

Troubleshooting Agentic Retrieval: A Traceable Architecture for Scalable, Heterogeneous Vector Fleets

Eric Allione

aigamma.com
eric@aigamma.com

Abstract

Proactive inspection can expose retrieval-system faults before a suspect answer triggers an investigation. This paper presents Codexproof, a tool for capturing and verifying declared retrieval evidence across application boundaries while corpora evolve. Typed lineage links address canonical BLAKE3 content-ids; a Rust/WebAssembly verifier checks bytes, signer authorization, closure and revocation under consumer-chosen anchors. I deployed authenticated capture across ten applications using Pinecone, Supabase pgvector and Qdrant under one operator. Two 300-attempt HTTPS/MCP campaigns produced 597 verified and restored captures. Trust renewal and a scheduled corpus update preserved earlier evidence. A subsequent proactive audit found and resolved expired trust at all ten producers and traced malformed literature passages to PDF extraction. Retained-payload replay measured 7.67 ms median complete consumer verification. Deployment and adversarial review exposed request-schema, trust-activation and cross-language list-selection defects. These observations inform inspection, retention and rollout practices. The tool authenticates declared provenance, not truth, completeness or faithful producer execution.

1 Introduction

My operating maxim is: *By being proactive, the idea is never to need to be reactive.* Applied to retrieval-augmented generation (RAG), this motivates inspection before a reported problem demands investigation. Operators can check trust readiness and captured evidence; when a result looks wrong, they can inspect its declared sources and configuration. Meanwhile, corpora change. Repeating yesterday’s query may return different evidence. Freezing the application impedes development; retaining only provider identifiers leaves earlier bytes unresolved.

Codexproof captures selected evidence on explicit authenticated interrogation and preserves it for later inspection. The consumer is a local library and operator CLI, with no signing key or vector-store credential. It can restore a capture without rerunning retrieval, check its declared origin and dependencies, then inspect a suspect passage. A failed check identifies an evidence or policy problem; successful verification does not distinguish a ranking error from a model reasoning failure.

Copyright © 2027, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

MCP standardizes tool invocation (Model Context Protocol 2026); HTTPS authenticates the current connection. Neither alone binds a cached answer to the exact bytes and typed relations of its declared evidence. Codexproof applies object security (Jacobson et al. 2009; Zhang et al. 2014) to that handoff. Each producer supplies a capture adapter; consumers share a verifier while choosing their own trust authorities. Authenticated logs and snapshots can preserve the same underlying evidence and complement this object; no logging-system comparison or measured integration-effort reduction is claimed.

The signed-link invariant and shared Rust/WebAssembly core are inherited from the AgentNet prototype (Allione and O’Hara 2026). The extension contributes *producer-owned retrieval and sealing with keyless consumer policy*, and a *measured ten-application deployment* with fixed live campaigns, restoration, trust renewal and corpus-update continuity. Request/configuration binding and observed revocation versions constrain acceptance. The original four-store results are excluded from the new totals.

2 Evidence and Consumer Policy

2.1 Canonical nodes and signed links

A node has four fields: *kind*, *name*, *attributes* and *links*. Kind is Entity, Activity or Agent; names are hierarchical. Canonical encoding uses type tags, sorted object keys and length prefixes. Capture adapters retain source-record strings and encode numbers outside the safe-integer profile as strings in typed attributes. Links name a parent content-id and one of five PROV relations (Moreau and Missier 2013): *used*, *wasGeneratedBy*, *wasDerivedFrom*, *wasAttributedTo* and *wasInformedBy*.

Let v be a node, $\text{Enc}(v)$ its canonical bytes, c_v its 32-byte content-id, σ_v (sigma) its signature and sk_v its signer’s private key. Subscript v means *for this node*; sk names a key, not multiplied variables. A key can sign many nodes:

$$\begin{aligned} c_v &= \text{BLAKE3}(\text{Enc}(v)), \\ \sigma_v &= \text{Ed25519}.\text{Sign}_{sk_v}(D||c_v). \end{aligned} \tag{1}$$

BLAKE3 hashes its input (O’Connor et al. 2020); Ed25519.Sign_{sk_v} signs with key sk_v (Bernstein et al. 2012). Parentheses enclose inputs. D is the 12-byte prefix `prov-sig:v1` followed by a newline; `||` appends the raw

content-id bytes. Signing receives 44 bytes, while encoding and hashing process the full node (Figure 1a).

