

KOREV AI

RESEARCH / TECHNICAL NOTE

From AI Logs to Verifiable Decision Records

Accountable capture, population continuity and hostile verification inside a declared agentic boundary

VERIFIED $\neq$ TRUE

Abstract

AI systems no longer only generate text. They call models, tools and APIs, delegate work to other agents, move files across application states and can trigger actions with legal, financial or operational consequences. Conventional observability is necessary for operating such systems, but observability alone does not answer a harder audit question: can an examiner detect that the history presented later has been altered, truncated, selectively shortened after admission, or left unresolved?

KOREV AI developed and tested a bounded answer inside KOREV Evidence / PRISM-Oracle. The implementation records an admission before execution, commits ordered events and payload digests into a durable journal, requires acknowledgements from a separately keyed witness at declared boundaries, and exports signed decision evidence packs that a verifier classifies as **VERIFIED**, **INCOMPLETE** or **FAILED**.

The central claim is not that the system proves truth. It is narrower and testable: once an admission has crossed a declared instrumented boundary and has been acknowledged by the configured witness, that admission must remain represented and reconcile to a terminal state before the population can be classified as complete.

This work goes beyond a component protocol. Runtime adapters cover selected model, tool, UI and HTTP boundaries, and the application path from Conversation to Dossier has been hardened against accepted-file substitution, concurrent admission races, uncertain execution, late acknowledgements, unsafe historical active content and orphaned uploads. Those application corrections were subjected to a hostile review in PR #30 and added to blocking CI.

The result is best understood as **accountable capture within declared boundaries**. It is not certification, regulatory conformity, proof of factual correctness, proof of human approval, or a claim that every possible action in production is captured.

1. The problem: a history can be internally consistent and still be incomplete

A conventional log answers “what did this system record?” An audit asks something stronger: “what should have been in the record, and can the party producing it silently remove part of the history?”

Per-record signatures do not solve that problem by themselves. An operator can delete an inconvenient tail, rebuild a new internally valid history and sign it again with a key it controls. Hash chaining detects an edit relative to a known chain, but it does not by itself tell an external examiner whether the chain was shortened before being presented.

That leads to three distinct properties:

1. **Record integrity** — the bytes presented later are the bytes that were committed.
2. **History continuity** — the ordering and linkage of events have not been silently rewritten or truncated relative to a retained reference.
3. **Population completeness after admission** — every witness-recognised admission is still represented and reconciled before the population can be accepted as complete.

The third property is the focus of this work.

The study asks:

Can an operator that controls its runtime and local journal provide an examiner with evidence that detects targeted alteration, truncation, post-admission omission and incompleteness, without requiring the examiner to trust the producer's local database alone?

For the declared and instrumented path, under the threat model and limitations stated here, the implementation and experiments support a conditional answer: **yes**.

2. What KOREV actually built

The work is not one cryptographic primitive. It is a chain of controls that must agree.

2.1 Durable admission before execution

In required-capture mode, selected admissions and operations are persisted before execution. The local transaction stores identity, input metadata and payload commitments before the runtime proceeds.

A crash between admission and completion therefore leaves an unresolved admission rather than an invisible action.

2.2 Ordered decision journal

Events carry a protocol/scope identifier, sequence number, previous-event hash, outcome metadata and a digest of separately retained payload bytes.

The local journal is append-only by API contract, but KOREV does **not** describe the underlying SQLite database as physically immutable. A producer able to rewrite local storage is explicitly part of the threat model.

2.3 Separately keyed witness

A witness process holds a distinct signing key and persistent state. It accepts only an exact extension of the history it already recognises and returns a signed checkpoint binding the scope, sequence, current head, time and challenge.

In the reported experiments the witness is process-separated and key-separated but remains under the same local administration. The architecture supports separate administration; the experiments do not claim operational independence.

2.4 Freshness and retained references

A verifier does not accept an old signed checkpoint merely because the signature is valid. Fresh challenges and current witness state are checked. A historical bundle that was valid at an earlier point can fail against a newer witness state.

2.5 Reconciliation, not disappearance

Admissions can terminate as success, refusal, error, cancellation, superseded state or another explicit terminal outcome. An interrupted admission remains visible as unresolved until it is reconciled.

The protocol therefore distinguishes “nothing happened” from “an admitted operation did not reach a known terminal state.”

