, replacement histories signed by a key holder, or incomplete verification material.

Determinism means equal input bytes, target selection, implementation version, and serialization behavior produce equal output bytes. A mismatch may arise from a changed record, changed selection, implementation drift, or serialization differences; it does not identify the cause. A digest computed locally is not an external authority or time anchor.

Test conditions. Independent reconstruction under the same declared profile must produce equal digests for identical inputs. A BROKEN result must not be presented as authenticated reconstruction. Cross-platform byte stability remains *(unverified)* without published test vectors and execution results.

3.9 Chain I/O and concurrency

The persistence path `lib/precog/idr.py::append_idr` appends JSON-lines records under an exclusive `fcntl.flock` lock and requests durability with `os.fsync`. This coordinates cooperative writers under the relevant filesystem assumptions. It does not prevent a privileged custodian from rewriting storage, guarantee recovery from partial writes, or make recording atomic with an external action.

Backward traversal follows predecessor identifiers and uses a visited set to detect cycles. Collision handling, malformed-tail recovery, crash semantics, and concurrent access require explicit tests. Proposed tests include interruption during append, duplicate identifiers, cyclic pointers, and concurrent writers; this section reports no measured stress-test or latency result.

Mandatory paired recording is a deployment requirement, not a demonstrated property of this append interface. The evaluated council path permits hash-only completion when IDR recording fails (`lib/sovereign_council.py::_seal_idr`). Therefore a completed decision need not have both its IDR and paired context report. A fail-closed action gateway, durable pending records, and reconciliation against independently observed actions remain future work.

3.10 Synthesis

The forest supports inspection of supplied decision records under explicit profiles. Semantic addresses bind projections; authenticators bind signable bodies; Merkle proofs bind leaves to particular roots; external timestamps bind particular digests to external time evidence. None alone establishes truthful reporting, complete recording, authorized external evaluation, pre-action timing, or currentness. HMAC verification requires a secret; public record authentication is restricted to the Ed25519 / ML-DSA-65 profiles with authenticated public keys.

Predictive and commercial limits. Record integrity is not evidence of predictive usefulness. The pre-hoc watcher measurement on 2026-09-10 used seed 0, five stratified folds, a 50-event window,

and 76 paired rows at a 13.2% base rate. No predictor's precision interval cleared that base rate; the watcher remains `WARN`-only and its hook is unbound.[18]

Nor does this integration establish a commercial moat. Four non-Anthropic councils, with zero Anthropic calls, judged the integrated system "a bundle of commodities with a nice story"; the supplied seals are `7dc01f0b739be243` and `9fb9b0fc0f2d4309`.[19] The individual mechanisms are shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI. Asqav is MIT-licensed, signs with the same ML-DSA-65 scheme, and ships RFC 3161 timestamps and RFC 8785 canonicalisation. The only surveyed differentiator is shipping attacks alongside the claim; that conclusion rests on reading four vendors' documentation, not running their products.[20] EU AI Act Article 12, in force on 2 August 2026, is a forced buyer of automatic record-keeping, not evidence that this implementation is unique or meets all applicable requirements.[21]

The bounded engineering contribution is to pair these commodity mechanisms with explicit commitment domains, negative results, and reproducible attacks. The public ledger-anchor bundle supports one such verification claim. Unreleased regression fixtures and proposed enforcement mechanisms must remain distinguishable from demonstrated behavior.

Revision provenance. This paper is authored by V» (Laurie Scheepers) with AI assistance. Claude Fable 5.1 and Astra (gpt-6-astra) assisted with the revision.

4 The Memory–Context Belief Chain

The decision record captures a supplied account of what an agent decided. The memory–context belief chain adds recorded context summaries, intended next steps, and optional references to decisions. We retain the term *belief chain* as the name of this structure, not as a claim that its contents reveal an agent's actual mental state. Authentication of a context report does not establish that the report faithfully describes internal model state, caused a decision, or was recorded contemporaneously with an external action.

The chain is a logically ordered sequence of authenticated records. Cooperative writers may append records, but an append interface does not prevent a privileged custodian from rewriting storage. Nor does this section establish mandatory recording, atomicity between records and actions, or complete decision–context–receipt composition.

Commitment and authentication domains. Let H denote SHA-256, let $P(n)$ be the semantic projection of a node, and let enc denote the implementation's serialization. The semantic content address is

$$a(n) = H(\mathrm{enc}(P(n))).$$

The projection excludes the volatile envelope fields `id`, `ts`, and `audit`; it is not a commitment to the complete signed envelope. The address and authentication domains must therefore be kept separate. We write $M(n)$ for the exact message authenticated under the selected record profile, rather than asserting a second signing-byte definition here. The relevant implementation references are

[18] Watcher measurement record dated 2026-09-10, verified by the host session on 2026-09-22. No positive predictive result is inferred from the IDR mechanisms.

[19] Council outcomes and seals, verified by the host session on 2026-09-22.

[20] Commodity-mechanism and Asqav comparison: documentation survey verified by the host session on 2026-09-22; no comparative product execution was performed.

[21] Regulation (EU) 2024/1689, Article 12; applicability date verified by the host session on 2026-09-22.

`lib/context_chain.py` and `lib/idr_forest.py`; a normative cross-language encoding specification and accompanying test vectors remain future work.

For the HMAC profile, verification requires the shared secret. Every holder of that secret can also create replacement authenticators. Only records actually signed under the Ed25519 / ML-DSA-65 profiles are publicly verifiable, and their verification requires independently authenticated public keys. Neither profile by itself prevents a signing-key holder from replacing an unanchored history.

4.1 The context-delta

A *context-delta* records a context report at a checkpoint. Its payload is carried in a `decision` body. Table 8 describes the schema documented in `MEMORY_CONTEXT.md`; these field descriptions are not evidence that the supplied report is accurate.

Field	Type	Role
`next_step`	object	Reported next-step intent, stored by value; `{}` when absent.
`summary`	string	Kernel summary, truncated to 600 characters on write (`lib/context_chain.py`).
`title`	string	Human-readable checkpoint label.
`tier`	string	Priority used when folding the chain.
`paramount`	bool	Marks a report as containing a fact intended to be retained.

Table 8: Context-delta payload fields described in `MEMORY_CONTEXT.md`.

The documented fold-priority order is

$$\mathsf{feedback} \succ \mathsf{project} \succ \mathsf{user} \succ \mathsf{reference}. \tag{1}$$

Classification is implemented in `lib/context_chain_classify.py`. This ordering controls selection under a budget; it does not rank the truth or evidential strength of reports.

The `next_step` field is operationally transient: a resuming session can consume it directly without resolving another object. That usage does not make it one of the excluded envelope fields in $P(n)$.

Integrity boundary. A change that changes $M(n)$ must invalidate the unchanged authenticator, under the selected profile's security assumptions. This is not a claim about every byte of a JSON file: insignificant serialization changes may leave the authenticated message unchanged. It is also not a claim against a party that can recompute a valid authenticator.

A required negative test changes a covered payload field while retaining the original authenticator. With the correct verification material available, accepting that record as authenticated would refute the integrity claim. Without that material, the result must be unverifiable rather than a positive authentication result. The publication of a revision-pinned context-delta mutation corpus is future work; its coverage is currently *(unverified)*.

4.2 The temporal spine and its genesis convention

Context-deltas form a sequence $C = (n_0, n_1, \ldots, n_m)$ through `predecessor_idr`. This field is a predecessor record reference, not the optional semantic-address citation to a decision described below. We use $\mathrm{ref}(n)$ for the predecessor identifier accepted by the chain resolver; we do not equate it with $a(n)$. The implementation reference is `lib/context_chain.py`.

For a complete linear chain, the intended structural invariant is

$$\begin{aligned} \mathrm{pred}(n_0) &= A_{\mathrm{genesis}}, \\ \mathrm{pred}(n_i) &= \mathrm{ref}(n_{i-1}) \quad (1 \le i \le m), \\ \mathrm{depth}(n_i) &= i, \end{aligned} \tag{2}$$

where $A_{\mathrm{genesis}} =$ `council:context-chain-genesis`.

This public constant supplies a common structural starting point. It is not an authenticated council attestation. Anyone can copy it, including a writer manufacturing a new chain. Similarly, an allowed `council:`, `human:`, or `ci:` prefix classifies a claimed evidence type; it does not authenticate its issuer or establish independence.

A structural test must reject an incorrect genesis predecessor, an incorrect depth, or an unresolved predecessor when claiming complete spine validity. A separate origin test must attempt to manufacture a chain using the *correct* genesis constant. Passing structural checks on that chain must not be reported as authenticated external origin. Establishing origin would require an attestation bound to the actual genesis envelope and checked against an authorized external key. Such a check is not established here.

Time and writer resistance. Sequence order is not wall-clock evidence. A signed timestamp authenticates a writer's timestamp claim, not the clock or the relationship between a record and an action. RFC 3161 trusted timestamps were added in GRIP#6190, "the external time anchor we lacked," merged 2026-09-17T17:22Z (`lib/rfc3161_anchor.py`; `lib/continuity_chain.py`). Before that addition, the anti-HARKing property protected commitments only against a third party without the signing key; it was not established against the writer.

Availability of timestamping code does not establish timestamp coverage for this context chain. A writer-resistant pre-action claim requires an independently verified timestamp on the relevant commitment and independent evidence that the action or outcome boundary followed it. That temporal relationship for the context records described here remains *(unverified)*.

4.3 Cross-referencing the decision record

A context-delta may carry `decision.records_idr`, a semantic content address naming a decision in `state/precog/idr.jsonl`. This optional citation is distinct from the same-chain `predecessor_idr` reference (`lib/context_chain.py`).

Three questions must be separated:

1. Is the citation field covered by the context record's authenticator?
2. Does the address resolve to a supplied decision record?
3. Does the target itself pass the required verification profile, and does the bundle contain all required links?

Including `records_idr` in the authenticated payload addresses the first question, not the other two. In particular, a semantic address does not bind target fields excluded from the semantic projection.

The documented citation probe treats an absent citation as vacuously resolved and an absent target as dangling:

$$\mathsf{cite}(n) = \begin{cases} \textsc{resolved} & \text{no } \texttt{records\_idr} \text{ is supplied,} \\ \textsc{resolved} & \text{the supplied address is present in the ledger,} \\ \textsc{dangling} & \text{the supplied address is absent.} \end{cases} \tag{3}$$

This is a local diagnostic convention, not a composition predicate. A dangling citation need not invalidate the context record's own authenticator. Conversely, a locally valid record with an absent or dangling citation cannot satisfy a profile requiring a paired decision.

Required composition checks. Complete decision–context–receipt verification must require the relevant links, resolve their targets, positively verify those targets, and check the specified relationships among them. The citation probe alone does none of this. A complete three-leg bundle and a revision-pinned composition checker are not demonstrated in this section.

Required negative tests include alteration of the citation without reauthentication, deletion of the target, substitution of an unauthenticated target, and omission of the citation. The first tests local authentication; the remaining cases must prevent a complete-composition result even if local context integrity still passes. These are test requirements, not reported passing results.

4.4 Folding the chain into a bounded kernel

A resuming session has a finite context budget. A *context-snapshot* records a bounded kernel derived from the chain and digests intended to make that derivation checkable. The implementation references are `lib/context_snapshot.py` and `MEMORY_CONTEXT.md`. Table 9 distinguishes output commitments from completeness claims.

Field	Meaning
`kernel_sha256`	Digest of the produced kernel bytes.
`substrate_digest`	Digest of the projected fold input, intended to exclude volatile envelope metadata.
`folded_nodes`	Reported number of context-deltas subsumed by the snapshot.
`completeness`	Shadow RFC 6962 Merkle root over semantic addresses of deltas marked `paramount`; not proof that their contents survive in the kernel.

Table 9: Context-snapshot fields and their bounded interpretation.

Let $S(C)$ denote the ordered substrate consumed by the reducer. Let $\mathrm{Fold}_{v,p}(S;\theta)$ be the kernel bytes produced by reducer version v, platform p, and configuration θ, including the size budget and selection policy.

Deterministic reconstruction. The reproducibility requirement compares independent runs rather than a digest with itself:

$$\mathrm{Fold}_{v,p_1}(S(C);\theta) = \mathrm{Fold}_{v,p_2}(S(C);\theta). \tag{4}$$

This requirement applies only under specified compatible runtime and serialization assumptions. A mismatch for identical substrate, version, and configuration refutes reproducibility under the tested conditions. A matching digest checks the produced bytes under the hash assumptions; it does not prove that the reducer preserved all relevant information. Cross-platform coverage remains *(unverified)*.

The codebase's "R2 idempotency" terminology refers here to reproducible reconstruction. A map from a chain to a kernel is not thereby an idempotent operator on a single domain.

Timestamp-free reconstruction. Excluding `id`, `ts`, and `audit` from semantic addresses does not by itself prove that the reducer ignores them. The required invariance is

$$S(C) = S(C') \implies \begin{aligned} &\mathsf{sub}(C) = \mathsf{sub}(C'), \\ &\mathrm{Fold}_{v,p}(S(C);\theta) = \mathrm{Fold}_{v,p}(S(C');\theta), \end{aligned} \tag{5}$$

where `sub` denotes the substrate digest. A metadata-only mutation test must preserve record resolution and chain order while checking that the actual substrate is unchanged. Merely observing equal digests is not a substitute for specifying the substrate.

This exclusion has a security consequence: a key holder can change an excluded timestamp and reauthenticate the record without changing its semantic address. A root over semantic addresses alone therefore does not prevent that replacement. A full-envelope, ordered checkpoint would be a separate commitment profile.

Completeness limitation. The shadow root commits to selected input addresses. It does not demonstrate that the corresponding reports, or their important facts, appear in the kernel. A reducer could omit a paramount report while retaining the correct shadow root. Detecting that omission requires an explicit retention check against the input inventory and a defined meaning of retention. Such a check is future work; the `completeness` field is not included as evidence of semantic completeness.

4.5 Two-axis verification

The local checks described in `lib/context_chain.py` concern chain authentication and referenced kernel blobs. We separate their evidence statuses rather than asserting a binary API return type.

Let $\alpha(C)$ report the required authentication and spine checks. Let $\beta(C)$ report presence and digest verification for every claimed `decision.kernel_blob_addr`. Each status is one of $\{\textsc{pass}, \textsc{fail}, \textsc{unknown}\}$. Missing keys or unavailable verification dependencies give UNKNOWN; a positively detected authenticator mismatch gives FAIL. A claimed blob that is confirmed missing or has a mismatching digest fails the blob check; an inaccessible store leaves the check unknown.

For the declared local profile, the reporting rule is

$$V(C) = \begin{cases} \textsc{broken} & \alpha(C) = \textsc{fail} \text{ or } \beta(C) = \textsc{fail}, \\ \textsc{verified} & \alpha(C) = \beta(C) = \textsc{pass}, \\ \textsc{unverifiable} & \text{otherwise.} \end{cases} \tag{6}$$

This is a specification of evidence interpretation, not a claim that the current function returns these exact tokens. If no blob is claimed, the blob-presence check has no obligations; this does not establish that a snapshot was required or that its fold was correct. In particular,

$$V(C) = \textsc{verified} \implies \alpha(C) = \beta(C) = \textsc{pass}. \tag{7}$$

Blob success cannot compensate for failed or missing authentication.

Known verifier failure and version boundary. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176, merged 2026-09-17T15:57Z, fixed the defect: "`-verify` never compared the record against the hash it sealed." GRIP#6193, merged 2026-09-17T17:28Z, replaced the single pass token

with an explicit list of what clean verification does not assert. These are council verifier changes, not evidence that all context-chain checks have been regression-tested.