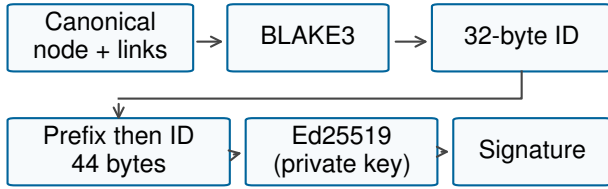
Object-key and typed-link insertion order are ignored; array order, relation labels and link multiplicity remain committed. Reordering distinct passage-CIDs in the result array changes its identifier; reordering the same typed links does not.

Binding invariant. In a finite, closed capture directed acyclic graph whose identifiers verify, a fixed root binds all reachable canonical node bodies and typed links under BLAKE3 collision resistance. Equal root bytes bind equal parent identifiers; repeating that argument along links binds their bodies, as in a Merkle DAG (Benet 2014). Acyclicity follows from the construction, not an explicit verifier cycle test. Surrounding signatures and trust material require separate checks (Supplementary Appendix C).

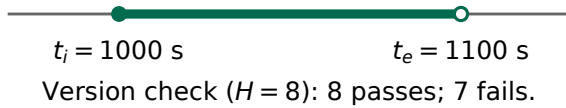
2.2 Authorization and current trust

Integrity establishes that signed bytes are intact; authorization establishes permission to assert their name. The NDN-style schema (Yu et al. 2015, 2023) requires a valid signature and certificate chain to a consumer-accepted anchor, with data and signer names inside its namespace. Each issuer’s name must prefix the name it certifies. Thus an authorized `/alpha/retriever` can sign `/alpha/evidence/e1`; a trusted `/beta` key cannot.

(a) Bind the evidence, then sign its ID



(b) Time and version must both pass



(c) Each hit adds nodes and links

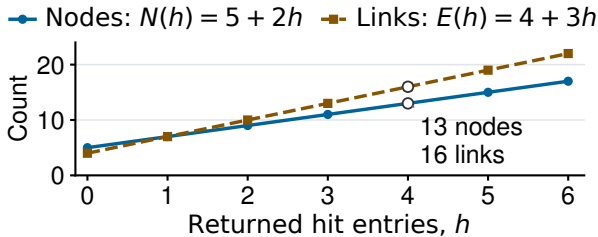


Figure 1: Three rules made visible. (a) The 44-byte signing input binds the ID (Equation 1). (b) Filled and open endpoints include issuance and exclude expiry; time and version gates both apply (Equation 2). (c) Exact construction counts, not timing predictions (Equation 3).

Chain walks have depth and step limits and fail closed on exhaustion. The rule isolates namespaces rather than roles.

An immutable capture can outlive its signer’s authorization. The full consumer requires exactly one valid revocation list signed by each authorizing anchor, revoking neither the node nor its signer. For each anchor, let t be trusted check time, t_i issuance and t_e expiry, all nonnegative integer Unix seconds; subscripts i and e label issuance and expiry. Let Δ be maximum lifetime in seconds, q the signed nonnegative version, and H the highest accepted version for this anchor’s name and key, initially zero. Versions have no time unit:

$$t_i \leq t < t_e, \quad 0 < t_e - t_i \leq \Delta, \quad q \geq H. \quad (2)$$

All three conditions must hold. The operators $\leq$ and $\geq$ include equality; $<$ excludes it. A list is usable at issuance but not expiry. Subtraction gives its lifetime, capped by Δ , which defaults to 86,400 seconds. Freshness concerns the list, not capture age. With $H = 8$, version 8 passes the version gate and 7 fails. Only after complete verification succeeds does the consumer set H to q ; rejection leaves it unchanged. Signature, namespace, request and graph checks still apply. A new consumer has no history; same-version issuer equivocation is not prevented. Lower-level integrity APIs omit this time/version gate.

3 Deployment Across Ten Applications

The September 7, 2026 observation covers ten applications under one operator (Figure 2). Three use Pinecone, two Supabase pgvector and five Qdrant. Existing stores were retained; new reference collections have explicit ingestion and separate query operations. They serve news, interviews, literature and technical or financial references. This is backend heterogeneity, not a vendor ranking or ten independent organizations. Supplementary Appendix A gives each application’s retrieval purpose and vendor rationale.

3.1 Capture, verification and restoration

Six producers use Netlify functions; Leaderlogic uses a Netlify proxy to Node on Fly. Learnrust, Selectsectors and Researchshop use native Rust on Fly. Each exposes authenticated HTTPS retrieval and an initialized MCP tool with the same request contract. A bounded query triggers permitted retrieval, capture construction, leaf-key signing, self-verification and retention before return. The consumer checks request binding, graph reachability, authorization and fresh trust. A later archive request returns that capture without another vector query. Producers accept retrieval requests, not arbitrary signing templates.