2.6 Release-boundary capture

The runtime extends capture to selected output boundaries. Model/provider results, tool results and errors, UI-visible output and selected HTTP response bytes can be persisted and acknowledged before release.

This proves what the instrumented boundary offered. It does not prove downstream receipt.

2.7 Signed export and independent verification surface

The exported evidence bundle combines the ordered event transcript with a decision evidence pack whose producer signature binds the terminal-record inventory and payload digests. The witness checkpoint separately binds the recognised history head. Trust keys and expected scope are provided separately to the verifier rather than accepted from inside the bundle being verified.

The verifier returns one of three technical verification states:

- **‘VERIFIED’** — inventory, bytes and current witness state reconcile and no required execution remains unresolved;

- **`INCOMPLETE`** — freshness or reconciliation is missing, or an admitted operation remains open/unknown;
- **`FAILED`** — signature, scope, sequence, hash, freshness or inventory verification fails.

These statuses are deliberately not regulatory labels.

3. The claimed property

The central property is:

For the configured and instrumented runtime boundary, every admission acknowledged by the configured witness must remain represented in the witnessed history and reconcile to a terminal state before the exported population can be classified as `VERIFIED`.

This does not assert that every real-world request reached the boundary, every remote side effect was captured, every source was true, every human-review field corresponds to an actual review, or every internal code path is instrumented.

It also does not assert certification, legal correctness, semantic correctness or global production exhaustiveness.

The point of the property is that it is falsifiable: if a witness-recognised admission disappears, remains unresolved, is substituted, or no longer matches the retained witness state, the verifier must not silently return VERIFIED.

4. Threat model

The producer may control and rewrite its local database and may re-sign producer-side bundles.

The witness has its own key and state. The verifier receives expected trust material outside the producer-generated bundle.

The model explicitly considers:

- local history rewrite;
- truncation;
- omission of admitted items;
- replay of an old receipt;
- stale or future receipts;
- wrong witness key;
- producer-side re-signing after local rewrite;
- loss of final acknowledgement;
- process interruption;
- concurrent writers;
- missing mandatory evidence;
- inventory substitution.

Outside or only partially covered:

- requests that never reached an instrumented boundary;
 - deliberate bypass of wrappers/adapters;
 - future uninstrumented paths;
 - compromise of producer and witness together without a previously retained external reference;
 - semantic truth;
 - certified timestamps;
 - public multi-auditor non-equivocation;
 - complete key rotation/revocation;
 - exhaustive network capture;
 - exactly-once remote side effects.
-

5. Protocol lifecycle

5.1 Admission

The admission is written transactionally before execution. SQLite is configured with `synchronous=FULL` for this path. A witness acknowledgement is then required before execution proceeds in required mode.

5.2 Execution

Provider/model/tool work occurs only after the admission has crossed the configured capture gate. Remote systems can still produce side effects between intent and result; this protocol does not claim exactly-once semantics.

5.3 Terminal outcome

The result, error, cancellation or other terminal state is recorded. A missing terminal outcome remains visible as an incomplete chain rather than being inferred away.

5.4 Release

Selected outputs are persisted and acknowledged before release at declared boundaries. Storage or witness failure in required mode blocks further execution/release for the affected chain.

5.5 Export

The exporter constructs a decision evidence pack from the retained journal and payload material, signs the pack and emits a verification report.

5.6 Verification

The verifier checks:

4. expected scope and externally supplied trust keys;
5. witness signature and challenge;
6. freshness;
7. alignment of witness head and presented transcript;
8. contiguous sequence and previous-hash linkage;
9. payload bytes against recorded digests;
10. admission-to-terminal reconciliation;
11. evidence inventory and producer signature.

A pack can therefore be structurally valid yet `INCOMPLETE` or `FAILED` relative to the current witness state.

6. Experimental evidence

The component study uses a synthetic recommendation workflow and the repository's real critical-output finalisation path. It avoids external LLM calls so that the hostile cases remain deterministic.

Two recorded batches contain 10 and 100 recommendations. Each batch executes 11 predetermined scenarios. Across 22 recorded observations, observed classifications match the expected classifications defined by the runner.

The targeted v2 suite contains 67 tests: 24 inherited from the earlier pack work and 43 added for durable capture and witness behaviour.

The tested cases include witness restart, concurrent writers, storage failure, abrupt process termination, lost acknowledgements, missing evidence, fully re-hashed local rewrites, stale/future receipts, wrong witness keys, old receipt under fresh challenges and old bundles evaluated against a newer witness state.