The failure rules out treating a self-consistent adjacent hash as authenticated provenance. It also motivates positive, separately reported checks here. A clean local context result does not assert complete recording, external origin, pre-action timing, currentness, truthful reports, or complete composition.

External checkpoint evidence is separate. The public production anchor covers 501 exact ledger lines in chain order, not sorted semantic content addresses. Its root is

`0bba0792bf9f71caa815d0c42daf27f45baf07ef9749523be666f103a7f1bcb4`

and its manifest SHA-256 is

`5d5e8854f23eb50e583e02b380f21059651a3a4e99b4454cb221d355725a45b9.`

The OpenTimestamps proof attests this digest in Bitcoin blocks 956991 and 956992; a second stamp of the same digest completed in blocks 956948, 956963, and 956970. Since 2026-09-22, the proof, manifest, 501 RFC 6962 leaf hashes, and standard-library verifier are public at `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. The command `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately, and reports `VERIFIED` only when all three pass. Explorer-based checking retains the explorer's chain-data trust boundary.

The records themselves are not published because their subjects name people. The public bundle therefore does not let a reader independently recompute each leaf from its private record. This timestamp evidence is not public HMAC verification and does not establish coverage of the context chain, its snapshots, or its decision cross-links.

4.6 Warm-start: consuming a locally verified context report

The documented warm-start operation selects recorded next-step intent from the newest eligible node in the supplied chain (`MEMORY_CONTEXT.md`; `lib/context_chain.py`). Define

$$I(C) = \{i : \ n_i \text{ passes the required local checks and } \texttt{next\_step}(n_i) \neq \{\}\}.$$

The intended selection rule is

$$\texttt{resolve\_next\_up}(C) = \begin{cases} \texttt{next\_step}(n_{\max I(C)}) & I(C) \neq \varnothing, \\ \mathsf{None} & I(C) = \varnothing. \end{cases} \tag{8}$$

An unknown authentication result is not eligible. Selection must return an existing recorded value rather than synthesize an intent.

"Newest" means newest in the supplied, resolved chain, not the authoritative current head. Fallback from an invalid head can return stale intent even when the older record authenticates. Such fallback should be reported explicitly; the diagnostic behavior is not independently demonstrated here *(unverified)*. An application that requires current intent must additionally authenticate the head and apply a freshness policy. Local warm-start integrity is not permission to execute the returned action.

4.7 Summary of falsifiable commitments

Table 10 lists required tests and the boundary each addresses. It is a test specification, not a report that every case has been run. Authentication concerns supplied records; stronger claims require additional evidence.

Property	Evidence required	Counterexample or negative test
Payload integrity	Positive authentication under the declared profile	Change a covered field without reauthentication; acceptance refutes integrity.
Genesis structure	Resolved predecessor and depth checks	Use an incorrect predecessor or depth; structural acceptance must fail.
External origin	Authorized attestation, not a prefix	Manufacture a chain using the correct genesis constant; structure alone must not establish origin.
Cross-record binding	Required link and verified target	Remove the citation or target; complete composition must not pass.
Fold reproducibility	Fixed substrate, version, and configuration	Independent runs produce different kernel bytes.
Metadata invariance	Explicit projected substrate	Metadata-only changes preserving resolution alter the substrate or kernel.
Kernel retention	Inventory-to-output retention check	Omit a paramount report while keeping its shadow root; the root alone cannot detect this.
Positive local verification	Authentication and blob checks both pass	Missing prerequisites or a corrupt claimed blob still produce a positive local result.
Warm-start selection	Eligible recorded intent	Return an unauthenticated or invented next step; stale fallback must not imply currentness.

Table 10: Required tests and evidence boundaries for the memory–context chain.

Revision provenance. This section was revised for the paper authored by V» (Laurie Scheepers), with AI assistance from Claude Fable 5.1 and Astra (gpt-6-astra).

5 Claim provenance and the three-leg composition

The decision forest (Section 3) records supplied decision reports; the belief chain (Section 4) records supplied context reports. Neither establishes that the reports exhaust the agent's actions or faithfully describe its internal state. The third leg, *claim provenance*, checks explicitly enumerated quotations against supplied, hash-pinned source snapshots. It does not establish source authenticity, semantic support, factual truth, or exhaustive coverage of assertions in the deliverable.

The implementation references for this leg are `lib/prove_it.py`, `lib/legal_receipt.py`, and `bin/grasp-verify-receipt`. These references identify implementation locations, not a proof of implementation correctness. We distinguish the described consistency checks from authentication and composition requirements that are not demonstrated by the receipt schema below.

5.1 The receipt

A receipt associates a deliverable digest with source digests, citation results, and optional references to decision and context records. The described fields are a `deliverable` block containing `path`, `sha256`, and `bytes`; a `sources` map containing labels, digests, and character counts; citation identifiers, source identifiers, statuses, and span offsets; summary fields; and a `chain` block. Listing 3 illustrates this structure. It is not a released receipt or evidence of successful verification. In particular, it contains no signature block.

```
{
  "deliverable": {
    "path": "<doc>", "sha256": "<hex>", "bytes": "<count>"
  },
  "sources": {
    "<sid>": {
      "label": "<source snapshot>",
      "sha256": "<hex>", "chars": "<count>"
    }
  },
  "citations": [
    {
      "id": "<citation id>", "source_id": "<sid>",
      "status": "<match status>",
      "start": "<offset>", "end": "<offset>"
    }
  ],
  "grounding_rate": "<reported rate>",
  "tally": {
    "verified": "<count>",
    "fuzzy": "<count>",
    "not_found": "<count>"
  },
  "chain": {
    "recorded": "<boolean>",
    "idr_addr": "<decision reference>",
    "memory_head": "<context reference>"
  },
  "filed_safe": "<policy result>",
  "council": { "ran": "<boolean>" }
```

```
}
```

Listing 3: Schematic receipt fields; placeholders are not verification evidence. No authentication envelope is specified here.

The quotation text comes from the citation specification used for verification; it is not present in the illustrated citation entry. Re-verification therefore requires that specification as well as the receipt, deliverable, and source snapshots. A source digest pins the supplied source bytes. A citation span identifies a range within the source, not within the deliverable.

The offset unit and decoding rules must be explicit. The presence of a `chars` count does not justify calling `start` and `end` byte offsets. This section does not establish whether the implementation uses byte or decoded-character offsets *(unverified)*. An interoperable profile must specify encoding, offset units, range bounds, and any normalization, and must test non-ASCII sources. Without those rules, two implementations can agree on the source digest but disagree on its quoted span.

An exact-match result means only that the submitted quotation matches its specified range in the supplied source under the verifier's text rules. An invented quotation inserted into an attacker-authored source can satisfy that condition. A genuine quotation can also be irrelevant to the associated claim. Neither case requires a hash collision.

For reporting purposes, the denominator of an exact-match rate must be the number of enumerated citations, not the number of sentences or factual assertions in the deliverable. The existing `grounding_rate` field must not be interpreted as exhaustive claim coverage; its precise implementation formula is *(unverified)* here. An empty citation inventory supplies no evidence of coverage. Likewise, `filed_safe` is a policy-result field, not a guarantee of legal sufficiency or factual correctness.

Negative tests. With the receipt and citation specification held fixed, changing a cited source so that its digest no longer matches, or changing a quoted span so that it no longer matches the specification, must prevent an exact-match acceptance. These tests concern consistency with retained expected values. They do not test resistance to replacing the source, specification, receipt, and expected digests together.

5.2 Independent re-verification

The local re-verification tool `tools/grasp-verify-receipt` in the public GRASP repository performs four deterministic checks — deliverable, sources, citations, tally — each failing loudly with exit status 1:

1. **Deliverable consistency:** recompute the deliverable digest and compare it with `deliverable.sha256`.
2. **Source consistency:** recompute source digests and the recorded source measurements.
3. **Quotation consistency:** compare each specification quotation with the corresponding source range. A `not_found` result does not satisfy exact matching.
4. **Summary consistency:** recompute citation counts and compare them with the receipt's tally.

These checks can be performed locally on supplied files. They establish consistency of those files with one another, not independent authenticity of their contents. The four checks do not, by themselves, authenticate the receipt, bind the citation specification, verify the chain references, or establish when the files existed.

The known verifier failure. The distinction between consistency and authentication is not hypothetical. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176 fixed the defect: "`-verify` never compared the record against the hash it sealed" (merged 2026-09-17T15:57Z). GRIP#6193 replaced the single pass token with an explicit list of what clean verification does *not* assert (merged 2026-09-17T17:28Z).[22] This was a council-verifier failure, not evidence of an identical defect in the receipt verifier. It nevertheless refutes the general argument that recomputing a digest stored beside editable content establishes tamper evidence against replacement of both.

For this section, `HASH_CONSISTENT` denotes only successful consistency checks against supplied expected digests. `AUTHENTICATED` additionally requires a valid authenticator over a defined message and an accepted key. `EXTERNALLY_ANCHORED` additionally requires a verified external commitment covering the relevant bytes. These are distinct evidence predicates, not a claim that the receipt CLI currently emits these labels. Missing material must leave the affected predicate `UNVERIFIABLE`, rather than count as a successful check.

Authentication is profile-specific. HMAC-SHA256 verification requires the shared secret; every holder of that secret can also forge authenticators. Only records actually signed under the Ed25519 or ML-DSA-65 public-signature profiles can be authenticated using public verification material, with independently authenticated public keys. The four file-consistency checks above can be public even when authentication of an associated HMAC record cannot be.

Authentication work still required. A complete receipt profile must define the exact authenticated bytes, canonicalization, algorithm selection, and key identification. Its protected message must include the deliverable and source digests, citation specification digest, citation results, derived summaries, and every chain reference. Successful signature verification must not substitute for recomputing those derived results. A published receipt envelope and mutation corpus demonstrating these bindings remain future work; full receipt authentication is *(unverified)* in this section.

5.3 A monotone advisory layer

The optional model-based layer in `lib/prove_it_council.py` is intended to flag quotations that are present but do not support their associated claims. Its policy is monotone toward rejection: it may add a warning or reject an otherwise matching citation, but it must not turn a failed deterministic match into a successful one. This is a policy constraint, not a truth guarantee. A receipt with `council.ran=false` reports no advisory assessment. The schematic listing is not evidence that any particular assessment ran.

The required regression test is to submit a `not_found` citation with a favourable model judgement and confirm that deterministic acceptance remains false. Conversely, a real but irrelevant quotation should be retained as an exact match while its support assessment is reported separately. Published results for these receipt-specific tests are *(unverified)* here. Provider separation alone does not establish evaluator independence or accuracy (Section 6).

A separate pre-hoc watcher supplies no positive predictive evidence for this advisory layer. Its 2026-09-10 measurement used seed 0, five stratified folds, a 50-event window, and 76 paired rows with a 13.2% base rate. No predictor's precision interval cleared that base rate. The watcher remains `WARN`-only and its hook is unbound.[23] These are watcher results, not a benchmark of citation-support assessment.

[22] Repair references: GRIP#6176 and GRIP#6193; incident and merge details verified by the host session on 2026-09-22.

[23] Watcher measurement and deployment status verified by the host session on 2026-09-22.

Nor does this composition establish a commercial moat. Four non-Anthropic councils, with zero Anthropic calls, judged the integrated system "a bundle of commodities with a nice story"; the supplied outcome references are seals `7dc01f0b739be243` and `9fb9b0fc0f2d4309`. Individual mechanisms are shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI. Asqav, under the MIT licence, signs with the same ML-DSA-65 and ships RFC 3161 timestamps and RFC 8785 canonicalisation. The only surveyed differentiator is shipping attacks alongside the claims; that comparison rests on reading four vendors' documentation, not running their products.[24] EU AI Act Article 12, in force from 2 August 2026, creates compulsory demand for automatic record-keeping. It does not establish that this receipt design satisfies a deployment's legal obligations.

5.4 The composition

The intended composition associates three supplied artifacts: a decision record, a context report, and a claim receipt. The fields `chain.idr_addr` and `chain.memory_head` name targets; their presence is not proof that the targets exist, authenticate successfully, or refer to the same decision. Optional, absent, or unresolved references may remain compatible with local record integrity, but they do not satisfy complete composition.

A `COMPOSITION_COMPLETE` predicate would require:

1. positive authentication of each required artifact under its declared profile;
2. authentication of the receipt's references and citation-specification digest;
3. resolution and verification of the referenced decision and context records, including their claimed relationship;
4. successful deliverable, source, citation, and summary checks; and
5. an explicit coverage statement identifying the enumerated claims and citations, without treating that inventory as exhaustive unless separately checked against the deliverable.

This is a required composition contract, not a description of the four-check receipt verifier. A public three-leg bundle and a checker demonstrating this contract are not supplied here *(unverified)*.

Temporal evidence and commitment scope. A signed timestamp authenticates a claimed time, not its wall-clock accuracy. Before RFC 3161 trusted timestamps were added, the anti-HARKing property bound only a third party without the signing key; it was not established against the writer. RFC 3161 support was added in GRIP#6190, "the external time anchor we lacked," merged 2026-09-17T17:22Z, in `lib/rfc3161_anchor.py` and `lib/continuity_chain.py`.[25] Its availability does not timestamp every receipt retroactively. A writer-resistant pre-action claim requires a verified timestamp covering the commitment and independently evidenced ordering relative to the relevant action or outcome.

The available public Bitcoin evidence has a narrower, explicit scope. It covers a production decision-chain prefix of 501 records with root

```
0bba0792bf9f71caa815d0c42daf27f45baf07ef9749523be666f103a7f1bcb4
```

and manifest SHA-256

[24] Council outcomes and capability survey verified by the host session on 2026-09-22.
[25] GRIP#6190 and module locations verified by the host session on 2026-09-22.

`5d5e8854f23eb50e583e02b380f21059651a3a4e99b4454cb221d355725a45b9.`

Its OpenTimestamps proof attests the root in Bitcoin blocks 956991 and 956992, whose Merkle roots begin `1b42ef4d` and `b3f15c01`, respectively. A second stamp of the same digest completed in blocks 956948, 956963, and 956970. The proof, manifest, 501 RFC 6962 leaf hashes, and a standard-library verifier have been public since 2026-09-22 at `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. The command `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately and reports `VERIFIED` only when all three pass.

Those leaves commit to the exact ledger lines in chain order, not sorted content addresses. The records themselves are not public because their subjects name people. Public checking of the supplied leaf hashes and timestamp proof therefore does not let readers recompute those leaves from undisclosed record bodies. Explorer-based chain checking also depends on the explorer's supplied Bitcoin data. This anchor does not demonstrate receipt membership, complete three-leg composition, continuous recording, or a current authenticated head. It is distinct from a semantic forest root that excludes fields from its address domain.

Binding is not truth or causation. Even a fully authenticated composition would establish only the bindings actually checked. It would not prove that a context report faithfully describes internal model state, that the reported context caused the decision, that a council roster independently proves provider participation, or that the recorded events are complete. Quote matching is not source authenticity or semantic support. Historical inclusion is not currentness. A verifier can examine authenticated recorded claims; it cannot infer "exactly what happened" from those records alone.

Composition tests still required. Mutating either chain reference, the specification digest, a referenced target, or a derived result must prevent complete acceptance. Removing a required artifact, key, or verification dependency must produce an unverifiable or incomplete result, never success. Tests must also cover joint replacement of payloads and adjacent hashes, re-signing by a key holder, stale valid bundles, omitted claims, fabricated source snapshots, and genuine but irrelevant quotations. These are proposed tests, not reported successful regressions. Passing local receipt consistency after a cross-link mutation would demonstrate the limit of that local check, not complete composition.