Ordinary page, chat/search, ingestion and financial-computation paths do not invoke capture. Source/build checks establish separation, not measured zero overhead or absence of infrastructure contention. Capture requests pay retrieval, sealing and retention costs. An uncaptured ordinary query cannot be reconstructed by later interrogation: that produces a new retrieval.

For investigation, CID or signature failures direct attention to returned bytes and signing identity; request or authorization failures to request construction and trust policy; missing

Explicit provenance interrogation across ten applications

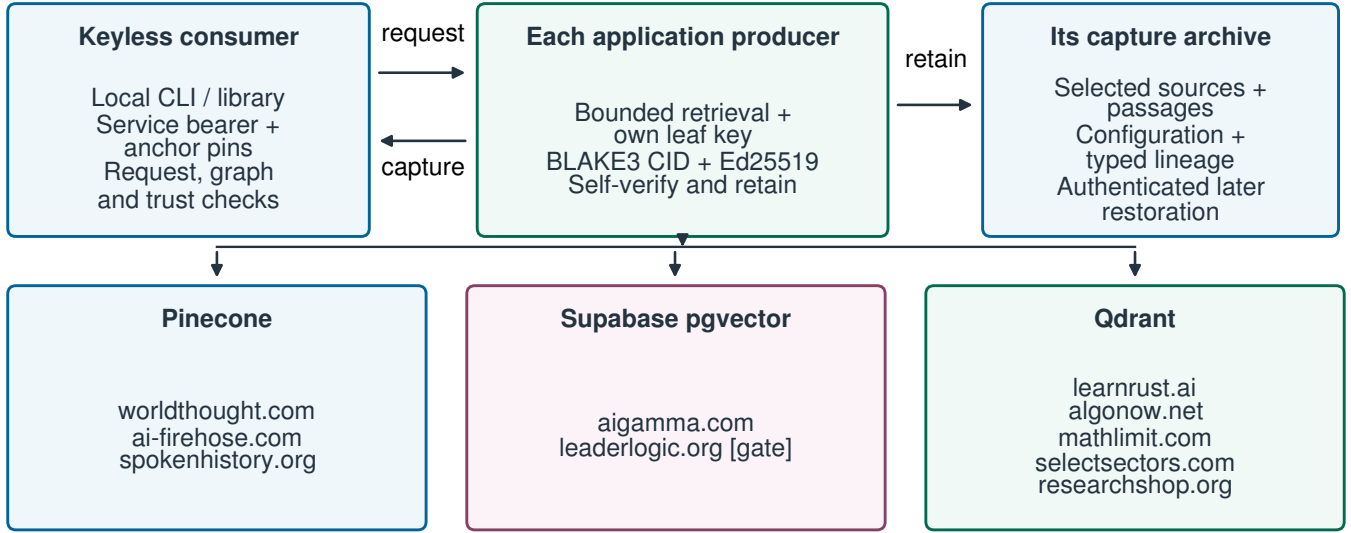


Figure 2: The measured deployment. Each named application has its own producer identity, leaf key and retained-capture storage; the lower boxes group applications by backend technology. An authenticated HTTPS or MCP request invokes retrieval, sealing and retention before returning a capture. The local keyless consumer verifies under its own anchor policy and can later restore the capture without repeating retrieval. Ordinary page, chat and ingestion paths do not invoke capture.

parents to unavailable dependencies. After successful verification, the operator can inspect retained sources, passages and configuration before investigating chunking, ranking or downstream model behavior.

3.2 What a capture preserves

Let h be the nonnegative integer count of returned hit entries. $N(h)$ and $E(h)$ count signed nodes and typed links for that many hits; parentheses mean *for h hits*. The terms $2h$ and $3h$ multiply the hit count; addition combines fixed and per-hit contributions:

$$N(h) = 5 + 2h, \quad E(h) = 4 + 3h. \quad (3)$$

Five nodes record request, configuration, selected-record generation, retrieval activity and result. Four links join the result to the activity and the activity to the three context nodes. Each hit adds a source, a passage and three links: passage to source, activity to passage and result to passage. Four hits require 13 nodes and 16 links (Figure 1c). All fifty captures from the first corrected campaign matched this construction. Counts exclude certificates, revocation lists and CID-valued attributes. Defaults of 1,024 nodes and 3 MiB per response, where 1 MiB is 2^{20} bytes, limit a response, not registered domains. These counts establish neither unbounded capture capacity nor linear latency.