6.1 Hostile scenarios

Scenario	Expected result	Property exercised
intact live checkpoint	VERIFIED	current witnessed population reconciles
reconciled cancellation	VERIFIED	interruption is closed explicitly
missing witness	INCOMPLETE	freshness/continuity cannot be established
interrupted admission	INCOMPLETE	admitted work lacks a terminal outcome
stale checkpoint	FAILED	old receipt is not silently accepted
old receipt under fresh challenge	FAILED	a challenge-bound receipt cannot satisfy a new challenge
truncated history	FAILED	witness head and presented history diverge
rewritten admission	FAILED	event commitment no longer matches
omitted refusal	FAILED	population reconciliation fails
invalid exporter signature	FAILED	producer bundle authentication fails
old bundle vs newer witness	FAILED	historical validity is not current validity

These results establish detection for the specified attacks and boundary only.

7. From protocol to application boundary: Conversation → Dossier

A verifiable protocol is weakened if the application can hand the wrong bytes into the protected path. PR #30 therefore treated the Conversation → Dossier transition as part of the evidence boundary rather than as “just UI.”

The initial audit identified nine findings. The correction report then recorded eleven DEF items found during hostile re-review; all were corrected in the scoped change. The resulting changes include:

- staging outside the active dossier context until admission is accepted;
- an ordered accepted-file manifest with SHA-256 commitments;
- re-checking staged bytes before publication so an extension cannot modify them after hashing;
- admission under lock with deterministic handling of busy/drain/concurrency states;
- idempotence tied to server-side JSON plus ordered file manifest, not filename/size alone;
- explicit pending, claimed and accepted launch states;
- retryable deterministic refusals, while uncertain execution is not automatically replayed;
- purge of active uploads on deletion and protection against stale-context recreation;
- late acknowledgements reconciled back to the originating draft without overwriting newer user edits;
- safe rendering of historical content, with HTML escaped and executable links/images filtered;
- preservation of text and download affordances without inventing approval or validation states.

This work matters because “the journal is cryptographically sound” is not sufficient if the application can silently swap the file set before admission or infer a stronger historical verdict than the evidence contains.

The PR #30 report explicitly retains limitations: there was no authenticated production deployment test, backend and UI must be deployed together, legacy receipts without file digests need explicit review, and deletion of active uploads does not automatically purge external backups or independent memory systems.

8. CI and validation record

8.1 Protocol/publication freeze

PR #29 was merged into `main` as `f7cfe7ede525d5ba1b01f8f44ea8b2b54fe759eb`.

Before merge, the final release-candidate head `763be256836afd04156bc9308b79dd1a93c42c83` completed all four principal workflows successfully:

- Evidence outputs and execution: **SUCCESS**;
- Runtime capture: **SUCCESS**;
- Main Gate: **SUCCESS**;
- general tests workflow: **SUCCESS**.

The final cleanup removed a transitive PyMuPDF/AGPL dependency from the normal environment by packaging KOREV's existing ReportLab compatibility shim as the local `markdown-pdf==1.5` distribution and adding fail-closed checks that reject `fitz` / `pymupdf` if they reappear.

The legacy PyMuPDF parity suite is retained as optional historical migration coverage rather than pretending that a deliberately removed dependency should exist in the production test environment.

8.2 Post-freeze application hardening

PR #30 was merged into `main` as `13524a13bfaa1d312c84fa7dcc588ede742b6388`.

On its final head `7ac2d9f622eabe05b0ced9dbe1b651a10b40207c`, the reported blocking checks were green, including Core Tests/security, blocking gates, Conversation & Dossiers, rendering/publication, WebUI and runtime capture.

The PR records:

- 364 core tests in the blocking Core job;
- 476 security tests with 8 expected skips;
- 5,973 passing hermetic unit tests with 46 skips;
- 61 passing extended E2E/integration/infra/property tests;
- 121 passing UI tests in the local targeted validation;
- 17 passing targeted admission regressions;
- 2 Chromium journeys at desktop/mobile widths;
- output-view coverage of 97.93% lines, 92.12% branches and 95.12% functions.

The same PR also discloses that the optional Extended Tests (Warning) job still reported 22 failures among 5,322 passes, principally OCR/legacy-report assertions, and that the full-report suite reached an asyncio timeout also observed on `main`. Those non-blocking results are not hidden or represented as a globally perfect product suite.