Revision provenance. This section is authored by V» (Laurie Scheepers), with AI assistance from Claude Fable 5.1 and Astra (gpt-6-astra). Assistance identifies revision provenance, not an independent attestation of the section's claims.

6 The exogenous-anchor law

Tamper evidence does not establish that a recorded decision was correct, that its evaluator was independent, or that the record preceded the action it describes. We retain "exogenous-anchor law" as the name of a governance policy, not as a theorem about learning or convergence. The policy is described in `rules/exogenous-anchor-law.md` and `rules/scientific-method.md`; a cross-provider implementation is `bin/grip-sovereign-council`. This section separates the policy from the checks and evidence that currently support it.

6.1 The law

> An improvement loop claiming external acceptance must obtain an evaluation from a principal whose acceptance authority the producer cannot grant to itself. Self-assessment may inform diagnosis, but does not by itself satisfy this external-acceptance policy.

Let a producer submit an artefact x, a self-assessment $s(x)$, and an evaluation $E(x)$. Policy conformance requires an authorized external evaluation of the specified artefact under the applicable policy. It does not imply that $E(x)$ is correct. Provider separation, human review, CI, and preregistration are candidate controls; none alone establishes evaluator independence or prevents manipulation. In particular, producers can overfit tests, select favourable reviewers, repeat queries, or withhold unfavourable results.

The rule that self-telemetry may lower but not raise acceptance confidence is a conservative governance choice, not a general claim that self-measurement has no information value. Likewise, deterministic reconstruction of recorded data checks record consistency; it is not an independent evaluation of the original decision.

Two questions must therefore remain separate. *Policy conformance* asks whether the required external evidence was obtained and authenticated. *Effectiveness* asks whether that policy reduces failures enough to justify its cost. The latter remains *(unverified)* as a general claim. A future comparison should prespecify workloads, failure labels, evaluator access, repeat-query budgets, and total costs, and compare self-evaluation with external evaluation under those conditions. No universal improvement guarantee follows from the policy definition.

6.2 Admissible anchors

The four root prefixes accepted by the decision forest (Section 3.4) classify claimed evidence:

`ci:` A test result for a specified artefact and test suite. Mutation testing can expose weak tests, but passing establishes only the properties exercised by those tests.

`human:` A claimed human authorization, such as a commit or merge. Human authorship and authority require evidence beyond the identifier.

`council:` A recorded evaluation by a model panel and judge. Distinct providers reduce one form of self-evaluation; they do not establish statistical independence, truthful participation claims, or immunity to prompting and selection.

`hypo:` A hypothesis commitment with a stated outcome criterion and deadline. Its temporal force depends on independently evidenced ordering.

Prefix validation is a namespace check, not authentication. An agent can write `human:forged` or `council:invented`; it can also copy a public genesis identifier. Rejecting `self:` therefore does not demonstrate external control of accepted roots.

Authenticated-origin acceptance would additionally require an attestation binding an issuer, artefact digest, policy and version, result, freshness information, and signature. Authorized issuer keys must be selected outside producer-controlled configuration. Enforcement of this complete attestation requirement is *(unverified)* here. Until demonstrated, external authorization is a deployment assumption rather than a construction guarantee.

Authentication profiles. A default HMAC record can be verified only by a holder of the shared secret; that holder can also forge records. Public verification applies only to records actually signed under the Ed25519 or ML-DSA-65 profiles and checked against independently authenticated public keys. Neither profile proves that a claimed human or model performed an evaluation. An external timestamp supplies a different kind of evidence: it binds a commitment's existence, not the evaluator's authority or the truth of its conclusion.

Pre-action commitments. Before the external time anchor was added, the anti-HARKing property bound only a third party without the signing key. It was not established against the writer: a key holder could observe an outcome, create a hypothesis afterwards, and sign an earlier claimed timestamp.

RFC 3161 trusted timestamps were added in GRIP#6190, "the external time anchor we lacked," merged on 2026-09-17 at 17:22Z (`lib/rfc3161_anchor.py` and `lib/continuity_chain.py`). This supplies an external timestamp mechanism. Writer-resistant pre-action ordering still requires a timestamp covering the actual commitment, obtained before an independently evidenced action or outcome boundary. A signature on a claimed local time is insufficient. The protocol must also account for withheld commitments, replacements, and missed deadlines; comprehensive enforcement of those conditions is *(unverified)* here.

Historical storage commitment. The production Bitcoin anchor is separate from evaluator authorization and from the forest's semantic-address commitment. It covers 501 exact ledger lines in chain order, not sorted content addresses. Its decision-chain root is

`0bba0792bf9f71caa815d0c42daf27f45`
`baf07ef9749523be666f103a7f1bcb4`,

and the manifest SHA-256 is

`5d5e8854f23eb50e583e02b380f210596`
`51a3a4e99b4454cb221d355725a45b9`.

Each displayed digest is the concatenation of its two lines. The OpenTimestamps proof attests this root in Bitcoin blocks 956991 and 956992, whose Merkle roots begin `1b42ef4d` and `b3f15c01`, respectively. A second stamp of the same digest completed in blocks 956948, 956963, and 956970.

The proof, manifest, 501 RFC 6962 leaf hashes, and standard-library verifier have been public since 2026-09-22 at `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. There, `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately, and reports `VERIFIED` only when all three pass. Explorer-based chain checking retains trust in the explorer's chain data; it is not equivalent to independent full-node validation. The underlying records are not published because their subjects name people. The public bundle therefore permits checking the supplied leaf-hash commitment and timestamp proof, not inspecting the private records or independently recomputing their leaf hashes from their bodies.

This historical commitment does not establish continuous anchoring, the current head, evaluator independence, or coverage of later council records, context reports, and claim receipts. Nor does it repair exclusions in a different commitment domain: a root over projected semantic bodies does not bind excluded timestamps or signatures merely because that root is externally timestamped.

Required negative tests. Authenticated-origin acceptance should reject an allowed-prefix root without an authorized attestation and a manufactured chain using the correct genesis constant. A

temporal acceptance test should let a writer learn the outcome, backdate and re-sign a commitment, and attempt acceptance. These are required tests for the stronger claims, not results established by prefix validation.

6.3 A sovereign instantiation

The name "sovereign council" denotes the cross-provider workflow implemented by `bin/grip-sovereign-council`: a panel answers a question, a separate judge ranks and synthesizes the answers, and the workflow seals a payload containing the prompt, answers, and verdict. It also attempts to append a signed decision record. The evaluated path can complete with a hash-only result when the IDR library is unavailable; mandatory authenticated recording is therefore not a demonstrated property of that path.

A content hash detects a payload change relative to an independently retained expected digest. It does not resist replacement of both the payload and an adjacent editable digest. Judge/panel separation is similarly narrower than authenticated independence: a sealed roster commits to recorded identities, not independently authenticated provider execution.

Observed verifier failure and repair. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176, "`-verify` never compared the record against the hash it sealed," fixed the hole and was merged on 2026-09-17 at 15:57Z. GRIP#6193, merged that day at 17:28Z, replaced the single pass token with an explicit list of what a clean verification does *not* assert. These changes establish a version boundary; the earlier `PROVEN` result cannot support the guarantees claimed here.

Verification reports must distinguish hash consistency, record authentication, external anchoring, complete cross-record binding, and freshness. A missing key, library, dependency, or referenced record is not positive verification of the corresponding property. In particular, HMAC authentication is unavailable to a public verifier without the secret. The Bitcoin bundle's `VERIFIED` result concerns its three named checks, not this wider set of council guarantees.

The demonstrated replacement attack is stronger than a fixed-digest single-byte mutation. Further regression coverage should include replacement of the payload and digest together, substitution of the IDR pointer, re-signing by the writer, missing verification dependencies, and replay of an older valid bundle. Completion of that entire corpus is *(unverified)* here.

Negative commercial evaluation. Four non-Anthropic councils, with zero Anthropic calls, judged the integrated system "a bundle of commodities with a nice story"; recorded seals include `7dc01f0b739be243` and `9fb9b0fc0f2d4309`. This refutes the commercial differentiation claim offered to those councils; it is not evidence for a moat.

The individual mechanisms are already shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI. Asqav is MIT-licensed, uses the same ML-DSA-65 scheme, and ships RFC 3161 timestamps and RFC 8785 canonicalisation. The only surveyed differentiator is shipping attacks alongside the claim. That comparison rests on reading four vendors' documentation, not running their products, and establishes neither exclusivity nor superior performance. EU AI Act Article 12, in force from 2 August 2026, creates a forced buyer for automatic record-keeping; it does not require this implementation or establish its commercial advantage.

Negative predictive evidence. External evaluation also does not justify treating an unvalidated predictor as an acceptance gate. The pre-hoc watcher measurement of 2026-09-10 used seed 0, five stratified folds, a 50-event window, and 76 paired rows, with a base rate of 13.2%. No predictor's

precision interval cleared that base rate. The watcher therefore remains `WARN`-only and its hook is unbound. This measurement does not support a claim of useful predictive discrimination.

The patch history, council outcomes, and watcher measurement reported above are the author-record evidence verified on 2026-09-22. They should not be confused with a publicly reproducible release of every underlying artefact; the public anchor bundle has the narrower scope stated above.

6.4 The bootstrap exception

A change that creates or widens acceptance authority must not be ratified using the authority it introduces. Under this policy, such changes require human authorization through the previously authorized procedure. Calling a root `human:` does not satisfy that requirement: the authorization must be authenticated and bound to the proposed change.

This is an access-control invariant, not an exception permitting self-approval. Its direct failure condition is acceptance of an authority-widening change solely through the new authority, without the required prior authorization. A future test should attempt that transition, including replacement of authorization configuration and forged human attestations. Complete enforcement across all change paths remains *(unverified)*; comparative claims about the resulting system's "trustworthiness" are not substitutes for this test.

Revision provenance. This paper is authored by V» (Laurie Scheepers), with AI assistance. Claude Fable 5.1 and Astra (gpt-6-astra) assisted the revision. Their assistance is not an external authorization or an independent verification of the claims above.

7 Verification, Replay Determinism, and Selective Disclosure

Verification checks specified properties of supplied records; it does not establish a complete or truthful account of an agent's conduct. This section distinguishes three checks:

1. **Record authentication**: checking signed bytes and recorded predecessor links under a declared cryptographic profile;
2. **Deterministic record reconstruction**: reconstructing a supplied trail without executing a language model; and
3. **Membership verification**: checking inclusion in a particular Merkle commitment without transmitting other record bodies.

Their conclusions depend on the covered bytes, available verification material, trusted keys, and checkpoint provenance. None alone establishes recording completeness, truthful context reports, authenticated external evaluation, pre-action commitment, or currentness.

Implementation references below identify files or known symbols, not source line ranges. A reproducibility bundle must additionally identify the actual revision, envelope version, serialization profile, and fixtures. This section does not impose a competing envelope-version declaration on records produced under other versions.

Known failure and revision boundary. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176 fixed the defect: "`-verify` never compared the record against the hash it sealed" (merged 2026-09-17T15:57Z). GRIP#6193 replaced the single pass token with an explicit list of what a clean

verification does *not* assert (merged 2026-09-17T17:28Z). These are recorded failures and fixes, not hypothetical cryptographic attacks (author's host verification record, 2026-09-22). The pre-fix result refutes the earlier verifier guarantee. The identified fix does not establish correctness of every verification path; a released, revision-pinned regression corpus is still needed to support that broader claim *(unverified)*.

7.1 Content addressing

Definition 7.1 (Semantic content address)**.** Let π be the semantic projection of an IDR body: it removes $\{\texttt{id}, \texttt{ts}, \texttt{audit}\}$ and applies the profile's normalization of optional fields, including omission of a null `decision_anatomy`. Let J_v denote the serialization used by the declared implementation profile v. Then

$$\texttt{content\_addr}_v(b) = \texttt{"sha256:"} \parallel \mathrm{hex}(\mathsf{SHA256}(J_v(\pi(b)))) \,.$$

The implementation and specification references are `lib/precog/idr.py`, `lib/idr_forest.py`, and `IDR.md`.

This is a name for projected content, not for the complete signed envelope. Bodies differing only in excluded fields have the same address under the same profile. Conversely, equal meaning in ordinary language does not imply equal projected bytes.

Serialization is part of the protocol. Python JSON options alone do not specify a cross-language canonicalization standard. Independent implementations need normative rules for numbers, Unicode, duplicate keys, non-finite values, separators, absent and null fields, and digest encoding. Cross-language address equivalence without those rules and test vectors remains *(unverified)*. RFC 8785 is a possible future normative profile; it must not be silently substituted for the bytes used by existing records.

Commitment domains. A semantic address, a signature, and an external timestamp cover different objects:

- A semantic address covers $J_v(\pi(b))$, not excluded envelope fields.
- An authenticator covers the signing message defined by its profile. The signed-body implementation is `lib/idr_forest.py::signed_body`; its projection must not be assumed identical to π.
- A semantic forest root covers the sorted semantic addresses, not complete signed envelopes or their original enumeration order.
- The public production ledger checkpoint described below covers exact ledger lines in chain order through RFC 6962 leaf hashes.
- An external timestamp binds its submitted digest. It does not expand that digest's commitment domain.

A writer can change an excluded timestamp and re-sign a record without changing its semantic address or a root constructed from that address. This is an expected consequence of the projection, not a hash collision. A separate, domain-separated full-envelope checkpoint is proposed for uses requiring those fields to be bound; deployment across the decision, context, and receipt records is *(unverified)*.

> *Tests.* Bodies identical after π must receive the same address under one profile. Changing an excluded timestamp and re-signing must leave the semantic root unchanged. A claim that this root detects that change is refuted by this counterexample.

Verdict	Required interpretation
VERIFIED	All required authentication and structural checks positively pass for the declared record set and profile. This is a local integrity result.
BROKEN	An available required check detects an invalid authenticator, malformed required structure, or inconsistent required link.
DEGRADED	Required keys, dependencies, records, or supported authentication mechanisms are unavailable, so the requested integrity conclusion is not established.

Table 11: Acceptance semantics for local integrity. Missing evidence is not a pass; a detected failure remains a failure even if other checks are unavailable.

7.2 Tamper-evidence via HMAC chaining

Definition 7.2 (Profile-specific authentication)**.** Let $M_v(n)$ be the exact signing message for node n under profile v, including any predecessor reference covered by that profile. Verification checks the stored authenticator against $M_v(n)$ using the required key material. This notation deliberately does not choose between signing a body and signing its digest: that choice, including digest encoding, must be fixed by the profile rather than by a competing equation in this section.

For the HMAC-SHA256 profile, verification requires the shared secret. Every holder of that secret can also create authenticators; this is not public verification or attribution to a unique signer. Only the Ed25519 and ML-DSA-65 public-signature profiles support verification using public keys, and only for records actually signed under those profiles. The keys must be authenticated independently of the supplied record. A combined profile additionally needs explicit component-verification and downgrade rules; an unavailable component must not silently weaken the selected profile.

Proposition 7.1 (Conditional detection of signed-message alteration)**.** *If verification material is available, the expected key and authenticator are fixed, and a mutation changes the authenticated message $M_v(n)$, authentication rejects that mutation except under the selected scheme's security assumptions. The claim does not cover a key holder replacing the message and authenticator together.*

This proposition concerns the authenticated message, not every byte of a serialized file. Whitespace changes normalized away before signing may not change that message. A failure may also arise from a corrupted authenticator, a wrong trusted key, or malformed structure; it does not by itself identify who altered which field. Verification failures can result from implementation and binding defects, as GRIP#6176 demonstrates, without breaking SHA-256.