The signed graph binds passage order, request ID, operation, namespace and arguments alongside implementation commit, configuration, access classification and available generation metadata. An outer response cannot replace its signed result. A separate census checks this producer topology; consumer acceptance is broader (Supplementary Appendix C).

Canonical bytes preserve the producer’s selected source strings, not necessarily the upstream HTTP serialization or full document. PDF-derived records retain document digests and page references, but extraction loses layout. Adapters identify unavailable query vectors, unselected candidates and corpus versions. Restricted records retain their access boundaries. Netlify Blobs uses collision checks/readback; other adapters use unique database keys or exclusive files on persistent volumes. These mechanisms do not establish universal write-once storage. Signatures detect alteration but cannot force archive availability.

3.3 Trust activation and corpus continuity

Each producer has a distinct root, directly certified leaf and service credential. Root keys remain in a protected local vault; production receives its own leaf. Renewal retains keys, carries forward revocations and advances signed versions. Consumer history persists across restarts.

All ten restored an earlier signed bundle and root CID unchanged under renewed trust. During rapid activation, three services returned `revocation_not_fresh`; a fourth comparison failure lacked response bytes and has no established cause. HTTP dates lagged by about 25 seconds, consistent with a not-yet-valid list. A Researchshop renewal with a 60-second activation lead passed without backdating issuance or relaxing freshness; this is not a universal skew bound.

Proactive inspection. A separate September 8 audit initialized MCP and invoked one fixed retrieval per application. All ten returned `revocation_not_fresh`; their 24-hour lists had expired naturally. After renewal, the same queries produced ten verified/restored captures; all ten older cap-

tures also verified unchanged. One isolated upload omitted a runtime dependency and required repackaging. The audit retained failed activations and did not alter the earlier campaign totals. It diagnosed trust maintenance before a reported bad-answer incident.

After successful verification, inspection found interleaved PDF-column text in two Researchshop hits. The signed spans and original-file digest located the source page; rerunning the declared extractor reproduced both passages exactly. The diagnosis concerned extraction order, not altered capture bytes. Both passages were reproduced without changing the corpus; this single-page diagnosis does not measure corpus-wide quality.

AI Firehose’s scheduled ingestion changed 4,319 generated data, cache, feed or sitemap paths while runtime and provenance code stayed identical. An earlier capture restored with the same signed bundle and root CID under current trust. The ingestion manifest changed from 4,578 to 4,577 IDs; a separate live Pinecone count was 4,491 vectors. These are distinct views, not interchangeable index snapshots.

The operating lessons are concrete: *retain captured evidence separately from serving data*, since a commit pin cannot supply missing bytes; *validate request schemas before a fleet campaign* (Section 4); and *test trust activation with restoration*, preserving failures and current freshness checks. Pruning an index and retiring a capture are separate operations.

4 Live Evaluation and Retained-Payload Replay

The evaluation separates cross-application capture/restoration, rejection of altered declarations and service versus verification costs. Verification is a protocol outcome, not a relevance judgment or a measure of user decisions.

4.1 Design and failure accounting

Each September 7 campaign pinned identities, source/deployment revisions, consumer/runtime hashes and a fixed schedule: five purpose-specific queries per application, maximum two through six hits (top- k), two transports and three repetitions, giving 300 sequential attempts in interleaved ten-application blocks. MCP negotiated 2025-06-18. Monotonic total time includes response buffering, verification and MCP initialization where applicable. Raw-body persistence and a separately timed archive restore follow. Every success must restore the same signed bundle and verify again. Raw bodies remain encrypted outside Git; public receipts retain hashes, configuration and outcomes.

The initial campaign yielded 269 successes and 31 failures. Thirty requests omitted Spokenhistory’s required null interview filter; the producer correctly rejected them. One Leaderlogic MCP initialization notification returned HTTP 503. An independent JSON Schema validator then checked all 50 corrected query shapes against live MCP schemas; five negative controls reproduced the missing-field rejection.

Application	OK/30	HTTPS	MCP	Verify
Worldthought	30	0.96	1.44	6.44
Aigamma	30	1.19	1.82	5.62
AI-Firehose	30	1.26	2.02	8.53
Spokenhistory	30	1.23	1.82	7.40
Learnrust	30	0.57	0.88	7.26
Leaderlogic	29	1.65	2.66	6.13
AlgoNow	30	1.12	1.78	9.16
Mathlimit	30	1.41	2.00	10.73
Selectsectors	30	0.76	1.28	11.95
Researchshop	30	0.88	1.25	11.23

Table 1: Prospective live replication. Successes include verified archive restoration. HTTPS and MCP are median total seconds; verify is median consumer milliseconds pooled across both transports. Each application has 30 attempts, 15 per transport. Decimal ties round to even.

