

Provenance-Native Retrieval: A Signed, Content-Addressed Thin Waist for Agent Evidence

Eric Allione
aigamma.com
eric@aigamma.com

Dr. Dustin O'Hara
Western Washington University
oharad@wwu.edu

Abstract—Agent systems increasingly pass answers across stores, tools, and organizations, but the evidence behind those answers loses checkability at each trust boundary. We present Codexproof, a minimal provenance substrate that turns each evidence unit, tool result, and answer into a signed, content-addressed object. Its central invariant is simple: the 32-byte target of every typed provenance link is also the integrity commitment to its parent. The provenance graph is therefore also the integrity graph. One verifier checks exact canonical bytes, signer authorization, typed lineage, and fail-closed revocation under consumer-chosen per-domain anchors. Just-in-time capture seals evidence only when it crosses a trust boundary, leaving the retrieval hot path crypto-free. A shared Rust core produces identical content-ids in native code and WebAssembly. In a four-store, single-operator loopback testbed, the off path takes about 2 μ s per request; capture adds 88.6 μ s p50 end to end at one request in flight, including about 15 μ s of sealing per node. The producer-side marginal cost is \$0.065 per million sealed nodes; fallback re-embedding, LLM judging, and re-derivation cost two to five orders of magnitude more. Codexproof authenticates declared provenance, not truth: any post-signing change to a declared evidence edge changes a content-id or invalidates a signature.

I. INTRODUCTION

When agents hand one another answers at machine speed, the evidence behind those answers does not automatically remain checkable by the next consumer. A retrieval-augmented generation (RAG) agent cites chunks from stores it does not operate; a planning agent acts on a tool result from an API it cannot inspect; one organization's agent adopts another's answer as evidence. Each hop builds on evidence its consumer never controlled, so tampering or misattribution compounds across caches, systems, and organizations.

The Model Context Protocol (MCP) standardizes tool invocation [1], Agent2Agent (A2A) standardizes agent discovery and task exchange [2], and identity layers govern delegation [3]. Codexproof operates at a different layer: it binds exchanged artifacts to declared evidentiary lineage. A signature over answer text, by itself, binds neither the exact bytes of cited evidence nor the answer's declared reliance on that evidence. HTTPS authenticates only the current host, then says nothing after evidence is cached, reranked, or repackaged. Information-centric networking diagnosed this failure mode nearly two decades ago: secure the object, not the channel [4], [5].

The workshop asks whether large-scale agent ecosystems need an architectural thin waist, the hourglass pattern behind the Internet's scaling [6]: one minimal shared layer, with no global agreement on frameworks or trust authorities. This paper offers a candidate for the answer-to-evidence edge, an existence proof that such a layer can be small (four fields; measured structural overhead below 800 B per node) and cheap. Adoption, not one paper, would establish it as a standard. We demonstrate feasibility across namespaces, stacks, and backends under one operator; the architecture is designed for cross-organization interoperability, which remains untested (Section VI).

The key abstraction is a verifiable evidence object, not another agent protocol. The 32-byte target of each typed W3C PROV [7] link is the BLAKE3 [8] content-id of its parent; each object is signed with Ed25519 [9]. Hash the bytes, sign the name, type the links, and let links be names. Given the serialized graph and consumer-chosen anchors, one verifier checks exact bytes, signer authorization, declared lineage, and revocation without trusting stores or delivery channels.

The design stands on inherited parts; Section III draws the inherited-versus-new line explicitly, and Section V tags each neighbor. The contribution is one invariant, one operationalization, and one measured existence proof:

(1) *A provenance-native object.* Each typed provenance link targets its parent's content-id, so a self-contained DAG is simultaneously a lineage graph and an integrity commitment.

(2) *An operational trust boundary.* Consumer-chosen per-domain anchors, name-schema authorization in the Named Data Networking (NDN) tradition [10], mandatory fail-closed revocation, and just-in-time capture via a keyless gateway make that invariant enforceable across agent stacks without a global authority.

(3) *An implemented and measured existence proof.* One Rust core runs natively and in WebAssembly with CI-checked identical content-ids. In a four-store, single-operator loopback testbed, capture adds 88.6 μ s p50 end to end at one request in flight, including about 15 μ s of sealing per node; integrity and authenticity verification remains about 27 μ s per node through 512 nodes, and producer-side marginal cost is \$0.065 per million sealed nodes.