Definition 7.3 (Integrity-result interpretation)**.** The forest verifier is `lib/idr_forest.py::verify_chain_integrity`. Table 11 states the acceptance semantics required in this paper for its tri-state vocabulary. It is not evidence that every caller and failure path already implements these semantics.

For reporting, we distinguish `HASH_CONSISTENT`, `AUTHENTICATED`, `EXTERNALLY_ANCHORED`, `COMPOSITION_COMPLETE`, and `CURRENT`. These are separate predicates, not interchangeable success tokens or a claim about existing API return values. Missing prerequisites make the affected predicate `UNVERIFIABLE`. In particular, "not BROKEN" is insufficient for authentication. A hash recomputed from an editable payload and compared with an adjacent editable hash establishes only consistency of the supplied pair.

> *Required negative tests.* Alter an authenticated message while holding its authenticator fixed; corrupt the authenticator; supply a wrong pinned key; remove required keys or dependencies;

change the scheme identifier; and strip a required signature component. None may produce a positive result for the original profile. Replace a council payload and its stored hash together: hash consistency must not be reported as authenticated or externally anchored provenance.

7.3 Deterministic replay

Here "replay" means *record reconstruction*, not re-execution of a decision or recovery of an actual reasoning process. The following `ReplayResult` listing is a proposed reconstruction contract, not a verified API: conformance requires a root-to-target sequence and digest, no fresh model call, and strict rejection of inputs already classified BROKEN; whether `lib/idr_forest.py` implements this contract is *(unverified)* — the reviewed revision exposes no `ReplayResult` symbol:

```
ReplayResult:
  sequence : recorded root-to-target trail
  verdict : local integrity result
  replay_digest : SHA-256 of the serialized sequence
```

Proposition 7.2 (Conditional reconstruction determinism)**.** *For identical supplied records, target selection, reconstruction version, serialization rules, and verification inputs, independent executions of a deterministic reconstruction must return identical sequence bytes and* `replay_digest` *values.*

The reconstruction described in `lib/idr_forest.py` does not call a language model. Model outputs may be stored as input data, but no fresh generation is needed. Agreement across unspecified platforms or serializer versions is not established by this design alone *(unverified)*. A digest mismatch can indicate different inputs, target selection, serialization, version behavior, or corruption; it does not identify the cause.

The proposed strict wrapper refuses a forest classified BROKEN; its implementation is *(unverified)* here. This rejection rule is narrower than a guarantee against laundering: a forged but accepted record, an unanchored replacement, or unavailable verification material remains a separate problem. Reconstruction of degraded data, if offered for diagnosis, must preserve its non-authenticated status.

> *Falsified if* identical declared inputs produce different reconstruction bytes or digests, reconstruction invokes a language model, or the strict wrapper returns an integrity-success result for input already classified BROKEN. These tests do not establish completeness or truth of the reconstructed account.

Reconstruction is not freshness verification. An old signed record may authenticate correctly. A signed predecessor reference describes a supplied linkage, and a signed timestamp authenticates a claimed time; neither establishes that the record is the current head. Membership in an anchored root likewise proves historical inclusion, not absence of later records.

A `CURRENT` result would require an authenticated checkpoint naming the log, head and sequence, evidence of extension, a freshness policy, and a way to discover sufficiently recent checkpoints and handle equivocation. Those checks are not supplied by record reconstruction. Without them, currentness is unknown. A future freshness test must serve stale checkpoints, inconsistent heads, and a valid proof for a superseded record, and require rejection or an explicit unknown result rather than currentness acceptance.

7.4 Selective disclosure via a Merkle commitment

Definition 7.4 (Semantic forest commitment)**.** The forest construction described in `IDR.md` and `lib/idr_forest.py` builds an RFC 6962 Merkle tree over sorted content addresses. Subject to the

hash assumptions and fixed leaf encoding, it binds the committed multiset of projected bodies. An inclusion proof supplies an authentication path of $O(\log N)$ hashes for a tree with N leaves.

Changing the committed multiset is expected to change the root under the hash assumptions. Changing only excluded fields, or reordering the forest before sorting, need not do so. Consequently, this root is not a commitment to all envelope bytes or to chronological order.

An inclusion proof avoids transmitting other record bodies. It reveals authentication-path hashes and structural metadata, including position or tree size when supplied by the protocol. Unsalted hashes of predictable records may permit dictionary testing. This is selective transmission, not a hiding commitment or a zero-knowledge guarantee. Membership is meaningful only relative to a separately authenticated expected root; an attacker can otherwise replace the leaf, proof, and root together.

Public production checkpoint: a different leaf profile. The production decision-chain checkpoint uses exact ledger lines in chain order, *not sorted content addresses*. Its 501-record root is

`0bba0792bf9f71caa815d0c42daf27f45`
`baf07ef9749523be666f103a7f1bcb4`

and the manifest SHA-256 is

`5d5e8854f23eb50e583e02b380f210596`
`51a3a4e99b4454cb221d355725a45b9.`

Line breaks in these displays are typographic only. An OpenTimestamps proof attests this root in Bitcoin blocks 956991 (Merkle-root prefix `1b42ef4d`) and 956992 (prefix `b3f15c01`). A second stamp of the same digest completed in blocks 956948, 956963, and 956970.

The proof, manifest, 501 RFC 6962 leaf hashes, and a standard-library verifier have been public since 2026-09-22 at `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. In that bundle, `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately and reports `VERIFIED` only when all three pass. These labels concern this anchor bundle, not the forest's local integrity verdict. Explorer mode depends on the supplied explorer chain data; it is not equivalent to independently validating Bitcoin consensus and chain selection.

The records themselves are not published because their subjects name people. The public leaf hashes permit reconstruction of the committed root and verification of its timestamp proof. Without the ledger bytes, they do not permit the public to recompute each leaf from its record or authenticate the records' contents. This checkpoint establishes neither continuous anchoring nor coverage of later council records, context chains, receipts, or this paper.

> *Tests.* Against a fixed authenticated root, change a committed leaf, its position where order is part of the profile, or an authentication-path hash; inclusion must fail. Excluded-field changes are not falsifiers of the semantic profile. Proof inspection must report metadata leakage rather than claim that no information about other records is revealed.

7.5 The trust boundary: exogenous anchoring

An authority label, an evaluator's attestation, and an external timestamp are distinct evidence types. The forest recognizes root prefixes

`("ci:", "human:", "council:", "hypo:")`

and stores an `anchor_id` (`lib/idr_forest.py`). These fields classify a *claimed* origin and record a structural reference. An agent can write a permitted prefix; its presence does not authenticate an external principal, evaluator independence, or authorization. Likewise, the public genesis identifier `council:context-chain-genesis` does not prevent a writer from manufacturing a new chain rooted at that identifier.

Authenticated external evaluation would require an attestation binding an authorized issuer, artifact digest, policy and version, result, and relevant freshness information. Issuer keys must be pinned outside agent-controlled configuration. Such checks must be demonstrated separately; prefix validation is not their substitute *(unverified)*.

External time and pre-action commitments. RFC 3161 trusted timestamps were added in GRIP#6190, "the external time anchor we lacked" (merged 2026-09-17T17:22Z), in `lib/rfc3161_anchor.py` and `lib/continuity_chain.py` (author's host verification record, 2026-09-22). Before that addition, the anti-HARKing property bound only a third party without the signing key; it was *not* established against the writer.

A validated timestamp binds a digest to the timestamp authority's time assertion under its trust policy. Establishing commitment *before an action* additionally requires independently evidenced action or outcome-availability timing. Addition of timestamp support does not show that every record obtained a pre-action timestamp. Signed local times alone do not provide this ordering, under either the HMAC or public-signature profile.

> *Required tests.* Submit fabricated `human:` and `council:` roots without authorized attestations, and manufacture a chain using the published genesis identifier. They must not receive an authenticated-origin result. Let the writer learn an outcome, backdate and re-sign a hypothesis, and request pre-action acceptance: without independently verified ordering evidence, the result must remain unestablished.

7.6 Memory–context chain: dual-axis verification and replay

The memory–context chain stores supplied context reports. Authentication does not establish that a report faithfully describes internal model state, caused a decision, or was captured contemporaneously with an external action.

The documented context-verification design checks both spine authentication and availability of referenced context blobs (`MEMORY_CONTEXT.md`). A positive result requires both. Presence alone does not establish blob integrity: where a content digest is specified, the retrieved bytes must also match it. The exact context API return type and its mapping to the forest's tri-state vocabulary need a revision-pinned conformance test *(unverified)*; this section does not assume identical return types.

For an HMAC-authenticated spine, checking authentication requires the secret. Public verification applies only if the actual context records use an Ed25519 or ML-DSA-65 profile and the verifier has authenticated public keys. Context reconstruction has the same conditional determinism as decision reconstruction: fixed inputs and serialization must give the same recorded sequence, without a fresh model call.

Local context integrity is separate from complete decision–context–receipt composition. An absent or unresolved cross-reference cannot satisfy `COMPOSITION_COMPLETE`, even if the local chain remains valid. Complete composition needs required links, authenticated targets, and checks that the targets describe the same decision. Nor does local integrity prove mandatory recording: the evaluated council path permits hash-only completion when IDR recording is unavailable. Enforcement of paired recording is therefore a deployment requirement, not an established guarantee.

Predicate	Scope	Counterexample or required negative test
Authentication	Declared signing message, profile, and trusted key	Alter the message with fixed authenticator; remove required verification material.
Reconstruction	Fixed records, target, version, and serialization	Obtain different sequence bytes or digests from identical declared inputs.
Strict rejection	Inputs already classified BROKEN	Obtain an integrity-success result from the strict wrapper.
Merkle membership	Exact leaf profile and fixed authenticated root	Accept a changed committed leaf or invalid authentication path.
Context availability	Authenticated spine and required content-addressed blobs	Accept missing blobs or digest-mismatched bytes.
Complete composition	Required authenticated cross-links and targets	Accept an omitted paired record or redirected required link.
External origin and time	Authorized attestation and independently evidenced ordering	Accept a permitted prefix alone, or a writer-backdated hypothesis.
Spec–code parity	The particular documented example	Change its structure or documented address incompatibly while its test passes.

Table 12: Scoped checks and required counterexample tests. A positive local check must not be promoted into a broader unsupported guarantee.

> *Required tests.* Remove a referenced blob, substitute bytes with a different digest, break spine authentication, omit either required paired record, and remove or redirect each composition link. No affected predicate may receive a positive result. These tests concern recorded bindings, not faithfulness of a context report to a mental state.

7.7 Falsifiability by construction: the spec–code parity self-test

The tests in `tests/test_idr_memory_specs.py`, present in the reviewed revision and recomputing the example's address through `lib.precog.idr.content_addr`, read a worked IDR example from the specification, exercise it through `lib/precog/idr.py`, and compare its computed content address with the documented value (`IDR.md`). This is a useful check of that example. It does not prove parity for every definition, signing profile, optional field, or claim in this paper. Passing tests also do not prove the absence of parser, key-management, or binding failures.

Table 12 summarizes tests that bound the claims in this section. Except for the explicitly reported historical failure and public anchor bundle, the table states required tests, not a claim that their artifacts and passing transcripts have been released.

Limits beyond cryptographic verification. Predictive performance and commercial differentiation are separate empirical questions: Section 9.2 reports the negative watcher result, and Section 10.5 gives the bounded vendor comparison and commercial findings.

The same record reports that four non-Anthropic councils, with zero Anthropic calls and seals `7dc01f0b739be243` and `9fb9b0fc0f2d4309`, judged the integrated system "a bundle of commodities with a nice story." The commercial moat claim is refuted by those evaluations. Individual mechanisms are shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI; MIT-licensed Asqav signs with the same ML-DSA-65 and ships RFC 3161 timestamps and RFC 8785 canonicalization. The only surveyed differentiator is shipping attacks alongside the claim, based on reading four

vendors' documentation rather than running their products. EU AI Act Article 12, in force 2 August 2026, is a forced buyer of automatic record-keeping, not evidence of this implementation's uniqueness or compliance. Neither demand nor surviving selected tests establishes a general security guarantee.

Revision provenance. This paper is authored by V» (Laurie Scheepers), with AI assistance. Claude Fable 5.1 and Astra (gpt-6-astra) assisted the revision. The dated failure history, measurements, and release facts cited here are from the author's host verification record of 2026-09-22; proposed tests are explicitly distinguished from completed evidence.

8 External anchoring to Bitcoin via OpenTimestamps

External timestamps bind a specified commitment to evidence outside the record custodian's control. They do not establish that the committed account is true, complete, or contemporaneous with the actions it describes. This section distinguishes the semantic forest commitment from the production ledger commitment, reports the public Bitcoin verification bundle, and states the remaining temporal and disclosure limits.

8.1 Commitment domains: semantic bodies and ordered ledger lines

An intent-decision record (IDR) is a persisted provenance record produced by `precog/idr.py` and organised by `lib/idr_forest.py`. Its semantic content address is not a hash of its entire signed envelope. Let $P(n)$ denote the semantic projection defined in Section 3.2, including its exclusions of `id`, `ts`, and `audit`, and let $a_v(n) = \texttt{content\_addr}_v(n)$ be the prefixed hexadecimal address defined in Section 7.1. The proposed semantic forest root is

$$R(F) = \mathrm{MT}_{6962}\big([L_v(a_1), \ldots, L_v(a_N)]\big), \tag{9}$$

where $(a_1, \ldots, a_N)$ is the sorted address list with multiplicities retained, MT_{6962} is the RFC-6962 tree-hash construction, and the exact leaf-byte encoding L_v and the implementation's conformance to it remain *(unverified)*. Implementation references are `lib/idr_forest.py` and `lib/merkle.py`; this equation does not specify a new cross-language canonicalization profile.

The production anchor reported below uses a different domain: the exact ledger lines in chain order, not sorted content addresses. Writing those line byte strings as $\ell_1, \ldots, \ell_N$, its root is

$$R_{\text{ledger}} = \mathrm{MT}_{6962}([\ell_1, \ldots, \ell_N]).$$

The released bundle supplies the corresponding RFC-6962 leaf hashes. These two root profiles must not be substituted for one another: they commit to different bytes and have different ordering semantics.

Root identifiers such as `ci:`, `human:`, `council:`, and `hypo:` classify claimed evidence types. Neither those prefixes nor a timestamp authenticate an external evaluator. Establishing such authority requires a verified attestation from an authorized principal, bound to the artifact and policy being evaluated. Prefix acceptance alone is not that check.

Proposition 1 (sensitivity within the commitment domain). Under the hash-security assumptions, changing the committed multiset of semantic addresses changes $R(F)$, and changing the ordered ledger byte sequence changes R_{ledger}, except through a hash collision. This is a property of the specified tree construction, not a proof of implementation correctness.

The stronger statement that *any node change moves the semantic root* is false. Changing only an excluded timestamp and re-signing the envelope can leave $R(F)$ unchanged. Sorting also removes record enumeration order from that commitment. An external timestamp of $R(F)$ therefore does not protect excluded fields or establish chain order. An ordered full-envelope checkpoint is a separate design requirement; the production line commitment demonstrates its own byte domain, not universal checkpoint coverage of all IDR envelopes. A timestamp-change-and-re-sign regression test remains required for the semantic profile; its execution is *(unverified)* here.

Proposition 2 (membership without other record bodies). An RFC-6962 inclusion path permits verification of one leaf against a retained root using $O(\log N)$ authentication-path hashes. It avoids transmitting the other record bodies. It does not reveal nothing: sibling hashes, tree position, and associated size information are disclosed, and predictable unsalted records can be vulnerable to dictionary testing. Membership also does not establish truth, authorship, completeness, or currentness.