The corrected campaign yielded 298 verified/restored captures: Selectsectors had one HTTPS retrieval timeout; one Firehose MCP failure has no specific retained cause. Replication repeated the schedule after renewed schema/deployment checks. Worldthought’s scheduled graph update and Spokenhistory’s navigation cleanup changed deployed revisions but not provenance source; both were pinned. No failed trial was replaced.

The prospective replication completed all 300 attempts: 299 verified captures restored successfully, and 1 attempt failed. The failure was 1 Leaderlogic (producer operation failure, HTTP 503). Table 1 reports every application’s success count and latency among successful attempts. Figure 3 separates total interrogation time from consumer verification. Failed operations remain in the denominator; their elapsed time is retained separately and is not treated as successful-request latency.

Verified responses contained 2 to 6 hits and 9 to 17 signed nodes. The serialized signed wrapper ranged from 19.5 to 387.0 KiB; this excludes MCP’s additional transport encoding. Wrapper size includes captured selected records and available retrieval context, not just cryptographic envelope overhead.

Percentiles use linear interpolation. Fifteen attempts per transport/application do not support precise tails or independent-sample inference. These short campaigns establish neither vendor rankings, cold/warm state, a provenance-off counterfactual nor long-term availability.

Repeated queries. Pairing the two corrected campaigns yields 297 verified pairs and three unmatched failures. All pairs retain the same source-ID multiset, yet 97 records in 25 pairs differ only in score fields after matching IDs. Stable identifiers therefore do not preserve all earlier metadata. This retrospective comparison measures neither relevance nor the cause of score variation (Supplementary Appendix B.4).

4.2 Verification, tampering and composition

Seventeen local mutations of each application’s first verified nonempty HTTPS capture test bytes/signatures, links,

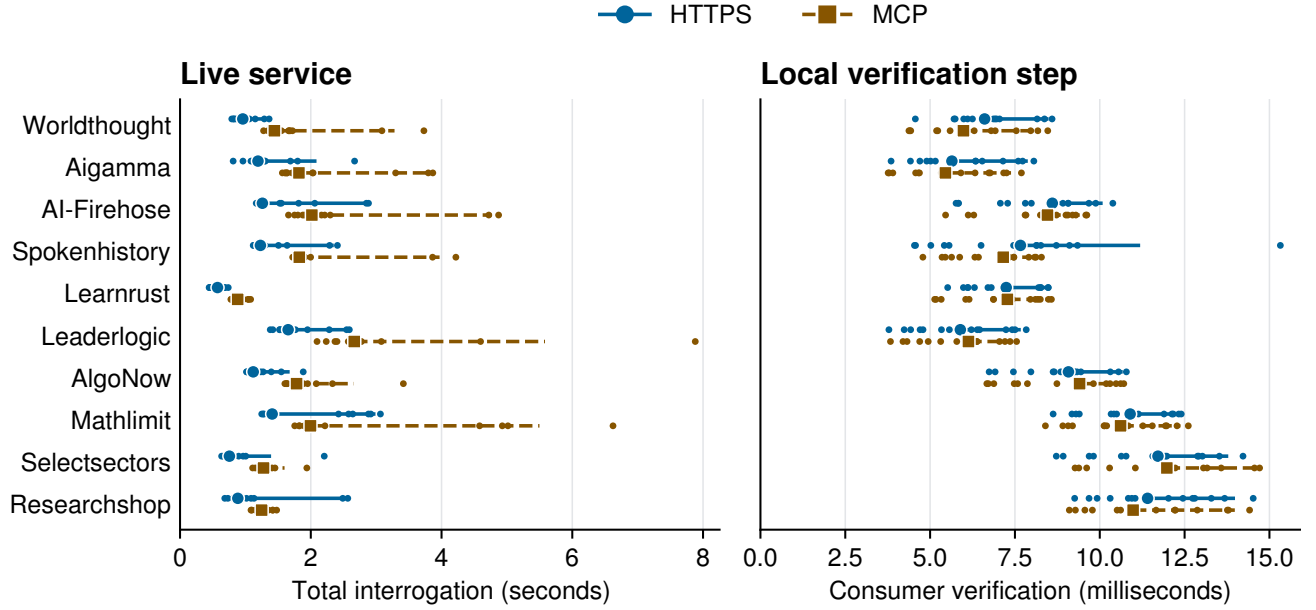


Figure 3: Prospective live replication, successful attempts only. Points are individual observations; larger circles (HTTPS) and squares (MCP) are medians and horizontal lines extend to the descriptive 95th percentile. Each transport has up to 15 observations per application. Total includes MCP initialization where applicable; verification is the consumer’s graph and authorization check after response buffering. The axes use different units.