Codexproof verifies declared lineage, not whether the declaration is true, complete, or faithful to the producer's actual execution (Section IV-D).

II. A SHARED OBJECT FOR AGENT EVIDENCE

A. Three agent workflows, one evidence edge

These three workflows share one boundary: an artifact from one trust domain becomes evidence in another. In RAG, signed chunks and an answer’s `wasDerivedFrom` links commit to exact bytes; in tool invocation, a result is an Entity generated by an Activity and attributed to the tool’s Agent; in multi-agent workflows, answers become evidence and `wasInformedBy` and `wasDerivedFrom` links chain the relay. The same model covers all three and composes with MCP and A2A: they organize exchange, while Codexproof secures the exchanged artifacts (the gateway in Section III-C exposes MCP tools and an A2A Agent Card). The Agent Identity Protocol (AIP) [3] sits beside it for delegation.

B. Why the shared layer is an object

We argue that a shared evidence object can simplify cross-stack verification by standardizing what evidence *is* rather than how it moves. Figure 1 shows the consequence: without a shared object, every distinct producer stack can require a separate interface to every consumer. Backends, embedders, transports, and runtimes create more such stacks; with the object, each stack writes one emit-as-node shim and each consumer uses one verifier. The object standardizes the *mechanism*, not the *trust decisions*: anchors stay per-domain.

The shared format has two levels. The minimal one is the **content-id**: the BLAKE3 hash of the object’s canonical bytes. A signature alone would not suffice: it must bind to *some* name, and vendor-assigned names (a Pinecone ID, a pgvector row, a Qdrant point) drag interoperation back into pairwise reconciliation. Content addressing removes that reconciliation. The complete **packet** is the signed, PROV-typed node of Section III.

Invariant. A node’s content-id is BLAKE3 over its canonical encoding. Each typed link carries relation r and its parent p ’s content-id, $\text{cid}(p)$. Length prefixes and type tags make the encoding injective; with collision resistance, a root cid commits transitively to the canonical bytes of every reachable node and the label of every reachable edge. The 32-byte target of a typed provenance link is therefore also its integrity commitment; lineage and integrity cannot disagree because they are one DAG. Signatures and name-schema authorization

establish who may assert each named node; they sit above this structural invariant and cannot substitute for it: a link carrying a mutable name instead would bind to whatever that name later denotes, letting a producer restate its own past evidence with every signature still valid.

A shared object must cross runtimes. Codexproof uses one pure, I/O-free Rust core, compiled to native code for producers and WebAssembly for browsers and other JavaScript clients. Identical content-ids across the two builds show runtime portability, not independent-implementation interoperability: both share the core. No normative encoding profile or second implementation is provided.

III. DESIGN

To separate inherited mechanisms from our contributions: inherited are content addressing and hash-derived naming (ICN and IPFS [11], [12]), data-name-to-signer-name binding (NDN trust schemas [10], [13]), the W3C PROV vocabulary [7], and BLAKE3 and Ed25519 [8], [9]; new are the three contribution groups in Section I. Together they yield one offline verifier for integrity, authenticity, schema authorization, and revocation across typed lineage, a property no component provides alone (Section V). NDN contributes hierarchical signed names, the trust-schema walk, and the secure-the-object stance; content addressing, PROV typing, unification, just-in-time capture, and the economics are independent of it.

A. Node format

A node is $\{kind, name, attributes, links\}$. *Kind* is a W3C PROV type: an Entity (an evidence unit, an answer), an Activity (a retrieval, a generation), or an Agent (a retriever, a model). *Name* is hierarchical and NDN-style, e.g., `/alpha/evidence/e1`. *Attributes* are canonical JSON. *Links* are $\{relation, target\}$ pairs; the relation is one of five PROV relations: `used`, `wasGeneratedBy`, `wasDerivedFrom`, `wasAttributedTo`, or `wasInformedBy`, and the target is the parent’s content-id.

Native and WASM builds share one encoding function. It sorts keys and links, length-prefixes fields, and applies deterministic domain separation. The content-id is BLAKE3 over these bytes; the signature is Ed25519 over a domain-separated message containing it. The node carries a self-certifying address (the cid), a schema-matchable name (Section III-B), an authenticity proof, and typed lineage. The signed bytes include the hierarchical name, as in NDN [5], and links extend that binding to lineage. Signing cost is payload-independent because the signature covers the 32-byte cid (Section IV); signing that cid adds no new hash assumption beyond the DAG’s BLAKE3 collision resistance. The substrate fixes evidence identity, not granularity: an evidence unit may be a corpus, document, segment, or claim.