8.2 Internal authentication and the observed verification failure

Authentication claims depend on the signing profile. For HMAC-SHA256 records, verification requires the shared secret; every holder can also forge records. Only records actually signed under the Ed25519 or ML-DSA-65 public-signature profiles can have their signatures publicly verified, using authenticated public keys and the required verification dependencies. A combined profile must enforce its specified signature requirements rather than accept a stripped component. The relevant integrity checker is `lib/idr_forest.py::verify_chain_integrity`.

A valid authenticator checks the covered message under a key. It does not prevent a key holder from replacing that message and authenticating the replacement. Similarly, deterministic record reconstruction checks supplied records; it does not re-execute the agent or establish that recorded actions occurred. A reconstruction mismatch can reflect changed inputs, serialization, versions, target selection, or implementation behavior, not just tampering.

Known failure and repair. On 2026-09-17, a sealed council verdict was rewritten to say the opposite and its hash recomputed; the verifier still reported `PROVEN`. The defect was fixed in GRIP#6176, merged 2026-09-17T15:57Z: "`-verify` never compared the record against the hash it sealed." GRIP#6193, merged 2026-09-17T17:28Z, replaced the single pass token with an explicit list of what clean verification does not assert.[26]

This was a verifier failure, not a SHA-256 collision. A hash stored beside editable content provides no protection when the attacker can replace both. The repair does not retroactively validate earlier `PROVEN` results. A release-pinned public regression bundle for this failure is not established here *(unverified)*. Verification must positively check each claimed layer; an unavailable key, library, referenced record, or timestamp is missing evidence, not a successful check.

8.3 External time evidence

OpenTimestamps aggregates commitments and supplies proof paths to Bitcoin block-header attestations [5]. A pending calendar attestation is not a confirmed Bitcoin anchor. A completed proof must be checked against the exact commitment and the referenced block's transaction Merkle root, with an explicit source of Bitcoin chain evidence.

[26] Sources: GRIP#6176 and GRIP#6193, as verified in the author-maintained host record on 2026-09-22. These issue identifiers identify the repair history; they are not a public regression archive.

Layer	Evidence checked	Boundary
HMAC profile	Covered message and tag under the shared secret	Not publicly verifiable; secret holders can forge
Public-signature profile	Covered message, required signatures, and authenticated public keys	Does not prevent replacement by an authorized signer
Merkle commitment	Leaf encoding, ordering, and path or complete leaf-hash list	Binds only the specified domain to the expected root
External timestamp	Commitment-to-attestation proof and external time or chain evidence	Establishes historical commitment evidence, not truthful contents or a current head

Table 13: Separate evidence layers. An external timestamp can bind a digest publicly even when the underlying records use HMAC; it does not make those HMAC authenticators publicly verifiable.

An explorer can supply that evidence, but then the verifier relies on the explorer's chain view. Independently validated Bitcoin data offer a different trust boundary. Neither mode removes Bitcoin's consensus and reorganization assumptions. Bitcoin block timestamps are not precise observations of record creation or action execution; the relevant result is inclusion of a commitment in a particular historical block.

```
No proof supplied: external anchoring not established
Pending calendar proof: Bitcoin confirmation not established
Completed proof: check exact commitment and proof operations
Referenced block: check Merkle root and accepted chain evidence
Historical inclusion: does not establish currentness or completeness
```

Listing 4: Evidence states, not a new wire schema. A field claiming an anchor is not a verified anchor.

RFC 3161 and pre-action commitments. RFC 3161 trusted timestamps were added in GRIP#6190, "the external time anchor we lacked," merged 2026-09-17T17:22Z. The implementation files are `lib/rfc3161_anchor.py` and `lib/continuity_chain.py`.[27] Before that addition, the claimed anti-HARKing property—commitments before actions—bound only a third party without the signing key; it was not established against the writer.

The new timestamp mechanism supplies external time evidence, not an automatic proof of pre-action ordering for every record. Writer-resistant preregistration requires a verified timestamp of the relevant commitment before an independently evidenced action or outcome boundary. A signed, self-reported timestamp does not supply that boundary. RFC 3161 additionally requires trust in the timestamp authority and validation of its token and signing credentials.

8.4 The public production anchor

As of 2026-09-22, the proof, manifest, 501 RFC-6962 leaf hashes, and a standard-library Python verifier are public at `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. This bundle is the source for the production commitment reported here. The full values are:

[27]Source: GRIP#6190 and the host verification record of 2026-09-22. The verified deployment paths were `/Users/void/.grip/lib/rfc3161_anchor.py` and `/Users/void/.grip/lib/continuity_chain.py`.

```
Production decision-chain root (501 records):
0bba0792bf9f71caa815d0c42daf27f45baf07ef9749523be666f103a7f1bcb4

Manifest SHA-256:
5d5e8854f23eb50e583e02b380f21059651a3a4e99b4454cb221d355725a45b9
```

The OpenTimestamps proof attests the production commitment in Bitcoin blocks 956991 and 956992, whose transaction Merkle roots begin `1b42ef4d...` and `b3f15c01...`, respectively. A second stamp of the same digest completed in blocks 956948, 956963, and 956970. These are multiple attestations of the same commitment, not additional covered records or newer ledger checkpoints. The abbreviated block Merkle roots here are descriptive; verification uses the complete values through the public proof and chain checks.

The leaves are the exact ledger lines in chain order. The records themselves are not published because their subjects name people. Consequently, a public verifier can reconstruct the root from the released leaf hashes and check its timestamp evidence, but cannot independently recompute every leaf hash from the withheld ledger lines. A recipient who is separately given a record and its exact leaf encoding can check that record's membership. Publication of hashes alone does not prove the contents, signatures, or completeness of the underlying records.

From the downloaded bundle, the documented invocation is:

```
python3 verify.py --explorer
```

The verifier reports `ROOT`, `PROOF`, and `CHAIN` separately, and emits `VERIFIED` only when all three pass. `ROOT` checks the commitment reconstruction, `PROOF` checks the timestamp proof, and `CHAIN` checks the external chain evidence. The explorer invocation is not offline and retains the explorer trust boundary. Missing chain evidence must not be represented as completed external verification.

This public bundle supports a claim about one 501-record ledger commitment. It does not demonstrate anchoring of the later council artifact, the entire semantic forest, belief reports, claim receipts, or the paper's complete source tree.

8.5 Coverage, freshness, and operational scope

Continuous automatic anchoring on every IDR write is not established by the published production proof. The existence of an anchor field, timestamp code, or a successful historical stamp does not establish deployment-wide coverage. The proposed financial-records pilot's independent coverage remains *(unverified)* in this section.

Historical inclusion also does not identify the current head. Currentness requires an authenticated checkpoint and head, a freshness policy, evidence of extension, and a way to discover sufficiently recent checkpoints. A valid old proof can remain valid after newer records exist. Continuous checkpoint publication and stale-head or equivocation detection are future work unless separately demonstrated.

The contribution claimed here is not an exclusive cryptographic combination. Four non-Anthropic councils, associated with seals `7dc01f0b739be243` and `9fb9b0fc0f2d4309` and zero Anthropic calls, judged the integrated system "a bundle of commodities with a nice story." The individual mechanisms are shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI. Asqav is MIT-licensed, uses the same ML-DSA-65 signature scheme, and ships RFC 3161 timestamps and RFC 8785 canonicalisation. The only surveyed differentiator is shipping attacks alongside the claim; that

comparison rests on reading four vendors' documentation, not running their products.[28] EU AI Act Article 12, in force on 2 August 2026, creates a forced buyer of automatic record-keeping, not evidence of a moat for this implementation or proof that this anchor alone satisfies the legal obligation.[29]

Anchoring likewise provides no evidence of predictive utility. The watcher measurement of 2026-09-10 used seed 0, five stratified folds, a 50-event window, and 76 paired rows at a 13.2% base rate. No predictor's precision interval clears that base rate. The pre-hoc watcher therefore remains `WARN`-only, and its hook is unbound.[30] These negative results constrain operational and commercial claims independently of whether the timestamp proof verifies.

8.6 Offline receipt checks are a separate predicate

The receipt checker `tools/grasp-verify-receipt` in the public GRASP repository concerns supplied deliverable bytes, source bytes, and enumerated quotation spans, rather than Bitcoin inclusion (Section 5.1). Its deterministic checks can be performed offline when those inputs are available. The earlier production receipt transcript and its numerical results are not independently established by the anchor bundle *(unverified)*; they are not repeated as measured evidence here.

```
Compare delivered bytes with the expected deliverable digest.
Compare supplied source bytes with the expected source digests.
Check each enumerated quotation against its supplied source.
Report quote matches separately from unexamined claims.
```

Listing 5: Scope of receipt consistency checks; this is not an execution transcript.

Even a complete quote-match result establishes only that the enumerated spans occur in the supplied, hash-pinned sources. It does not authenticate source acquisition, establish semantic support or factual truth, or show that every claim in the deliverable was enumerated. An attacker-supplied source can contain an invented quotation and still hash correctly. Receipt authorship and links to decisions or context reports require separate signature and target-resolution checks; the checks above alone do not establish them.

8.7 Verification conditions and remaining tests

Table 14 separates publicly checkable claims from tests requiring additional records, secrets, or fixtures. These are acceptance conditions and proposed negative tests, not a claim that every test has been published or run.

The public result is a checkable historical commitment over a specified ledger prefix. Its value is that replacement can be tested against evidence retained outside the custodian, within the commitment's actual byte domain. It is not evidence that the custodian recorded every event, described events faithfully, or supplied the latest history.

Revision provenance. This section was revised by V» (Laurie Scheepers), with AI assistance from Claude Fable 5.1 and Astra (gpt-6-astra), using the host-verified record of 2026-09-22.

[28] Source: the four-council outcomes and vendor documentation survey recorded in the host verification of 2026-09-22, including the two seal identifiers above. This is not a product benchmark.

[29] Source: Regulation (EU) 2024/1689, Article 12 and its applicable commencement date, as verified in the host record on 2026-09-22.

[30] Source: watcher measurement record of 2026-09-10, verified by the host on 2026-09-22.

Claim or boundary	Failure observation	Required material
Semantic root sensitivity	A changed committed semantic multiset has the same root without a hash collision; excluded-field edits are not counterexamples	Exact projection and serialization
Ordered ledger commitment	Released leaf hashes do not reproduce the reported root	Public leaf-hash list and verifier
Production timestamp	The proof fails to bind the expected commitment to the stated blocks under the accepted chain view	Public proof and chain evidence
Positive acceptance	Missing proof or chain evidence still yields completed external verification	Public verifier and negative fixtures
Record membership	A disclosed record fails to reproduce its claimed leaf or inclusion path	Exact record bytes, encoding, and path
HMAC integrity	An altered covered message passes under an unchanged expected tag and key	Shared secret; not a public-only test
Writer-resistant ordering	A post-outcome, backdated commitment is accepted as pre-action without independent ordering evidence	Timestamp and action-boundary evidence
Receipt quote matching	An absent quotation is accepted against fixed source bytes	Receipt, specification, and source bytes

Table 14: Scoped verification conditions. Public timestamp verification does not expose private ledger records, authenticate HMAC records, or establish action completeness.

9 Evaluation: production dogfood and overhead

This section separates production observations, known failures, public verification evidence, and measurements that remain to be done. The council instrument, `bin/grip-sovereign-council`, exercises a payload hash, a decision authenticator, and a forest commitment. These are three integrity layers, not the three legs defined elsewhere in the paper: decision record, recorded context report, and claim receipt. The evaluated council artefact does not demonstrate a complete, authenticated three-leg bundle.

The evidence comprises a historical council record, the subsequent verifier failure and repairs, a public timestamp bundle covering a different ledger prefix, and negative results concerning predictive precision and commercial differentiation. These observations do not constitute a latency benchmark or evidence of complete recording of production actions.

Evidence and revision provenance. Implementation references below identify files or known symbols, not source line ranges. The host-session verification record of 2026-09-22 supports only the explicitly supplied failure-and-fix history, the RFC 3161 addition, the watcher measurement, the commercial findings, the public anchor facts, the historical seal's identity and signing scheme, and authorship provenance; other implementation descriptions in this section remain *(unverified)* unless separately supported, and none are presented as independently reproduced experiments. This paper is authored by V» (Laurie Scheepers), with AI assistance. This revision was assisted by Claude Fable 5.1 and Astra (gpt-6-astra).

Table 15: Required evidence for a future overhead evaluation; these are reporting requirements, not measured results.

Quantity	Required report
Latency	Absolute times, median and tail latency, repetitions, and uncertainty using a monotonic timer.
Resource cost	Stored bytes, payload and history sizes, durable-write policy, and anchoring costs.
Comparison	Matched workload with and without recording, hardware, concurrency, and a prespecified practical threshold.
Effect size	Optional standardized summary alongside absolute costs; not a substitute for uncertainty or operational relevance.

9.1 The dogfood instrument

The council dispatches a prompt to speaker models, submits their answers to a distinct judge model, and persists a verdict (`bin/grip-sovereign-council`). Judge–speaker separation avoids the specific case of a model judging its own submitted answer. It does not establish evaluator independence, factual correctness, or resistance to prompt manipulation.

The historical record described in this evaluation is `state/sovereign-council/b7e2828e3a298879.json`. Its payload is

$$P = \{\texttt{prompt},\ \texttt{speakers},\ \texttt{judge}\}.$$

Speaker entries contain recorded model identifiers, status, and answer text; the judge entry contains a recorded model identifier and verdict. The persisted envelope additionally contains a payload digest, a claimed sealing time, an IDR reference, and an `anthropic_calls` count. These record-level fields are not all covered by the payload hash.

The historical record exercises the HMAC-SHA256 profile, not public-key verification: its IDR leaf `precog-1783476751-79074cfb` carries `scheme: hmac-sha256` and envelope version `1.3` (host-verified on 2026-09-22 by reading the private state file). Its reported sealing time is 2026-07-08T02:12:31Z; that field is a recorded timestamp, not independent evidence of wall-clock time. The underlying private state artefact is not supplied here for independent reproduction *(unverified)*. In particular, this section does not establish that the record has an authenticated paired context node, a claim receipt, or an external timestamp.

9.2 The measurement discipline

The deployment's measurement policy is described in `rules/scientific-method.md`. Policy is not measurement evidence. Cohen's d is a standardized effect size: it does not determine statistical significance, a universal noise threshold, or practical importance. Session snapshots from `delight_v2.py` and `conversation_entropy.py` are not established latency instruments.

Table 15 states the reporting requirements for a future overhead study. No such study is reported here. Practical thresholds must be chosen for the deployment before inspecting results, rather than inferred from a generic effect-size band.

The deployment also defines auxiliary "shadow" metrics (Table 16). A rule allowing them only to reduce a score is a scoring convention, not demonstrated resistance to gaming.

Table 16: Candidate shadow metrics listed in `rules/scientific-method.md`. They are described candidates, not host-verified deployment facts; their implementation and effectiveness as safeguards against gaming are *(unverified)*.

Primary signal	Auxiliary metric	Intended proxy
Velocity	`velocity_shadow`	Revert rate
Flow state	`flow_shadow`	Confusion
Entropy	Inverted-U penalty	Extreme values

Measured negative result: pre-hoc watcher. The watcher measurement on 2026-09-10 used seed 0, five stratified folds, a 50-event window, and 76 paired rows, with a base rate of 13.2%. No predictor's precision interval clears the base rate. The pre-hoc watcher therefore remains `WARN`-only and its hook is unbound.[31] This is not evidence for predictive enforcement or improved decision quality. A future evaluation should publish the row construction, confusion matrices, interval method, and treatment of dependence between overlapping windows; those materials are not supplied in this section.