Operation	Median	P95	Range
Canonical CID, all nodes	0.37	1.07	0.06–1.17
Ed25519, all CID messages	0.24	0.31	0.16–0.31
Capture and self-verification	9.38	15.94	4.55–17.20
Complete consumer verification	7.67	12.51	3.83–13.52

Table 2: Local JavaScript/WASM replay: milliseconds per whole capture. Columns summarize 50 per-input medians, not 50 independent deployments. Operations overlap and must not be added.

request/query/namespace binding, anchors/certificates, revocation, version poisoning/rollback, duplicate nodes, malformed CIDs and outer-result substitution. Recipes and version-state outcomes are retained. Synthetic controls use ephemeral keys.

All 170 mutations of live captures were rejected without advancing version state. Two validly signed synthetic revocations, one for a CID and one for a leaf key, were also rejected. A correctly signed false statement, $2 + 2 = 5$, was accepted as an authentic declaration, as the protocol requires. The consumer additionally verified a union of ten retained retrieval graphs containing 110 signed nodes under ten pinned anchors; deleting a required source node broke graph closure. This is a collection of producer-signed graphs with an unsigned local root manifest, not a newly generated answer or a signed claim of cross-domain semantic derivation.

Adversarial review exposed a list-selector defect: JavaScript compared literal anchor names/keys while Rust

normalized aliases. The current boundary rejects aliases and duplicate identities before verification. Ten synthetic exploit variants passed the historical consumer and failed the stricter boundary without state changes. Fifty retained captures produced identical outputs under one-anchor and ten-anchor policies, 100 comparisons. The live campaigns and timings precede this repair (Supplementary Appendix C). The lesson is to make cross-language identity selection agree before applying signature and freshness checks.

4.3 Controlled cost on live payloads

Replay selects the replication’s first verified HTTPS response for each query/application, fifty inputs. Each operation receives 20 warmups and 50 batches of five iterations, with operation order rotated: canonical IDs for all nodes; signing prepared CID messages with an ephemeral key; the capture wrapper including graph construction, signing and self-verification; and complete keyless consumer verification. Network and durable state I/O are excluded. Historical captures are checked at retained list issuance without changing production clocks.

All 50 inputs completed replay, with 250 measured iterations per operation and input. Each batch’s elapsed time is divided by five; each input contributes the median of its 50 batch means. Table 2 summarizes those 50 input medians with equal weight. The host was Intel(R) Core(TM) Ultra 9 285K, Node v22.23.2, Windows; raw batch samples are retained separately.

These are whole-capture JavaScript/WASM costs on one host, not production overhead against a disabled baseline.

Native/WASM agreed on 650 CIDs and authorizations from fifty captures in the first corrected campaign. A separate Python encoder reconstructed their CIDs and checked signatures within an integer profile; it does not implement authorization or request policy. Supplementary Appendices B and C.3 give these checks and separate single-node loopback timings, which exclude deployed retrieval and full capture assembly.

5 Related Work

NDN secures named objects (Zhang et al. 2014); its trust schemas supply the name-based authorization used here (Yu et al. 2015, 2023). Secure Web Objects (Yu et al. 2024) bind names to signers, while FLIC (Tschudin et al. 2025) organizes hash-linked collections. Codexproof binds typed derivation links to parent content. PROV (Moreau and Missier 2013) supplies semantic types without cryptographic binding; IPFS supplies Merkle DAGs (Benet 2014). in-toto checks supply-chain steps and artifact relationships (Torres-Arias et al. 2019), SLSA specifies build provenance (OpenSSF 2023), and Sigstore supplies signing infrastructure (Newman, Meyers, and Torres-Arias 2022). Transparency mechanisms (Laurie, Langley, and Kasper 2013; Birkholz et al. 2026) and revocation audits (Golaszewski et al. 2026) inform distinct trust concerns; logs do not prove complete disclosure.