B. Authorization: a name-based trust schema

Integrity and authenticity establish *that* someone signed exactly these bytes; authorization establishes that the *right signer*

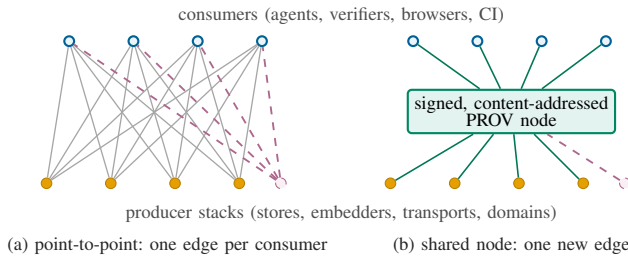


Fig. 1. Integration through a shared object. (a) Without the object, integrations multiply; the dashed newcomer adds one edge per consumer. (b) With it, each stack needs a shim and each consumer a verifier; a newcomer adds one edge.

did. Codexproof reuses the NDN trust-schema idea [10], [13]: trust rules relate data names to signer names, and the verifier walks the certificate chain to an anchor. Each domain publishes one named public key as a trust anchor, exposed as a `did:key` [14]; certificates bind subordinate keys to names. V1 enforces one fixed rule: *a node is authorized only if (a) its bytes match its content-id and its signature verifies, (b) its signer’s certificate chain reaches a trusted anchor with every issuer’s name a prefix of the name it certifies, and (c) both the node’s data name and the signer’s name lie under that anchor’s namespace.*

For example, `/alpha/evidence/e1` is signed by the `/alpha/retriever` key certified under the `/alpha` anchor. If a fresh revocation list revokes neither the node nor the signer, the verifier authorizes the node and reports the authorizing anchor. A `/beta`-rooted signature is rejected even if the verifier trusts both anchors; so is a chain containing an intermediate certificate for a name outside its issuer’s namespace, though every signature is valid. Candidate-chain walks are bounded by depth and total steps and fail closed on exhaustion; within those bounds, one planted decoy cannot mask a valid chain. Thus one domain cannot vouch for another’s data, yet a verifier may validate one graph spanning several trusted anchors. The fixed V1 rule isolates domains, not roles within one; declarative pattern rules with per-role nesting [10], [13] are the generalization.

C. Just-in-time capture across trust boundaries

Signing everything taxes the hot path, while centralizing keys collapses independently anchored domains into one gateway compromise. Codexproof does neither. It computes provenance *just in time*, on an *interrogation*: a request across a trust boundary. The gateway turns store records into unsigned node templates and dispatches each to its domain’s signing proxy. The proxy validates the template’s namespace, computes the content-id, signs with a key held only by the proxy process (never by the gateway and never on the wire), and returns the sealed node. The gateway returns the signed subgraph and public trust material; the consumer verifies client-side.

The security consequences are structural: the gateway holds no signing keys (its configuration surface has no key type). The proxy, in turn, checks the *template*’s namespace, not the caller’s identity: it signs any in-namespace template that reaches it, so its reachability perimeter is a deployment obligation, not a protocol property. A compromised gateway can deny service or induce reachable proxies to sign false in-namespace declarations, within the declared-lineage concession (Section IV-D); it cannot extract keys, sign without proxies, or alter sealed history. Compromising one proxy makes only that proxy’s domain forgeable; the others remain unforgeable. Dispatch is concurrent with strict timeouts, so a stalled domain yields a structured error naming it. The gateway also exposes MCP tools and publishes an A2A Agent Card, making the artifact service available through existing agent interfaces.