9.3 Provenance overhead: three deterministic layers

Layer 1: payload hash. Let $c(P)$ denote the exact serialization produced by the council implementation and let H be SHA-256. The stored seal is

$$\sigma = H(c(P)).$$

The seal digest is SHA-256 over `canonical_bytes(payload, seal_format)` (`lib/sovereign_council.py::canonical_bytes`); the historical format is described as Python `json.dumps` with `sort_keys=True` and `ensure_ascii=False`, then UTF-8 encoding *(exact options unverified beyond the source description)*. Those options alone do not define an interoperable canonicalization protocol. Independent implementations also need exact separator, number, duplicate-key, and normalization rules. Cross-language equivalence remains *(unverified)*. The filename uses a digest prefix; it is a lookup convention, not an independent commitment to the payload.

Layer 2: decision authentication. The council's decision-record path records the judge result and payload digest in an IDR (`bin/grip-sovereign-council`; `lib/idr_forest.py`). The historical evaluated profile is HMAC-SHA256. Verification requires the shared secret, and every holder of that secret can also forge authenticators. It is therefore not publicly verifiable provenance.

Public-key verification is available only for records actually signed under the Ed25519 / ML-DSA-65 profiles, using independently authenticated public keys and the required algorithm checks. A companion exercise is described in `tests/test_pq_dual_exercise.py`, with a private state artefact at `state/pq-eval/pq-eval-record.json`. Its reported positive and mutation cases are not a proof of implementation correctness, and independent reproduction from a released bundle remains *(unverified)*. It does not change the profile of the historical council record.

Layer 3: forest commitment. The council computes a root for verdict records (`bin/grip-sovereign-council`). A semantic forest root commits only to its defined projected bodies. It does not commit to excluded envelope fields merely because those fields appear in the stored record. For

[31] Host-session verification record, 2026-09-22: watcher measurement dated 2026-09-10.

example, changing an excluded timestamp and re-signing with the writer's key need not change a semantic address or the resulting root. A full-envelope, ordered checkpoint is a distinct commitment profile; it must not be inferred from a semantic root.

Operation counts, not measured overhead. Payload hashing is linear in serialized payload size. Authentication also processes the profile's signed message. Rebuilding a Merkle tree over m leaves requires a linear number of hash operations, in addition to leaf preparation. These operations invoke no language model, but zero model calls does not imply negligible latency, storage, lock contention, or timestamping cost. The claim that provenance cost is dominated by deliberation remains *(unverified)*. No council-root anchoring cost is amortized here: external anchoring of this council record has not been established.

Property E1: fail-open recording is a limitation. The evaluated council path permits completion with a payload hash when IDR recording fails: `lib/sovereign_council.py::_seal_idr` catches any exception from `build_idr`/`append_idr`, logs "IDR signing degraded" and keeps the SHA-256-only record (host-verified 2026-09-22). This is availability-oriented degradation, not evidence of authenticated provenance. It contradicts any unqualified claim that every completed decision necessarily has both an IDR and a paired context record.

A future complete-recording claim requires fail-closed mediation or independent reconciliation of actions against required records, together with crash and retry tests. Omission of either required record must fail that claim. Those controls are not demonstrated by this evaluation.

9.4 Tamper-evidence: known failure and bounded checks

Observed counterexample and version boundary. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176 fixed the defect: "`-verify` never compared the record against the hash it sealed." The fix merged at 2026-09-17T15:57Z. GRIP#6193, merged at 2026-09-17T17:28Z, replaced the single pass token with an explicit list of what a clean verification does not assert.[32]

This was an implementation and binding failure, not a SHA-256 collision. Pre-fix `PROVEN` results are not evidence of the repaired check. The identified repairs do not establish that every adversarial mutation is now rejected. A public failing fixture, pinned pre- and post-fix executions, and a complete regression corpus are not supplied here; independent regression reproduction remains *(unverified)*.

Property E2: payload consistency against a fixed digest. Reconstructing P and comparing $H(c(P))$ with an expected digest detects changes to the serialized payload under the hash assumptions only if that expected digest is retained independently or authenticated. If an attacker can replace both the payload and its adjacent digest, recomputation alone establishes only self-consistency. Editing fields outside P need not affect this check.

The relevant negative tests therefore include replacement of payload and digest together, substitution of the filename and IDR pointer, replacement with a re-signed IDR, omission of verification dependencies, and replay of an older valid bundle. They are required tests for broader claims, not results claimed as completed here.

[32] GRIP#6176 and GRIP#6193, merge records and failure description verified in the host session on 2026-09-22.

Separate verification predicates. A verification report must distinguish the following questions. These names describe evidence predicates, not a claim that every name is an implemented command-line status.

`HASH_CONSISTENT` Does the supplied payload match the supplied digest under the specified serialization?

`AUTHENTICATED` Does the required authenticator verify, and is the council payload digest bound to the authenticated IDR? For HMAC this requires the secret; public verification requires an asymmetric profile.

`EXTERNALLY_ANCHORED` Does an independently checked timestamp or checkpoint cover the actual claimed commitment domain?

`COMPOSITION_COMPLETE` Are the decision, context report, and receipt present, authenticated, and correctly cross-linked?

`CURRENT` Is there authenticated head information, extension evidence, and a satisfied freshness policy?

Missing keys, records, dependencies, or proofs make the corresponding predicate `UNVERIFIABLE`; they are not positive evidence. In particular, "not `BROKEN`" is insufficient for acceptance. Local integrity checking (`lib/idr_forest.py::verify_chain_integrity`) does not by itself establish the last three predicates.

9.5 External timestamps and their coverage

RFC 3161 time anchor. RFC 3161 trusted timestamps were added in GRIP#6190, "the external time anchor we lacked," merged at 2026-09-17T17:22Z. The implementation modules are `lib/rfc3161_anchor.py` and `lib/continuity_chain.py`, deployed beneath `/Users/void/.grip/`.[33] Before this addition, the anti-HARKing property bound only a third party without the signing key; it was not established against the writer.

A trusted timestamp can establish that a commitment existed by the timestamped time, under the timestamp authority's trust assumptions. To establish pre-action commitment against the writer, that evidence must also precede an independently evidenced action or outcome boundary. Adding the module does not retroactively timestamp earlier records or demonstrate this ordering for every production decision.

Public Bitcoin anchor. A separate production decision-chain prefix of 501 records has root

`0bba0792bf9f71caa815d0c42daf27f45`
`baf07ef9749523be666f103a7f1bcb4`.

The manifest SHA-256 is

`5d5e8854f23eb50e583e02b380f210596`
`51a3a4e99b4454cb221d355725a45b9`.

An OpenTimestamps proof attests this root in Bitcoin blocks 956991 and 956992, whose Merkle roots begin `1b42ef4d` and `b3f15c01`, respectively. A second stamp of the same digest completed in blocks

[33] GRIP#6190 and deployment module paths, verified in the host session on 2026-09-22.

956948, 956963, and 956970. These are stamps of the same digest, not additional independently covered histories.

The proof, manifest, 501 RFC 6962 leaf hashes, and standard-library verifier have been public since 2026-09-22.[34] From that directory, the command is:

```
python3 verify.py --explorer
```

The verifier reports `ROOT`, `PROOF`, and `CHAIN` separately, and reports `VERIFIED` only when all three pass (`docs/tmif/anchor/verify.py`). Explorer-based verification depends on the explorer's chain data; it is not equivalent to independently validating Bitcoin consensus and the relevant confirmation policy.

The anchored leaves are the exact ledger lines in chain order, not sorted content addresses. The records themselves are not public because their subjects name people. Thus the public bundle permits reconstruction from the published leaf hashes and checking the timestamp proof, but not independent recomputation of those leaf hashes from undisclosed record bytes or inspection of their contents. This public hash-and-timestamp verification does not verify HMAC authenticators without the secret.

The anchor covers this particular ledger prefix. It does not establish anchoring of the later historical council record, continuous anchoring of the semantic forest, or coverage of context nodes and claim receipts. Historical inclusion also does not establish currentness, completeness, or truthful contents.

9.6 Recorded model provenance and commercial findings

The council payload contains model identifiers for speakers and judge. Binding those fields to an authenticated commitment preserves the recorded roster; it does not independently prove that the named providers executed the requests or that no unrecorded contribution occurred. The separate `anthropic_calls` field is outside the payload triple and cannot acquire integrity merely through agreement with that roster. The stronger interpretation of "sovereignty" as a complete audit of all cognition is therefore not established.

There is nevertheless a concrete negative commercial result: four non-Anthropic councils, with zero Anthropic calls, judged the integrated system "a bundle of commodities with a nice story." The associated seal identifiers are `7dc01f0b739be243` and `9fb9b0fc0f2d4309`.[35] The commercial differentiation claim failed these evaluations. Council judgment is negative evidence, not a market experiment.

The individual mechanisms are already shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI. Asqav is MIT-licensed, signs with the same ML-DSA-65, and ships RFC 3161 timestamps and RFC 8785 canonicalisation.[36] We claim neither a new cryptographic primitive nor a demonstrated commercial moat. The only surveyed differentiator is shipping attacks alongside the claim. That finding rests on reading four vendors' documentation, not running their products; it does not establish exclusivity.

EU AI Act Article 12, in force from 2 August 2026, is a forced buyer of automatic record-

[34] `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`, public bundle verified in the host session on 2026-09-22.

[35] Four-council commercial evaluation, associated seals and zero-Anthropic-call result verified in the host session on 2026-09-22.

[36] Comparative implementation facts verified in the host session on 2026-09-22 from the named projects' documentation.

keeping.[37] This regulatory demand does not demonstrate compliance by this system, procurement preference, or willingness to pay for this implementation.

9.7 Concurrency overhead

The council fans speakers out through a `concurrent.futures.ThreadPoolExecutor` with one worker per speaker (`lib/sovereign_council.py`). Concurrency is distinct from provenance overhead. For positive aggregate speaker latency $S = \sum_{i=1}^{n} \ell_i$ and a wall-clock interval T covering only that fan-out, a descriptive overlap statistic is

$$\omega = 1 - \frac{T}{S}.$$

Under idealized serial execution, $T = S$ and $\omega = 0$. Under idealized balanced parallel execution with negligible overhead, $\omega = 1 - 1/n$. Scheduling and dispatch costs can make ω negative; a zero or poorly resolved denominator should be reported as undefined, not perfect overlap.

No measured concurrency result is supplied here. A thread pool does not guarantee useful overlap when providers serialize requests, rate-limit calls, or have uneven latency. A future benchmark must define the timing boundaries, exclude judge and sealing time from a speaker-only statistic, and report failures and retries alongside completed requests.

9.8 Summary

The strongest publicly inspectable evidence in this section is the 501-leaf hash-and-timestamp bundle, with its stated privacy and chain-data limitations. It is not a public release of the underlying records or a verification of their HMAC signatures.

The council evaluation demonstrates why hash consistency, authentication, external timing, complete composition, and freshness must remain separate. A rewritten verdict passed the old verifier; GRIP#6176 repaired the missing binding check, and GRIP#6193 replaced the single pass token with explicit non-claims. RFC 3161 support supplies an external time-anchor mechanism, but pre-action ordering still requires evidence about the action boundary.

The watcher did not establish precision above its base rate, the commercial differentiation claim failed four councils, and negligible recording overhead remains unmeasured. Priorities for future work are a released verifier regression corpus, one complete three-leg bundle, and a reproducible overhead benchmark. None is treated here as an already completed result.

10 Related work and complementarity

This work applies established provenance mechanisms to agent-authored decision records, context reports, and claim receipts. We claim no new cryptographic primitive, exclusive mechanism combination, or demonstrated commercial moat. The demonstrated contribution comprises the reported verifier failure and the public 501-record anchor-verification bundle; reproducible publication of the broader attack-to-claim corpus remains *(unverified)*, notwithstanding the documentation survey's identification of shipping attacks alongside claims as the only surveyed differentiator. This section distinguishes that contribution from existing mechanisms, identifies possible integrations, and states the limits of the comparison.

[37] Regulation (EU) 2024/1689, Article 12; effective-date and record-keeping fact verified in the host session on 2026-09-22.

10.1 Foundations: tamper-evident logging

Merkle trees support compact commitments and logarithmic-size membership proofs under the specified tree construction [1]. Linked timestamping provides evidence about document existence and ordering under its witnessing assumptions [2]. Certificate Transparency specifies Merkle inclusion and consistency proofs for append-only logs [3]. OpenTimestamps binds digests to Bitcoin block commitments [5, 4]. These mechanisms authenticate particular byte commitments; they do not establish that recorded events happened, that all events were recorded, or that a supplied history is current.

The commitment domain matters. A root over projected semantic bodies does not commit to excluded timestamps, identifiers, audit fields, or signatures. An external timestamp on that root cannot restore those omitted bindings. Likewise, a membership proof avoids transmitting other record bodies but reveals authentication-path hashes and structural metadata. It is not a hiding commitment and does not generally prevent inference about predictable contents.

Table 17 identifies existing implementations of the mechanisms used here. It is a documentation comparison, not a product benchmark or an exhaustive feature survey.

System	Relevant existing capability
Sigstore	Artifact signing and verification, with transparency-log integration.
Rekor	Transparency logging of signing-related records and verifiable inclusion.
`git commit -S`	Signatures on Git commit objects; not, by itself, an independent time anchor.
OpenTimestamps	Digest timestamping through Bitcoin commitments.
KERI	Key-event logs and cryptographically verifiable identifier/key-state histories.
Asqav	MIT-licensed signing with ML-DSA-65, RFC 3161 timestamps, and RFC 8785 canonicalisation.

Table 17: Commodity mechanisms relevant to this work. Application schemas and verification policies still determine what each deployment establishes.

The comparison draws on the projects' documentation and the author-maintained comparison record verified on 2026-09-22.[38] In particular, use of ML-DSA-65 together with external timestamps is not unique to this system. Nor does use of JSON serialization alone establish conformance to RFC 8785.

Verification also depends on the signing profile. HMAC records require the shared secret for authentication; a verifier receiving that secret also gains the ability to forge authenticators. Only records actually signed under the Ed25519 / ML-DSA-65 public-signature profiles permit public-key verification, subject to authenticated keys and the profile's required signature checks. Public verification of a digest timestamp is a separate operation and does not make an HMAC record publicly authenticatable.

10.2 Machine-learning lineage and AI bills of materials

Machine-learning lineage and AI bills of materials describe relationships among datasets, transformations, models, evaluations, and outputs. The lifecycle system discussed below is one concrete

[38] Documentation entry points: `https://docs.sigstore.dev/`, `https://github.com/sigstore/rekor`, `https://git-scm.com/docs/git-commit`, `https://opentimestamps.org/`, and `https://keri.one/`. The Asqav licensing and capability comparison is recorded in the author-maintained survey verified on 2026-09-22; no product execution is claimed here. Protocol specifications: `https://www.rfc-editor.org/rfc/rfc3161` and `https://www.rfc-editor.org/rfc/rfc8785`.

example [7]. Such records can support checks on claimed construction history. They do not, merely by being committed or signed, prove that a model was built correctly or that its declared inputs are complete.

Our application schema instead emphasizes supplied records from agent operation. The distinction is one of emphasis, not an exclusive technical boundary: lineage systems can include deployment events, and general-purpose logs can store agent records. Here, a "belief" record means an instrumented context report, not verified access to internal mental state. A claim receipt checks enumerated quotation spans against supplied, hash-pinned source bytes; it does not by itself establish source authenticity, semantic support, factual truth, or exhaustive claim coverage. Whether particular lineage products support the same application schema without modification remains *(unverified)*.