9. The counterexample that defines the boundary

The suite deliberately retains a semantically false source. If the originally admitted bytes contain a false statement and those exact bytes remain intact, the evidence layer can still classify the record as `VERIFIED`.

That is intentional.

Integrity answers whether evidence changed. It does not answer whether the evidence was true.

Likewise, the presence of a `human_review`, `sources` or `consensus` artifact proves that a required documentary role contains integrity-bound bytes. It does not prove that the source was authoritative, the consensus was reasonable, or a human genuinely performed the review.

For that reason the classification is not a contract of business sufficiency.

10. Runtime boundary and application boundary

The current implementation instruments declared boundaries including:

- top-level agent admissions and task execution;
- model attempts, streams and embeddings;
- selected direct providers;
- tool arguments, results and errors;
- UI-visible output;
- selected HTTP request/response bytes;
- accepted file sets crossing from Conversation into Dossier.

Important exclusions remain:

- requests that never reach an instrumented boundary;
- raw internal shell/SSH/browser behaviour beyond captured tool boundaries;
- remote state that is not returned through a captured boundary;
- future adapters that bypass capture;
- client receipt of offered HTTP bytes;
- semantic reasoning hidden inside a model;
- uninstrumented external machines.

The accurate description is **capture at declared instrumented boundaries**, not “capture of everything the system does.”

11. Relationship to gateway-centred trust planes

Traefik Labs publicly describes a gateway-centred Sovereign Trust Plane that spans model, MCP and API traffic, commits hashes of gateway/access logs into an append-only Merkle log, and uses signed checkpoints plus witnesses. KOREV has not benchmarked or hostile-tested Traefik's implementation and makes no superiority claim.

The technical scopes are different enough to be useful to distinguish:

Dimension	KOREV work described here	Traefik public description
Primary observation point	declared agent runtime and application boundaries	gateway across model/MCP/API traffic
Continuity primitive	ordered hash chain + witness-recognised checkpoints	append-only Merkle log + signed checkpoints/witness
Central audit property	population reconciliation after witness-recognised admission	integrity/inclusion of gateway records and witnessed history
Application state	explicit admission/terminal reconciliation and Conversation→Dossier handoff	gateway decision/policy/identity record
Evidence payload	retained payload bytes + digests + signed pack	original logs retained by customer; hashes committed
Status model	VERIFIED / INCOMPLETE / FAILED	verifier/checkpoint/witness validation

These approaches are not treated as equivalent. Gateway evidence concerns communication boundaries visible to the gateway. KOREV's current work focuses on accountable admission and reconciliation inside a declared runtime and on the application handoff that determines which bytes enter the protected record.

12. Why a separately retained witness matters

A producer that controls both a log and its signing key can create a shorter self-consistent history and sign it again. A separately keyed witness changes the question from “is this history internally consistent?” to “does this history extend the state already retained elsewhere?”

That still does not eliminate all trust. If producer and witness are compromised together before an external reference is retained, the protocol cannot reconstruct a lost truth from nowhere.

This is why the roadmap includes stronger external checkpoint retention, separate administration and multi-auditor/non-equivocation work.

13. Verification without publishing the private repository

The development repository remains private. The publication model is designed so that the property can still be challenged without exposing unrelated proprietary code.

The intended public reproduction pack includes:

- the manuscript and claim ledger;
- the protocol/threat model;
- a minimal verifier sufficient for the published fixtures;
- synthetic fixtures and expected verdicts;
- public verification keys;
- raw observations and digests;
- exact dependency versions;
- reproduction commands;
- immutable private-source reference identifiers;
- a SHA-256 manifest of the public release pack.

It must exclude private keys, credentials, customer/employee data, production databases, proprietary prompts unrelated to the proof and internal infrastructure details.

Until an external party actually reruns the material, the correct claim level remains “reproduction materials provided,” not “independently reproduced.”

14. Regulation and engineering context

The EU AI Act establishes automatic logging and traceability obligations for high-risk AI systems and log-retention duties in defined circumstances. Those requirements make trustworthy records operationally important, but they do not prescribe this protocol and do not make a successful KOREV verification result equivalent to legal compliance.

Sigstore/Rekor and SLSA provide established engineering references for explicit provenance, verification and trust-root separation in software supply chains.