MCP (2026), A2A (A2A Project 2026) and AIP (Prakash 2026) concern invocation, task exchange and delegation. PROV-AGENT captures workflow interactions using PROV and MCP (Souza et al. 2025); a recent survey maps broader agent-provenance problems (Wang et al. 2026). This evaluation instead measures checks on exchanged evidence bytes and authority. DIG retains investigative sources (Kejriwal 2026); SARA traces legal reasoning (Bonfim et al. 2026). Their applications motivate inspectable evidence without transferring their task-quality or institutional-impact claims.

6 Threats and Limitations

Adversaries and trust. Changed node bytes, forged signatures, re-parented links and unauthorized namespaces fail their respective checks. Missing parents break closure; withheld trust material fails closed. A stolen consumer/service bearer permits request abuse or suppression but provides no producer key. One compromised producer can forge its domain’s declarations; the shared operator vault exposes multiple domains. Version rollback protection depends on observed history and trustworthy consumer storage. Roots are accepted out of band; there is no transparency-log consistency check.

Meaning and execution. An authorized producer can omit contrary evidence or lie about retrieval. Verification checks declared links and closure, not all PROV constraints, redundant result projections or execution. Remote attestation (Birkholz et al. 2023) addresses a separate execution claim. Signatures establish neither factual correctness nor completeness.

Evidence scope. The proactive audit diagnosed trust expiry and an extraction defect; no user-reported bad-answer

incident has been investigated. Live capture, retention and controlled faults do not establish measured user benefit, root-cause accuracy, long-term adoption or time saved. Ten applications share one operator and full consumer implementation. The issuer is a protected local process; unattended renewal independent of that workstation is not established. Expired material fails closed on provenance requests. Retention is not perpetual availability or reconstruction of the whole historical index.

7 Conclusion

Codexproof made proactive inspection useful across all ten domains within a short deployment window. One operator gained a common way to interrogate retrieval evidence, check current trust and restore earlier captures across heterogeneous stores and runtimes.

I had spent weeks running overnight panels and scans in separate applications to identify improvements to their vector stores. But after venturing an AI-driven run with Codexproof on the entire fleet’s scope, this guided inspection surfaced actionable improvements across all ten domains within minutes, and the AI was able to deploy the corrections almost immediately. This is a qualitative operator account, separate from the timed measurements.

The development audit exposed dropped source IDs and hashes, omitted ranking text, an embedding API mismatch and unchecked ingestion writes. AlgoNow’s corrections used the contextual embedding API, restored full catalog text to reranking, reused unchanged embeddings and verified writes before activating retained catalog generations. AI Firehose’s capture adapter preserved older glossary summaries or titles when passage text was absent and recorded which field supplied the evidence. These were concrete changes to retrieval and evidence handling.

Across three vector-store technologies, the subsequent deployment produced 597 verified and restored captures in two 300-attempt campaigns. Trust renewal and a corpus update preserved earlier signed evidence. Proactive interrogation then exposed expired trust across all ten producers and localized a PDF extraction defect, without waiting for a user-reported bad-answer incident.

An AI coding assistant can use the shared capture contract as a concrete pattern for scaffolding adapters. Once connected, authenticated HTTPS/MCP interfaces let an assistant steer bounded retrieval and inspect retained evidence across domains; it can then help implement corrections under operator control. The operator can check and challenge a declaration; its truth and the cause of a suspect result require investigation.

Three practices follow: retain selected evidence separately, validate request contracts, and check trust activation alongside restoration. For this fleet, the short observation window captured an early practical return: AI-assisted inspection and corrective action across evolving applications, with earlier evidence still verifiable.

AI-assistance disclosure. An AI ensemble assisted with pressure testing, review, and validation. The author originated and verified the work and takes full responsibility.