D. Revocation and selective disclosure

Content-addressed objects are immutable, so retraction and freshness need a mutable layer above them, and the C2PA experience shows what goes wrong when that layer is optional: conforming validators simply skip it [15]. Codexproof makes revocation checking mandatory and *fail-closed*. A node is authorized only against a fresh revocation list signed by the resolved anchor over the whole sorted ID set, with a cryptographically bound expiry: no fresh, valid list for that anchor, no authorization; dropping an entry breaks the list signature, so a truncated list is invalid rather than silently incomplete. All full-authorization paths (command-line `authorize`, store HTTP/MCP, browser verifier) run this gate; lower-level integrity APIs do not perform authorization. We claim the narrow true thing: mandatory in-path checking with an anchor-chosen staleness bound, not non-equivocation; within the freshness window a stateless verifier cannot detect rollback to an older still-valid list, though each list carries a signed version a stateful verifier can pin, and transparency-log consistency proofs [16], [17] are the named ladder beyond it.

For selective disclosure, a producer mints a sensitive attribute as a salted BLAKE3 commitment inside the signed bytes, with the opening out of band: withholding an opening later never changes the content-id, and the 16-byte salt defeats brute force on low-entropy fields.

E. The verification path

Verification answers four questions: Do these bytes match the claimed content-id? Which key signed them? Is that key authorized for this name under consumer-chosen anchors and fresh revocation state? Does every typed lineage edge resolve to the exact parent bytes? Over serialized bytes and those anchors, the verifier recomputes each cid, checks each signature, walks each certificate chain, and checks each revocation list. No store, channel, or intermediary is trusted; this carries NDN’s defining property [5] to agent evidence. The same verifier core compiles to WASM and backs a command-line interface (CLI) whose exit code is the CI oracle. The verifier is an untrusted-input parser, tested like one. An adversarial suite drives eleven attacks, including flipped signatures, false content-ids, certificate cycles and decoys, over-deep chains, malformed encodings, and mixed honest-and-corrupt bundles; each must be rejected without panicking or hanging.

IV. EVALUATION

Our testbed mirrors three production RAG services we operate. The first, `/alpha`, uses Pinecone with Voyage embeddings and reranking in JavaScript/React; `/beta` uses Supabase pgvector and JavaScript; `/gamma` uses Qdrant and Rust. A deliberately different synthetic feed, `/delta`, contains whole, compact, label-scored items. Each domain has its own anchor, but one operator runs all four (Section VI). The harness drives the real gateway and per-domain proxies over loopback TCP. Payloads preserve production schemas and lengths, which determine serialization, hashing, and envelope bytes; Ed25519 signs the fixed-length cid.

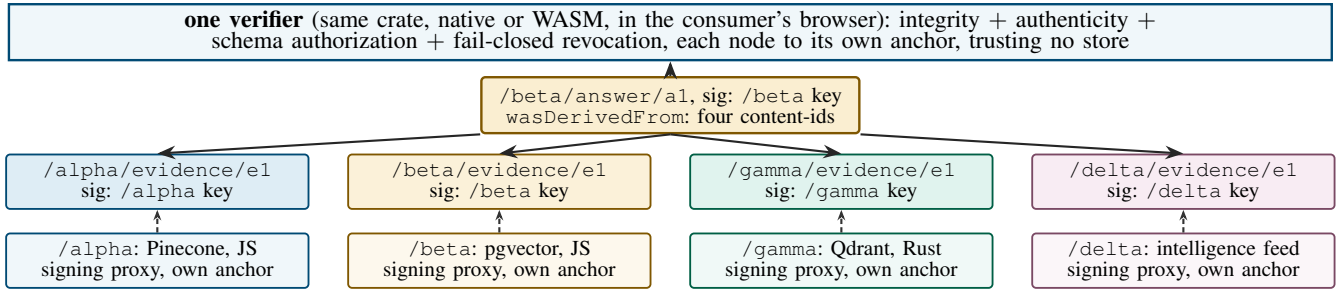


Fig. 2. The demonstration object: one answer whose `wasDerivedFrom` links are the content-ids of four evidence units, each signed under a different domain’s anchor (dashed: just-in-time capture); authorization maps each node to its own anchor, and revocation checking fails closed without trusting a store.

One measurement campaign on an Intel i9-14900K under Windows/MSVC used release-optimized cryptography (a toolchain defect capped transport optimization), 30k to 50k iterations per microbenchmark operation, and 2000 measured requests per mode (off, jit) at each concurrency level after 1000 warm-ups; two independent harnesses cross-validated results. Artifact commands rerun every benchmark. The in-browser verifier and demonstration bundle are live,¹ and CI on an independent Linux runner tests the reported functional properties.