Traefik's September 2026 Sovereign Trust Plane announcement illustrates that AI-agent evidence is also being addressed at the infrastructure layer, rather than only as application logging. Its public materials are used here as industry context, not as a benchmark result.

15. What this work demonstrates

Subject to the declared boundary and threat model, the implementation and recorded tests support these statements:

1. witness-recognised admissions must reconcile before a population is classified as VERIFIED;
2. the tested alteration, truncation, replay and omission scenarios are detected;
3. interrupted admissions remain representable rather than disappearing;
4. historical validity is not silently treated as current freshness;

5. trust material can be supplied separately from the producer bundle;
 6. selected outputs can be captured before release at declared runtime boundaries;
 7. the Conversation→Dossier handoff now binds accepted file bytes and admission state more tightly to the protected record;
 8. external examination can be supported without publishing the entire private source tree, provided the agreed reproduction material is released.
-

16. What this work does not demonstrate

It does **not** demonstrate:

- certification or legal conformity;
 - factual truth or model correctness;
 - business adequacy of a decision;
 - effectiveness of human review;
 - global completeness outside declared boundaries;
 - operationally independent witness administration in the reported experiment;
 - protection against coordinated producer+watcher compromise without an external retained reference;
 - exactly-once remote effects;
 - production-scale latency/throughput qualification;
 - authenticated production deployment of the PR #30 Conversation→Dossier changes;
 - independent third-party reproduction;
 - superiority to Traefik, Sigstore, SLSA-based systems or another product.
-

17. Open work

The strongest next steps are:

1. deploy the witness under separately controlled administration and key custody;
 2. add external checkpoint retention and stronger non-equivocation guarantees;
 3. publish the redacted reproduction pack and obtain an external rerun;
 4. qualify required-capture latency, throughput, storage growth and failure behaviour under load;
 5. exercise selected production-like pipelines with declared capture maps;
 6. formalise key rotation and revocation;
 7. qualify backup, retention, encryption and deletion semantics for evidence stores;
 8. run authenticated staging/production tests for Conversation→Dossier with backend and UI deployed together.
-

18. Conclusion

The useful distinction is not “logs versus no logs.” It is the distinction between **a history asserted by the system under examination and evidence whose declared properties can be challenged by an examiner.**

KOREV's work adds a specific property to that discussion: population continuity after admission. If an admission crossed a declared instrumented boundary and the configured witness acknowledged it, the system must continue to account for it. A later history cannot be called complete merely because the records it still contains are individually well signed.

That claim is deliberately narrower than truth, global exhaustiveness or compliance. It is also more testable.

The application hardening completed after the protocol freeze reinforces the same principle at the point where real user files and state cross from Conversation into Dossier: evidence quality begins before cryptography, at the moment the system decides what has actually been admitted.

For AI systems that contribute to consequential decisions, that is the level at which accountability has to start.

References

1. Regulation (EU) 2024/1689, Article 12 and Article 19, EUR-Lex: <https://eur-lex.europa.eu/eli/reg/2024/1689/oj>
 2. Sigstore, "Security Model — Rekor Security Model": <https://docs.sigstore.dev/about/security/>
 3. Sigstore, "Rekor": <https://docs.sigstore.dev/logging/overview/>
 4. SLSA v1.2, "Requirements": <https://slsa.dev/spec/v1.2/requirements>
 5. SLSA v1.2, "Build: Verifying artifacts": <https://slsa.dev/spec/v1.2/verifying-artifacts>
 6. Traefik Labs, "Traefik Labs Introduces the Sovereign Trust Plane, Bringing Verifiable Evidence to AI Agent Governance", 15 September 2026: <https://traefik.io/press/traefik-labs-introduces-the-sovereign-trust-plane-bringing-verifiable-evidence-to-ai-agent-governance>
 7. Emile Vauge, Traefik Labs, "Agentic Telemetry Is Not an Audit Trail: Why We Built the Sovereign Trust Plane", 14 September 2026: <https://traefik.io/blog/why-we-built-the-sovereign-trust-plane>
-

Publication control

Protocol/publication baseline: f7cfe7ede525d5ba1b01f8f44ea8b2b54fe759eb. Current hardened implementation reference after PR #30: 13524a13bfaa1d312c84fa7dcc588ede742b6388.

Public communication must remain aligned with CLAIMS_LEDGER.md. A new implementation commit must not silently replace these references in the manuscript without rerunning the stated publication checks.