References

- A2A Project. 2026. Agent2Agent (A2A) Protocol Specification v1.0. <https://a2a-protocol.org/latest/specification/>.
- Allione, E.; and O'Hara, D. 2026. Provenance-Native Retrieval: A Signed, Content-Addressed Thin Waist for Agent Evidence. Accepted paper, AgentNet workshop at IEEE ICNP.
- Benet, J. 2014. IPFS - Content Addressed, Versioned, P2P File System. arXiv:1407.3561.
- Bernstein, D. J.; Duif, N.; Lange, T.; Schwabe, P.; and Yang, B.-Y. 2012. High-Speed High-Security Signatures. *Journal of Cryptographic Engineering*, 2(2): 77–89.
- Birkholz, H.; Delignat-Lavaud, A.; Fournet, C.; Deshpande, Y.; and Lasker, S. 2026. An Architecture for Trustworthy and Transparent Digital Supply Chains (SCITT). IETF RFC 9943.
- Birkholz, H.; Thaler, D.; Richardson, M.; Smith, N.; and Pan, W. 2023. Remote Attestation Procedures (RATS) Architecture. IETF RFC 9334.
- Bonfim, F. C. J.; Silva, S. P.; Neves, A. S.; Santos, H.; Neto, J. A. M.; Araujo, J. L. B.; Pires, R. S.; da Silva Costa, R. A.; Furtado, V.; and Pinheiro, V. 2026. SARA: Leveraging LLM Agents and Jurisprudential Ontologies for Automated Legal Reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(47): 39911–39919.
- Golaszewski, E.; Krawetz, N.; Sherman, A. T.; Ziegler, E.; Matukumalli, S. K.; Yus, R.; et al. 2026. Verifying Provenance of Digital Media: Why the C2PA Specifications Fall Short. arXiv:2604.24890.
- Jacobson, V.; Smetters, D. K.; Thornton, J. D.; Plass, M. F.; Briggs, N. H.; and Braynard, R. L. 2009. Networking Named Content. In *Proc. ACM CoNEXT*, 1–12.
- Kejriwal, M. 2026. A Deployed Investigative AI Search Engine for Combating Human Trafficking at Web Scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(47): 40015–40024.
- Laurie, B.; Langley, A.; and Kasper, E. 2013. Certificate Transparency. IETF RFC 6962.
- Model Context Protocol. 2026. Model Context Protocol Specification, 2026-07-28. <https://modelcontextprotocol.io/specification/2026-07-28>.
- Moreau, L.; and Missier, P. 2013. PROV-DM: The PROV Data Model. W3C Recommendation.
- Newman, Z.; Meyers, J. S.; and Torres-Arias, S. 2022. Sigstore: Software Signing for Everybody. In *Proc. ACM Conference on Computer and Communications Security (CCS)*.
- O'Connor, J.; Aumasson, J.-P.; Neves, S.; and Wilcox-O'Hearn, Z. 2020. BLAKE3: One Function, Fast Everywhere. <https://blake3.io>.
- OpenSSF. 2023. SLSA: Supply-chain Levels for Software Artifacts, Specification v1.0. <https://slsa.dev/spec/v1.0/>.
- Prakash, S. 2026. AIP: Agent Identity Protocol for Verifiable Delegation Across MCP and A2A. arXiv:2603.24775.
- Souza, R.; Gueroudji, A.; DeWitt, S.; Rosendo, D.; Ghosal, T.; Ross, R. B.; Balaprakash, P.; and da Silva, R. F. 2025. PROV-AGENT: Unified Provenance for Tracking AI Agent Interactions in Agentic Workflows. In *Proc. 2025 IEEE International Conference on e-Science*, 467–473. ArXiv:2508.02866.
- Torres-Arias, S.; Afzali, H.; Kuppusamy, T. K.; Curtmola, R.; and Cappos, J. 2019. in-toto: Providing Farm-to-Table Guarantees for Bits and Bytes. In *Proc. USENIX Security Symposium*, 1393–1410.
- Tschudin, C.; Wood, C. A.; Mosko, M.; and Oran, D. R. 2025. File-Like ICN Collections (FLIC). IRTF Internet-Draft, draft-irtf-icnrg-flic-07.
- Wang, Y.; Zhang, J.; Cai, T.; et al. 2026. From Agent Traces to Trust: A Survey of Evidence Tracing and Execution Provenance in LLM Agents. arXiv:2606.04990.
- Yu, T.; Ma, X.; Patil, V.; Kocaoğullar, Y.; Zhang, Y.; Burke, J.; Kutscher, D.; and Zhang, L. 2024. Secure Web Objects: Building Blocks for Metaverse Interoperability and Decentralization. arXiv:2407.15221.
- Yu, T.; Ma, X.; Xie, H.; Kocaoğullar, Y.; and Zhang, L. 2023. A New API in Support of NDN Trust Schema. In *Proc. ACM Conference on Information-Centric Networking (ICN)*, 46–54. Doi: 10.1145/3623565.3623709.
- Yu, Y.; Afanasyev, A.; Clark, D.; kc claffy; Jacobson, V.; and Zhang, L. 2015. Schematizing Trust in Named Data Networking. In *Proc. ACM Conference on Information-Centric Networking (ICN)*, 177–186.
- Zhang, L.; Afanasyev, A.; Burke, J.; Jacobson, V.; kc claffy; Crowley, P.; Papadopoulos, C.; Wang, L.; and Zhang, B. 2014. Named Data Networking. *ACM SIGCOMM Computer Communication Review*, 44(3): 66–73.