A. Cross-stack verification under one operator

The demonstration object (Fig. 2) is one answer derived from four evidence units under different anchors. The CLI confirms that each node is authorized under its own anchor; the browser re-verifies the bundle via WASM. Adding `/delta` required a schema case, not gateway or signer changes: one integration rather than one per consumer. This demonstrates the mechanism within one operator, not cross-organization deployment. CI confirms identical native/WASM content-ids, verifies and authorizes native fixtures in WASM, and observes zero drift.

Tampering is localized. Flipping one evidence byte invalidates exactly that node within its domain; the answer’s derivation then fails because its link no longer resolves. Re-parenting while preserving signatures is rejected because the signed answer commits to both the relation and parent cid: a swap breaks a hash or leaves a dangling link. A dedicated CLI command and the adversarial suite exercise this property.

B. Overhead: microseconds

Table I gives operation costs, Table II the gateway A/B, and Fig. 3 behavior under load. Three findings carry the argument.

First, with provenance off, the gateway’s p50 is 1.3 to 2.3 μ s across stores and concurrency levels (pooled: 1.5 to 2.1 μ s), with no cryptography, template work, or dispatch. The off baseline is protected by absence: the off path imports no tracing runtime, and a test asserts that off-mode telemetry carries no per-store field.

Second, sealing costs about 15 μ s per node, dominated by about 11 μ s of Ed25519 signing. Because the signature covers the 32-byte cid rather than the payload, cost varies

TABLE I
NATIVE PER-OPERATION COST (SINGLE THREAD; MEANS OVER 30K TO 50K ITERATIONS; P99 FROM SEPARATE RUNS; ROWS DO NOT SUM).
AUTHORIZE INCLUDES NODE VERIFICATION AND THE CHAIN WALK, NOT REVOCATION.

operation	mean	p99
BLAKE3 hash (1 KB)	1.5 μ s	1.7 μ s
Ed25519 sign (over the cid)	10.8 μ s	16.9 μ s
Ed25519 verify (over the cid)	26.7 μ s	48.1 μ s
node verify, cid + signature (1 KB)	27.5 μ s	45.5 μ s
authorize (node + chain to anchor)	54.1 μ s	114.0 μ s

TABLE II
PER-STORE GATEWAY A/B (LOOPBACK, ONE REQUEST IN FLIGHT; 2000 REQUESTS/MODE): “OFF” = CRYPTO-FREE BYPASS; “OVH” = JIT MINUS OFF; “SEAL” = SERIALIZE + BLAKE3 + Ed25519 PER NODE; PAYLOAD = MEAN EVIDENCE BYTES.

store	off p50	jit p50	ovh p50	seal/node	payload
/alpha (Pinecone)	1.3 μ s	86.5 μ s	85.2 μ s	14.6 μ s	1472 B
/beta (pgvector)	1.5 μ s	89.7 μ s	88.2 μ s	15.3 μ s	1319 B
/gamma (Qdrant)	1.5 μ s	92.3 μ s	90.8 μ s	15.2 μ s	1844 B
/delta (feed)	1.5 μ s	90.6 μ s	89.1 μ s	14.2 μ s	505 B

little as payloads range from 505 to 1844 B; gateway and microbenchmark results agree.

At one request in flight, capture adds 88.6 μ s p50 end to end; the roughly 74 μ s beyond sealing is predominantly loopback proxy wait, not measured sealing CPU time. At 100-way concurrency, sealing reaches only 19 μ s; Fig. 3 reflects queueing against four fixed proxies. Adding proxy replicas should flatten the latency curve, but that expectation is unmeasured.

The pooled jit p99-to-p50 ratio ranges from 1.7 to 2.9 across the measured concurrency levels; the off path’s p99 remains under 17 μ s even at 100-way concurrency.

Third, slow interrogations remain attributable to their stores: `/gamma`’s rich schema is heaviest to hash, whereas `/delta`’s compact feed is lightest. Consumer integrity and authenticity checks cost about 27 μ s per node. Full authorization also checks the chain and revocation list; Table I times node-plus-chain only. Checking the list adds one list-signature verification and membership tests per authorization. Telemetry uses low-cardinality axes (store, backend, mode), never object IDs.