10.3 Policy-as-code and verifiable computation

Policy-as-code engines, such as Open Policy Agent, evaluate rules over supplied facts.[39] They can govern action admission or evaluate recorded evidence. A rule engine alone does not authenticate its inputs, but it can be integrated with signed records and transparency logs. Blocking an action is compatible with recording the attempted action and the denial; there is no necessary conflict between enforcement and forensic recording.

Trusted-execution attestation and zero-knowledge computation address different predicates. An attestation can bind a measurement to a platform under its hardware and key assumptions. A proof system can establish a specified computation relation under its circuit and cryptographic assumptions. Neither automatically proves a broadly stated claim that the "right model ran," and neither automatically authenticates the surrounding action history. Integration would require explicit bindings among the attested or proved computation, input and output digests, decision record, and relevant action identifier. Such an end-to-end integration is future work, not a demonstrated result here.

External evaluation is likewise a governance policy rather than a theorem of correctness. A `human:`, `ci:`, or `council:` prefix classifies a claimed source; it does not authenticate that source or establish its independence. Provider separation can reduce one form of self-evaluation, but does not alone establish resistance to shared errors or manipulation.

The pre-hoc watcher is not evidence of successful predictive enforcement. The measurement on 2026-09-10 used seed 0, five stratified folds, a 50-event window, and 76 paired rows, with a base rate of 13.2%. No predictor's precision interval clears that base rate. The watcher therefore remains `WARN`-only, and its hook is unbound.[40]

10.4 A declaration layer above: standardised transparency claims

The Transparency Metadata Interchange Format (TMIF) [6] describes a format for communicating transparency claims, threats, mitigations, and supporting artifacts. Its role is to make claims available for evaluation, rather than to produce the underlying agent-runtime records.

This work can supply evidence for some such claims: signed records, specified commitment domains, timestamp proofs, and executable checks. The relationship is potentially complementary, but surviving attempted refutation does not prove a transparency level. Evaluators must distinguish a passed predicate from an untested claim and from unavailable evidence. A clean signature check,

[39] `https://www.openpolicyagent.org/docs/`

[40] Author-maintained watcher measurement record dated 2026-09-10, verified by the host session on 2026-09-22. No positive discrimination result or enforcement deployment is claimed.

for example, says nothing by itself about recording completeness, action timing, context-report fidelity, or factual truth.

There is now a public, narrowly scoped anchor bundle. Since 2026-09-22 it has contained a proof, a manifest, 501 RFC 6962 leaf hashes, and a standard-library verifier at `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. The production decision-chain root and manifest digest are, respectively:

```
root:
0bba0792bf9f71caa815d0c42daf27f45baf07ef9749523be666f103a7f1bcb4
manifest sha256:
5d5e8854f23eb50e583e02b380f21059651a3a4e99b4454cb221d355725a45b9
```

The root commits to 501 records using the exact ledger lines in chain order, not sorted content addresses. Its OpenTimestamps proof is attested in Bitcoin blocks 956991 and 956992, whose Merkle roots begin `1b42ef4d` and `b3f15c01`, respectively. A second stamp of the same digest completed in blocks 956948, 956963, and 956970.[41] The command `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately and reports `VERIFIED` only when all three pass. Explorer-backed chain checking retains an external data-source trust assumption.

The record bodies are not public because their subjects name people. Consequently, the public bundle supports checking the root from the supplied leaf hashes and checking its timestamp proof, but not independently recomputing those leaf hashes from the private ledger lines. It does not establish record truth, continuous anchoring, or coverage of the belief and receipt chains. Nor is it a complete self-verification release of this paper: Appendix B must not be read as supplying a detached manifest of all paper sources, fixtures, and dependencies unless that separate bundle is actually released.

10.5 The nearest neighbour, and how it composes

The lifecycle comparator [7] is reported to represent datasets, features, models, metrics, and human oversight records as a Merkle-style directed acyclic graph, to evaluate constraints off-chain with Open Policy Agent, and to enforce constraints on-chain with a smart contract; that description was not independently inspected for this revision *(unverified)*. Its combination of provenance and policy evaluation overlaps with this work. The two may compose by binding a deployed artifact's digest to lifecycle evidence and then binding runtime records to that artifact. Technical overlap and commercial substitution are not ruled out.

Figure 2 shows a possible division of responsibilities, not a proof that the categories are disjoint or jointly sufficient for accountability.

Negative evidence about novelty and commercial defensibility. Four non-Anthropic councils, with zero Anthropic calls, judged the integrated system "a bundle of commodities with a nice story." The associated seals are `7dc01f0b739be243` and `9fb9b0fc0f2d4309`.[42] The commercial moat claim is therefore withdrawn. The only surveyed differentiator is shipping attacks alongside the claim. That finding rests on reading four vendors' documentation, not running their products, and does not establish market-wide uniqueness.

[41] Public proofs, manifest, leaf hashes, and `verify.py` in the anchor bundle linked above, published 2026-09-22.

[42] Author-maintained commercial-claim council record, identified by these seals and verified on 2026-09-22. The seals are record identifiers, not independent proof of provider participation.

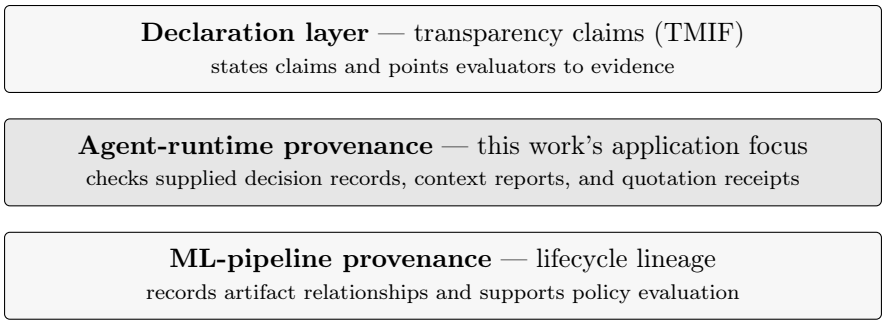

Figure 2: A possible accountability architecture. These responsibilities can overlap. Composition requires authenticated cross-layer bindings and does not by itself establish completeness, correctness, or currentness.

EU AI Act Article 12, in force from 2 August 2026, creates a forced-buyer requirement for automatic record-keeping in its applicable high-risk AI scope.[43] This is demand for a capability, not evidence that this implementation is compliant, uniquely suitable, or commercially defensible.

Attack-backed positioning and its limits. The strongest reason to publish attacks is that a plausible cryptographic description can coexist with a broken verifier. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`. GRIP#6176 fixed the defect: "`-verify` never compared the record against the hash it sealed" (merged 2026-09-17T15:57Z). GRIP#6193 replaced the single pass token with an explicit list of what a clean verification does not assert (merged 2026-09-17T17:28Z).[44] A recomputed hash beside replaceable content is not an independent commitment. The failure was a missing binding check, not a break of SHA-256.

Temporal claims have a separate version boundary. RFC 3161 trusted timestamps were added in GRIP#6190, "the external time anchor we lacked," merged 2026-09-17T17:22Z.[45] Before that addition, the anti-HARKing property bound only a third party without the signing key; it was not established against the writer. A timestamp can establish that a bound commitment existed by the attested time, under the timestamp authority's assumptions. Establishing commitment before an action or outcome additionally requires independently evidenced timing of that action or outcome. Availability of the timestamp module is not evidence that every relevant commitment used it.

The contribution should therefore be evaluated by inspecting the released attacks, their covered claim domains, and reproducible outcomes. Missing fixtures, nonreproducible results, or guarantees exceeding the released evidence count against it. A complete public attack-to-claim regression bundle remains *(unverified)* here; releasing and independently running that bundle is future work. This criterion is narrower and more useful than claiming novelty from an exact conjunction of application-specific features.

Revision provenance. This paper is authored by V» (Laurie Scheepers), with AI assistance. Claude Fable 5.1 and Astra (gpt-6-astra) assisted this revision; authorship and responsibility for its claims remain with V».

[43] Regulation (EU) 2024/1689, Articles 12 and 113: `https://eur-lex.europa.eu/eli/reg/2024/1689/oj`.

[44] Author-maintained change records GRIP#6176 and GRIP#6193, verified on 2026-09-22. These identify the historical failure and patch boundary; this section does not claim a publicly reproduced regression corpus.

[45] GRIP#6190; implementation files `lib/rfc3161_anchor.py` and `lib/continuity_chain.py`, verified on 2026-09-22.

11 Discussion, Limitations, and Falsification

The construction makes selected properties of supplied records checkable; it does not establish complete runtime conduct, faithful reports of internal model state, or the truth of recorded claims. Deterministic record reconstruction is not execution replay. An external timestamp binds the existence of a particular commitment, not the accuracy or completeness of the history it describes. Accordingly, the mechanisms in Sections 7 and 8 must be interpreted through their actual commitment domains, verification profiles, and available evidence.

This section separates checkable predicates, governance policies, observed failures, and proposed empirical studies. Implementation changes and measurements reported below are drawn from the author-maintained host-session record verified on 2026-09-22; that record is not a substitute for a public regression bundle. The public Bitcoin verification bundle is identified separately.

Revision provenance. This paper is authored by V» (Laurie Scheepers), with AI assistance. Claude Fable 5.1 and Astra (gpt-6-astra) assisted this revision.

11.1 Falsifiability by construction as the organising principle

The defensible objective is to publish narrow claims together with the evidence and procedures needed to challenge them. A decision record authenticates a recorded decision description under a specified signing profile. A belief-chain node authenticates a supplied context report, not an actual mental state or its causal role. A deterministic claim receipt checks enumerated quotation spans against supplied source bytes; it does not establish source authenticity, semantic support, or coverage of every assertion in a deliverable.

Definition 11.1 (Falsifiable record)**.** For a declared predicate P, verification profile Π, and evidence set E, a record r is *falsifiable by construction* if an effective procedure $V_\Pi(r, E)$ can be independently executed and returns

$$V_\Pi(r, E) \in \{\textsc{Confirm}, \textsc{Refute}, \textsc{Unverifiable}\},$$

with a specified counterexample class that would refute P. Confirm means only that the declared predicate passes under the profile's assumptions. Missing prerequisites produce Unverifiable, not confirmation.

Independence here means that the verifier can execute the procedure without delegating its verdict to the record's author. It does not mean that keys, software, external services, or chain data require no trust. HMAC verification requires the shared secret; every holder of that secret can also forge HMAC records. Only records actually signed under the Ed25519 or ML-DSA-65 profiles support public-key verification, and the verifier must authenticate the public keys independently. A dual-signature claim additionally requires checking both required signatures and rejecting downgrades.

A useful report therefore separates hash consistency, authentication, external anchoring, composition completeness, and freshness. These are distinct predicates, not interchangeable names for success. For example, a record can have a valid signature but no external timestamp, or historical inclusion but unknown currentness. This separation is a reporting requirement here, not a claim that every existing verifier already exposes this complete interface.

Known verifier failure and version boundary. On 2026-09-17, a sealed council verdict was rewritten to say the opposite, its hash was recomputed, and the verifier still reported `PROVEN`.

GRIP#6176 fixed the defect: "`-verify` never compared the record against the hash it sealed." The fix was merged on 2026-09-17 at 15:57Z. GRIP#6193, merged that day at 17:28Z, replaced the single pass token with an explicit list of what clean verification does *not* assert.[46]

This is a concrete counterexample to the former verification claim, not a cryptographic collision. Hashing mutable content and comparing it with a replaceable adjacent hash establishes only internal consistency. Authentication requires comparison with a positively verified binding; resistance to a writer holding the signing key additionally requires an independently retained commitment covering the relevant bytes. No guarantee here applies retroactively to the pre-fix verifier. The patch history identifies the correction, but a public, revision-pinned pre-fix/post-fix regression corpus is still needed to support independently reproducible claims about this failure and its repair.

Definition 11.2 (Exogenous verifier)**.** Relative to a declared threat model, a verifier is *exogenous* to a producing agent when the acceptance policy and relevant evidence or verification authority are controlled outside that agent's authority, and their binding to the evaluated artifact is authenticated. Resistance to manipulation is a separate empirical property; it does not follow from organizational or model-provider separation.

Self-checking and independent acceptance. Self-checking can detect errors and provide useful telemetry. It is not independent evidence of the properties the producer merely reports about itself. Conversely, deterministic verification need not be unpredictable: a correct signature verifier should be predictable. The important questions are whether its inputs are authenticated, whether its predicate tests the claimed property, and whether the producer can substitute the evidence or acceptance policy. Surviving attempted refutation increases the available evidence for a bounded claim; it does not prove all claims about a record or system.

11.2 Shadow metrics and the monotonicity of self-doubt

A conservative governance rule can prevent self-telemetry from increasing a reported confidence value. This is a chosen update policy, not a theorem about epistemic reliability.

Proposition 11.1 (Monotonicity of self-doubt)**.** *Let $c \in [0,1]$ be a confidence value and let $\phi(s) \in [0,1]$ be a ceiling derived from a self-telemetry signal s. An update defined by*

$$c' = \min\big(c, \phi(s)\big)$$

satisfies $c' \leq c$.

The inequality follows directly from the definition. Whether deployed update paths obey it is a separate implementation question *(unverified)*. An observed increase caused solely by a shadow signal would refute implementation conformance, not the algebraic statement. Neither conformance nor a low confidence value establishes calibration.

Avoiding record-count rewards removes one obvious incentive to inflate logging, but does not establish Goodhart resistance. Outcomes such as disputes resolved, errors prevented, or legitimate claims blocked still require independent labeling, a counterfactual baseline, and an explicit cost model.

[46] Author-maintained host-session record, verified 2026-09-22: GRIP#6176 and GRIP#6193.

Candidate control	Required evidence and remaining limitation
CI evaluation	Authenticate the runner's result and bind it to the artifact and test version. Passing establishes the exercised tests, not every intended property.
Human approval	Authenticate an authorized person's approval of the exact artifact. A `human:` prefix or a copied genesis identifier is insufficient.
Cross-provider review	Bind the review to the artifact and document evaluator selection. Provider separation alone establishes neither independence nor correctness.
Pre-action commitment	Obtain an independent receipt before the relevant action or outcome. A writer-supplied timestamp alone does not establish ordering.

Table 18: Candidate external controls, not sufficient conditions for epistemic validity. Their effectiveness requires separate evaluation.

Measured watcher limitation. The pre-hoc watcher measurement of 2026-09-10 used seed 0, five stratified folds, a 50-event window, and 76 paired rows, with a base rate of 13.2%. No predictor's precision interval clears that base rate. The watcher remains `WARN`-only and its hook is unbound.[47] This result does not support using the watcher as an acceptance gate or claiming demonstrated predictive benefit. Further evaluation should publish the labels, fold construction, confusion matrices, precision and recall uncertainty, and baseline comparison. No positive result is inferred from the current warning mechanism.

11.3 External evaluation as a policy for closed loops

The earlier "exogenous-anchor law" is better stated as a governance policy. A deployment may require authenticated external evaluation before accepting a change produced by an improvement loop. That requirement is neither a necessary condition for every correct outcome nor a sufficient condition for correctness. A self-evaluated loop can sometimes succeed; an externally evaluated loop can still overfit weak tests or manipulate its evaluator.

For a loop L with producer A, the proposed policy requires an external acceptance check on the candidate being accepted. It does not merely require that some unrelated edge in the loop has an external label. Evaluator selection, repeat-query budgets, artifact binding, test confidentiality, and authority to change the policy all affect the strength of this control.