¹codexproof.com

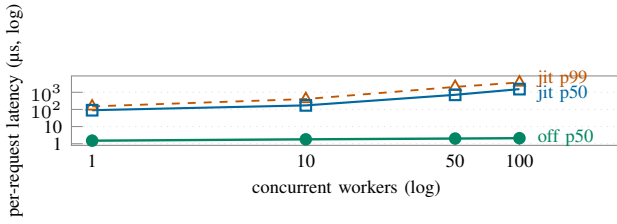


Fig. 3. Pooled per-request latency across concurrency levels (log-log). The off path stays flat at 1.5 to 2.1 μ s across two decades of load; jit grows with queuing against four fixed proxies.

C. Overhead: unit economics

Microseconds answer the engineer; a buyer asks what the layer and its fallback operations cost. Each measured interrogation sealed one node, so a six-interrogation turn costs about \$0.39 per million turns (Table III). The producer-side marginal-cost model combines 14.8 μ s of measured single-node sealing CPU time with an envelope of 638 to 781 bytes beyond the payload (718 pooled), totaling about \$0.065 per million sealed nodes. This model excludes consumer authorization and deployment-dependent dispatch wait.

When trustworthy lineage is unavailable, an operator may re-fetch, re-embed, ask another model to judge, or re-derive. At pooled prices, these fallback actions cost 1.78 $\times$, 297 $\times$, 11,903 $\times$, and 148,737 $\times$, respectively. They are not security-equivalent substitutes: re-fetch proves possession with the origin online, and none binds the answer to the exact bytes it used or yields third-party tamper evidence. Re-fetch is cheaper only for /delta (0.65 $\times$); an LLM judgment is an opinion, not a proof.

Egress accounts for about 99.75% of marginal cost, roughly 400 $\times$ the cryptographic cost. The lever is envelope compactness: never re-ship a cid-addressable payload; do not optimize already-cheap cryptography.

The list-price assumptions, overridable in the artifact, are CPU \$0.04/vCPU-hour, egress \$0.09/GB, voyage-3 \$0.06 per million tokens, Haiku 4.5 \$1/\$5 and Opus 4.8 \$5/\$25 per million input/output tokens, and about 4 bytes/token. Re-fetch downloads the payload from its origin and incurs egress only.

D. Threat model

Table IV bounds the claim: signatures do not prove producer honesty. We authenticate declared lineage; attestation and proof-of-inference can be layered above it (Section VI), while salted commitments make a producer’s declared manifest of considered evidence tamper-evident.

V. RELATED WORK

Data-centric security and named-data trust. NDN [5] secures objects, not channels; its trust schemas [10], [13] bind data names to signer names, providing the authorization model we reuse. Secure Web Objects [18] chain names to signers; our links also commit to derivation. Authorization requires revocation checks. FLIC [11] hash-links signed objects but does not unify typed lineage with integrity.

TABLE III
PRODUCER-SIDE COST PER MILLION SEALED NODES AND MULTIPLES FOR FALLBACK ACTIONS (LIST PRICES; ONE NODE SEALED PER INTERROGATION).

	\$/M	re-fetch	re-embed	LLM judge	re-derive
/alpha	0.058	2.30 $\times$	383 $\times$	14,203 $\times$	191,657 $\times$
/beta	0.061	1.96 $\times$	326 $\times$	12,854 $\times$	163,087 $\times$
/gamma	0.070	2.36 $\times$	393 $\times$	12,930 $\times$	196,302 $\times$
/delta	0.070	0.65 $\times$	107 $\times$	8,178 $\times$	53,722 $\times$
all stores	0.065	1.78$\times$	297$\times$	11,903$\times$	148,737$\times$

TABLE IV
ADVERSARY CAPABILITIES AND OUTCOMES (CONDENSED).

adversary capability	outcome
tamper with node bytes in transit	rejected (cid mismatch)
forge a signature without the signing key	rejected (Ed25519)
re-parent an answer	rejected (links are cids)
sign a node in another domain’s namespace	rejected (trust schema)
sign with an uncertified key	rejected (no chain)
compromise the gateway	no keys held; DoS or mis-declaration
compromise one signing proxy	only that domain forgeable
compromise an anchor key	domain forgeable until re-anchored (conceded)
stall one domain’s proxy	named timeout; others resolve
withhold list or alter signed set	fail-closed: no authorization
roll back to an older valid list	masks revocations until expiry (conceded)
omit contradicting evidence	out of scope (conceded)
lie about own inputs (valid key)	out of scope (conceded)

Content and supply-chain provenance. W3C PROV [7] supplies node and relation types but no cryptographic binding. in-toto [19], SLISA [20], and Sigstore [21] sign digest-correlated attestations; each Codexproof provenance link instead addresses its target, making the graph self-certifying. A C2PA audit [15] motivates fail-closed revocation; transparency logs [16], [17], including Sigstore’s Rekor, are a completeness layer we do not provide. IPFS [12] generalized content-addressed Merkle DAGs; CID links alone do not authenticate authors. Codexproof signs and authorizes linked nodes, making declared derivation attributable.

Agent interoperability and identity. MCP [1] standardizes tool invocation and A2A [2] agent discovery and task exchange; Codexproof binds the artifacts exchanged through either. AIP [3] chains authority, delegation, and optional completion blocks along invocations; Codexproof makes typed evidence links target parent content-ids; the layers can compose. Verifiable credentials [22] and did:key [14] encode anchors.

LLM-agent provenance. A recent survey [23] identifies trace schemas, semantic provenance, multi-agent provenance, safety, recovery, privacy, and evaluation as open problems; Codexproof focuses on cross-domain integrity and attribution. PROV-AGENT [24] extends PROV to agentic workflows but trusts its infrastructure: no signatures or cross-domain verification.

VI. LIMITATIONS, DEPLOYMENT, AND FUTURE WORK

Six limitations bound the claims. *Shared operator*: one organization runs all four domains; the demonstration crosses anchors, stacks, and languages, not organizations. *Synthetic payloads*: schema- and length-faithful templates preserve cryptographic timings and byte counts; the measured path contains no live vector query. *Single campaign*: reruns reproduce shapes and economics, not last digits. *Loopback floor*: deployment adds network round-trip time to capture, but not to provenance-off retrieval or offline verification. *Fixed schema*: one nesting rule, flat within a domain. *Anchor lifecycle*: `did:key` anchors cannot rotate.

Operating across organizations. Each organization runs a proxy and publishes an anchor; whoever exposes evidence runs a gateway. A compromised gateway leaks no key but may induce reachable proxies to sign false in-namespace declarations. Anchor discovery is out of scope: consumers obtain anchors out of band and choose which to trust. Anchors sign revocation-list expiries; consumers may demand shorter windows, trading availability for freshness. The proxy interface accommodates hardware security modules. Anchor rotation is not implemented; supporting it would require placing current `did:key` anchors below a rotatable anchor. Transparency-log consistency proofs [16], [17] are the rung above freshness.

Scaling, storage, and bandwidth. Verification was linear across the measured 4-to-512-node range (27.0 to 27.7 μ s per node; 13.8 ms for 512). Self-contained subgraphs verify independently, so consumers check only the slices they use. Growth lies in envelopes (638 to 781 B beyond payload), anchors, and revocation lists; lists grow with revoked artifacts and signer keys, not the corpus.

Composing with correctness attestation. Remote attestation [25] can support environment-state claims, while proof-of-inference addresses whether a computation ran as claimed; provenance binds declared inputs and outputs across domains. Binding an Activity to attestation evidence narrows the declared-lineage concession (Section IV-D); none proves correctness.

VII. CONCLUSION

Agents need not share a framework or global authority to identify the same evidence bytes and verify the same declared lineage; each consumer chooses which anchors to trust. That is IP's bargain [6], carried to agent evidence: agree on almost nothing, interoperate anyway. Codexproof makes lineage and integrity one self-certifying graph: hash the bytes, sign the name, type the links, and let links be names. In our four-store, single-operator loopback testbed, capture adds 88.6 μ s p50 end to end, including about 15 μ s of sealing per node; producer-side marginal cost is 6.5 cents per million sealed nodes. The system does not prove truth, but once a declaration is signed, no intermediary can change it without changing a content-id or invalidating a signature. This paper establishes feasibility. Whether the object becomes an architectural waist depends on cross-organization deployment.

AI-assistance disclosure. An AI ensemble assisted with implementation, testing, editing, and review. The authors originated and verified the work and take full responsibility.