Root prefixes such as `ci:`, `human:`, and `council:` classify claimed evidence types. They do not authenticate an issuer. Likewise, the literal `council:context-chain-genesis` supplies a structural starting point, not authenticated origin. An authorized external attestation should bind issuer identity, artifact digest, policy version, result, freshness information, and signature, with authorized keys pinned outside agent-controlled configuration. Enforcement of this full attestation requirement is not established here *(unverified)*.

Temporal boundary and anti-HARKing. Before the addition of an external time anchor, the anti-HARKing property bound only a third party without the signing key; it was not established against the writer. A key holder could create a commitment after learning an outcome and supply an earlier signed timestamp. RFC 3161 trusted timestamps were added in GRIP#6190, "the external

[47] Author-maintained host-session record, verified 2026-09-22: watcher measurement dated 2026-09-10.

time anchor we lacked," merged on 2026-09-17 at 17:22Z.[48]

RFC 3161 support makes an independent timestamp available; it does not show that every hypothesis used it before its relevant action. Writer-resistant preregistration requires the commitment bytes, validated timestamp evidence, an independently evidenced action or outcome boundary, and a policy for replaced or withheld commitments. A timestamp establishes existence by its attested time under the timestamp authority's assumptions, not when an external action actually occurred.

Hypothesis confirmation is not calibration. Let ρ denote the fraction of registered hypotheses confirmed within a stated window. This describes a selected hypothesis portfolio, not probabilistic calibration. There is no general justification for declaring $0.30 \leq \rho \leq 0.70$ a calibrated band: a precise system may legitimately confirm more than 70% of its predictions. Calibration requires probabilistic forecasts, defined outcomes, and appropriate scoring or calibration analysis. Whether every pull request has a timely, independently witnessed hypothesis remains *(unverified)* in this section.

11.4 Limitations

Commitment domains matter. The semantic forest root and the production ledger checkpoint are different commitments. A semantic address that excludes `id`, `ts`, and `audit` does not bind changes confined to those fields. A writer can change an excluded timestamp and re-sign without changing that semantic root. Thus "any node change moves the root" is not a valid general claim. Writer-resistant envelope integrity requires a commitment covering complete envelopes and their order, not merely semantic projections. A separate domain-separated envelope digest and explicit checkpoint manifest remain appropriate future work where those bindings are absent.

What the public Bitcoin evidence covers. The production decision-chain checkpoint covers 501 records. Its root is

$$R = \texttt{0bba0792bf9f71caa815d0c42daf27f45}\\ \texttt{baf07ef9749523be666f103a7f1bcb4},$$

where the two displayed strings are concatenated. The manifest SHA-256 is

$$M = \texttt{5d5e8854f23eb50e583e02b380f210596}\\ \texttt{51a3a4e99b4454cb221d355725a45b9}.$$

An OpenTimestamps proof attests this root in Bitcoin blocks 956991 and 956992, whose Merkle roots begin `1b42ef4d` and `b3f15c01`, respectively. A second stamp of the same digest completed in blocks 956948, 956963, and 956970. The anchored leaves are the exact ledger lines in chain order, not sorted content addresses.

The proof, manifest, 501 RFC 6962 leaf hashes, and standard-library verifier have been public since 2026-09-22.[49] The command `python3 verify.py -explorer` reports `ROOT`, `PROOF`, and `CHAIN` separately, and reports `VERIFIED` only when all three pass.

The records themselves are not public because their subjects name people. Public readers can check the released leaf-hash commitment and timestamp evidence, but cannot independently

[48] Author-maintained host-session record, verified 2026-09-22: GRIP#6190; implementation files `lib/rfc3161_anchor.py` and `lib/continuity_chain.py`, under `/Users/void/.grip/`.

[49] Public anchor bundle: `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. See `verify.py` and the accompanying manifest, leaf hashes, and timestamp proofs.

recompute those hashes from undisclosed ledger lines or inspect their contents. This public hash-and-timestamp verification does not make an HMAC record publicly authenticatable. The explorer-based chain check also depends on the explorer's chain data; independent validation requires an appropriate Bitcoin validation and confirmation policy. The public bundle does not establish whether the evaluated council record is among the 501 committed ledger lines, nor does it establish continuous anchoring of the semantic forest, belief chain, or claim receipts. Historical inclusion is not proof of currentness.

Binding is not truth or completeness. A verified context report may misdescribe the agent's internal state. An authenticated model roster records claims about participants; it does not itself prove which provider executed a request. A matching quotation can be irrelevant to the associated claim or appear in a fabricated source snapshot. The construction permits inspection of authenticated statements, not recovery of exactly what happened.

Local integrity also does not establish complete decision–belief–receipt composition. Optional, missing, or dangling references cannot satisfy a separate complete-composition requirement. A single released bundle containing all three legs and a checker that authenticates every required link remains necessary future work; the council seal, IDR, and root are not substitutes for those three legs.

Recording completeness and freshness remain separate requirements. The evaluated council path permits hash-only completion when IDR recording is unavailable. Mandatory paired recording is therefore a deployment requirement, not a demonstrated invariant. A fail-closed action gateway, crash/retry semantics, and reconciliation against independently observed side effects would be needed to establish complete mediation. Omission of either required record must fail a paired-record requirement.

A signed timestamp authenticates a claimed time, not clock correctness. Currentness requires authenticated head information, a freshness policy, checkpoint discovery, and extension evidence. A stale but valid record can pass local integrity checks. Without the additional evidence, freshness is unknown.

Selective disclosure is not a hiding guarantee. Merkle inclusion proofs avoid transmitting other record bodies, but reveal authentication-path hashes and structural metadata. Unsalted hashes of predictable records can permit dictionary testing. The private status of the 501 ledger bodies should not be interpreted as a proof that the released hashes reveal nothing about them.

Verification capacity is distinct from adoption. A reproducible verifier remains available even if nobody uses it. Lack of independent verification is evidence of low adoption, not a refutation of the capacity to verify. Conversely, repeated invocation does not establish that the predicate is adequate: the historical `PROVEN` failure illustrates this distinction. The public anchor bundle is not a complete self-verification bundle for this paper's LaTeX inputs, fixtures, and implementation.

Reasoning traces are diagnostic artifacts, not causes. A generated reasoning trace is another model-produced report. It may help an analyst identify an error, but it does not establish the internal computation that caused an answer. The proposed diagnostic extension (`lib/thinking_trace.py`) should keep such artifacts outside authoritative record commitment domains and acceptance inputs. A trace may have its own content address without belonging to those domains. The reported isolation property and the value hypothesis identified as `H720` remain *(unverified)* here; neither a general benefit nor structural prevention of reuse is established by this section.

No fifth control is established. The earlier specification's reference to five external controls while listing four is an unresolved specification issue, not evidence for an additional mechanism (`exogenous-anchor-law.md`). No fifth control is claimed here. Any proposed addition needs a threat model, authenticated evidence, and tests.

Costs and gating outcomes are unmeasured. No measured latency, storage, contention, or anchoring-cost result establishes that recording is negligible relative to deliberation. Operation counts and zero additional model calls do not answer that question. Whether recording prevents otherwise-lost disputes, or whether provenance gating has positive net value, remains *(unverified)*. Quote checking can reject legitimate material while accepting irrelevant quotations; gating effectiveness requires labeled outcomes and explicit error costs. Policy should not automatically widen a gate merely because it rejects claims, nor treat a failed quote check as proof of falsity.

Commercial scope and bounded differentiation. Four non-Anthropic councils, with zero Anthropic calls and recorded under seals `7dc01f0b739be243` and `9fb9b0fc0f2d4309`, judged the integrated system "a bundle of commodities with a nice story." This refutes the commercial moat claim in the reported evaluation.[50] The individual mechanisms are shipped by Sigstore, Rekor, `git commit -S`, OpenTimestamps, and KERI. Asqav is MIT-licensed, signs with the same ML-DSA-65, and ships RFC 3161 timestamps and RFC 8785 canonicalisation.

We claim no new cryptographic primitive, exclusive mechanism combination, or demonstrated commercial moat. The only surveyed differentiator is shipping the attacks alongside the claim. That conclusion rests on reading four vendors' documentation, not running their products; it is not an exhaustive novelty finding. The contribution must therefore be judged by the attacks and reproducible outcomes actually released, not by promises of a future corpus. EU AI Act Article 12, in force from 2 August 2026, creates compulsory demand for automatic record-keeping, but does not establish demand for this particular implementation or its compliance.[51]

11.5 Consolidated falsification conditions

Table 19 separates concrete counterexamples from proposed experiments. The entries are requirements for evaluating the bounded claims, not a claim that all listed tests have been run. Implementation evidence should identify a revision and released fixture; code references should identify files or symbols, such as `lib/idr_forest.py::verify_chain_integrity`, rather than stale line ranges.

Future economic and loop-evaluation studies should specify practical thresholds before observing results and report uncertainty separately from effect size. A standardized effect size is neither statistical significance nor a universal measure of operational importance. Likewise, forecast calibration, watcher precision, and hypothesis confirmation rate answer different questions.

The resulting position is narrower than a claim of cryptographically established causation: specified bindings can be checked, failures can be exposed, and some historical commitments can be independently timestamped. Actual conduct, faithful context capture, complete recording, authenticated external authority, currentness, and net benefit each require additional evidence. Publishing the known failure alongside these boundaries is part of the contribution, not an exception to it.

[50] Author-maintained host-session record, verified 2026-09-22: four non-Anthropic council evaluations, the two listed seals, and the associated vendor-documentation survey.

[51] Regulation (EU) 2024/1689, Article 12; applicability date and commercial comparison recorded in the host-session verification of 2026-09-22.

Claim or proposed study	Counterexample or required evaluation
Authenticated council binding	Replace the payload and adjacent hash together. Acceptance as authenticated without checking an independent binding refutes the claim. This failure occurred on 2026-09-17.
Positive verification	Remove keys, libraries, required signatures, or referenced records. Confirmation of a profile requiring them refutes fail-closed acceptance.
Full-envelope commitment	Change an excluded timestamp and re-sign. An unchanged semantic root refutes full-envelope sensitivity, not semantic-projection integrity.
Authenticated external origin	Supply a permitted root prefix or fixed genesis without an authorized attestation. Acceptance as externally authenticated refutes the origin claim.
Writer-resistant preregistration	Create a backdated commitment after observing the outcome. Acceptance without independent temporal evidence refutes pre-action ordering.
Complete composition and recording	Omit either paired record, alter a required cross-link, or exercise an unrecorded side-effect route. Acceptance refutes the corresponding completeness claim.
Freshness	Serve a valid stale checkpoint. Reporting currentness without fresh head and extension evidence refutes the freshness claim.
Shadow-update conformance	An update solely from self-telemetry produces $c' > c$. This refutes implementation conformance to Proposition 11.1.
Watcher utility	The measured precision intervals do not clear the 13.2% base rate. A positive utility claim needs additional evidence, not promotion of the present warning.
Net benefit of recording or external review	Preregister workload, baseline, horizon, independent labels, and error costs; then estimate absolute costs and benefits with uncertainty. Not yet established.
Attack-backed contribution	Missing fixtures, nonreproducible outcomes, or guarantees broader than the released attacks weaken or refute the stated reproducibility claim.

Table 19: Observed failure, proposed negative tests, and unresolved empirical evaluations. No universal confirmation-rate band or record-count threshold is used as a substitute for the relevant predicate.

A Consolidated falsification conditions

Cryptographic causation is presented in the Popperian mode: each claim is stated with a specific observation that would refute it. Table 20 consolidates the central conditions; the per-section *Falsification* paragraphs give the full set. A single confirmed row, read under that row's declared verification profile, protected message domain, trusted inputs, and evidence availability, falsifies the corresponding claim; namespace acceptance, re-authentication by an authorized key holder, or changes outside the protected domain do not by themselves falsify local authentication.

Claim	Falsified if	Anchor
Local authentication	a changed authenticated message passes with its original authenticator under the required trusted key and available prerequisites	§3.3
Content-address stability	two records differing only in `id`/`ts` yield different addresses	§3.2
Authenticated external origin	producer-authored evidence is accepted as externally authorized without a valid attestation under verifier-controlled authorization	§3.4
Canonicalisation idempotence	$\mathcal{F}(\mathcal{F}(F)) \neq \mathcal{F}(F)$ for some forest F	§3.5
Selective-disclosure soundness	a non-member address plus a proof reproduces the root	§3.7
Replay determinism	two replays of an unchanged forest differ in `replay_digest`	§3.8
Belief–decision binding	a changed authenticated `records_idr` value passes with its original authenticator, or an unresolved required target is reported as complete composition	§4
Claim grounding	a receipt reports grounding 1.0 while a cited quote is absent from its source	§5.1
Advisory monotonicity	a model vote moves a citation `not_found` → VERIFIED	§5.3
External-acceptance policy conformance	acceptance is granted without the authorized external evidence required by the declared policy	§6.1
Complementarity	pipeline- and agent-provenance are shown to be substitutes	§10.5

Table 20: Consolidated central falsification conditions.

B Reproducibility and self-verification

A self-verification release of this paper is not established by the material supplied here *(unverified)*. Such a release would require an immutable source archive, an explicit assembly procedure, and a detached manifest that defines the exact hash targets without self-reference — a document cannot embed its own final digest. The `(sha256 stamped on release -- see Appendix B)` placeholder in the source is therefore exactly that: a placeholder, not a digest; an OpenTimestamps proof of this paper's source remains *(unverified)* in the supplied material.

The public 501-record production anchor described in Section 8 is a separate artifact. It is evidence about that ledger commitment, not evidence that this paper's source has been timestamped. The proof, manifest, 501 RFC-6962 leaf hashes, and standard-library verifier have been public since 2026-09-22 at `https://github.com/CodeTonight-SA/grasp/tree/main/docs/tmif/anchor`. The established procedure, `python3 verify.py -explorer`, reports `ROOT`, `PROOF`, and `CHAIN` separately and reports `VERIFIED` only when all three pass. The leaves commit to exact ledger lines in chain order, not sorted content addresses. The records themselves are not published because their subjects name people; this procedure does not authenticate undisclosed record contents.

For a future paper timestamp release, the reader would run `ots verify` on the detached proof against the assembled source archive, with the chain-data source and validation policy documented; `ots info` only inspects a proof, and a pending calendar attestation is not a completed Bitcoin attestation. A complete paper-specific decision–context–receipt bundle and its verification procedure likewise remain *(unverified)*; the reader is handed the refutation conditions in Appendix A, not yet

the artifacts to run them against.

References

[1] R. C. Merkle. *A Digital Signature Based on a Conventional Encryption Function*. Advances in Cryptology — CRYPTO '87, LNCS 293, Springer, 1988.

[2] S. Haber and W. S. Stornetta. *How to Time-Stamp a Digital Document*. Journal of Cryptology, 3(2):99–111, 1991.

[3] B. Laurie, A. Langley, and E. Kasper. *Certificate Transparency*. RFC 6962, Internet Engineering Task Force, 2013.

[4] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008.

[5] P. Todd. *OpenTimestamps: Scalable, Trustless, Distributed Timestamping on the Bitcoin Blockchain*. `https://opentimestamps.org`, 2016.

[6] B. Laurie, T. Santoro, P. Anthonysamy, S. de Haas, and A. Mathur. *A Standard for Claiming Transparency and Falsifiability*. Internet-Draft draft-laurie-tmif-01, Internet Engineering Task Force, June 2026. Work in progress.

[7] L. Gunasekara, D. De Silva, N. Mills, H. Moraliyage, and A. Jennings. *Blockchain-Based Provenance for Artificial Intelligence Lifecycle Traceability*. 2026 IEEE International Conference on Industrial Technology (ICIT), Monterrey, Mexico. DOI: 10.1109/ICIT64854.2026.11490281.