REFERENCES

- [1] Model Context Protocol, "Model context protocol specification, 2026-07-28," <https://modelcontextprotocol.io/specification/2026-07-28>, 2026.
- [2] A2A Project, "Agent2Agent (A2A) protocol specification v1.0," <https://a2a-protocol.org/latest/specification/>, 2026.
- [3] S. Prakash, "AIP: Agent identity protocol for verifiable delegation across MCP and A2A," arXiv:2603.24775, 2026.
- [4] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proc. ACM CoNEXT*, 2009.
- [5] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, K. Claffy, P. Crowley, C. Papadopoulos, L. Wang, and B. Zhang, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, 2014.
- [6] M. Beck, "On the hourglass model," *Communications of the ACM*, vol. 62, no. 7, 2019.
- [7] L. Moreau and P. Missier, "PROV-DM: The PROV data model," W3C Recommendation, 2013.
- [8] J. O'Connor, J.-P. Aumasson, S. Neves, and Z. Wilcox-O'Hearn, "BLAKE3: One function, fast everywhere," <https://blake3.io>, 2020.
- [9] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang, "High-speed high-security signatures," *Journal of Cryptographic Engineering*, vol. 2, no. 2, 2012.
- [10] Y. Yu, A. Afanasyev, D. Clark, K. Claffy, V. Jacobson, and L. Zhang, "Schematizing trust in named data networking," in *Proc. ACM Conference on Information-Centric Networking (ICN)*, 2015.
- [11] C. Tschudin, C. A. Wood, M. Mosko, and D. R. Oran, "File-like ICN collections (FLIC)," IRTF Internet-Draft, draft-irtf-icnrg-flic-07, 2025.
- [12] J. Benet, "IPFS - content addressed, versioned, P2P file system," arXiv:1407.3561, 2014.
- [13] T. Yu, X. Ma, H. Xie, Y. Kocaoğullar, and L. Zhang, "A new API in support of NDN trust schema," in *Proc. ACM Conference on Information-Centric Networking (ICN)*, 2023, doi: 10.1145/3623565.3623709.
- [14] D. Longley, D. Zagidulin, and M. Sporny, "The did:key method v0.9," W3C CCG Draft Community Group Report, 2022.
- [15] E. Golaszewski, N. Krawetz, A. T. Sherman, E. Ziegler, S. K. Matukumalli, R. Yus *et al.*, "Verifying provenance of digital media: Why the C2PA specifications fall short," arXiv:2604.24890, 2026.
- [16] B. Laurie, A. Langley, and E. Kasper, "Certificate transparency," IETF RFC 6962, 2013.
- [17] H. Birkholz, A. Delignat-Lavaud, C. Fournet, Y. Deshpande, and S. Lasker, "An architecture for trustworthy and transparent digital supply chains (SCITT)," IETF RFC 9943, 2026.
- [18] T. Yu, X. Ma, V. Patil, Y. Kocaoğullar, Y. Zhang, J. Burke, D. Kutscher, and L. Zhang, "Secure web objects: Building blocks for metaverse interoperability and decentralization," arXiv:2407.15221, 2024.
- [19] S. Torres-Arias, H. Afzali, T. K. Kuppusamy, R. Curtmola, and J. Cappos, "in-toto: Providing farm-to-table guarantees for bits and bytes," in *Proc. USENIX Security Symposium*, 2019.
- [20] OpenSSF, "SLSA: Supply-chain levels for software artifacts, specification v1.0," <https://slsa.dev>, 2023.
- [21] Z. Newman, J. S. Meyers, and S. Torres-Arias, "Sigstore: Software signing for everybody," in *Proc. ACM Conference on Computer and Communications Security (CCS)*, 2022.
- [22] M. Sporny, T. Thibodeau, Jr., I. Herman, G. Cohen, and M. B. Jones, "Verifiable credentials data model v2.0," W3C Recommendation, 2025.
- [23] Y. Wang, J. Zhang, T. Cai *et al.*, "From agent traces to trust: A survey of evidence tracing and execution provenance in LLM agents," arXiv:2606.04990, 2026.
- [24] R. Souza, A. Gueroudji, S. DeWitt, D. Rosendo, T. Ghosal, R. B. Ross, P. Balaprakash, and R. F. da Silva, "PROV-AGENT: Unified provenance for tracking AI agent interactions in agentic workflows," in *Proc. 2025 IEEE International Conference on e-Science*, 2025, pp. 467–473, arXiv:2508.02866.
- [25] H. Birkholz, D. Thaler, M. Richardson, N. Smith, and W. Pan, "Remote attestation procedures (RATS) architecture," IETF RFC 9334, 2023.