AUTHENTICATED LATE-BOUND EVALUATION OF PERSISTENT AGENT ARTIFACTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Persistent Agent artifacts can parse after an SDK upgrade yet change a later consumer's result. We study *authenticated late-bound evaluation*: an origin authenticates a complete historical view; a receiver binds a later consumer, context and interpreter, then compares historical and current results. Omitting consumer-observable distinctions precludes exact evaluation. We implement signed snapshots, proof-backed consumer execution (PBCE), and authenticated keyed events. Closed-IR simulation and authenticated openings yield conditional, IR-relative soundness; materialization and current execution remain trusted. Unsupported semantics abstain. Source checkers cover five AutoGen schemas and a corrected LangGraph reducer under explicit projection/conversion premises. An unfiltered AutoGen inventory certifies only 5/22 workloads. An empirical AppWorld confirmation covers 18/36 tasks with a frozen replay profile; retrospective repairs reach 36/36, but the Agent solves only 15/36. These empirical extensions remain outside the closed-IR theorem and do not establish independent transfer. Full snapshots favor small dense objects, and keyed events can be cheaper than PBCE. On 35 restored development checkpoints, all 2100 local receiver queries agree. Including selective sender work but granting prepacked full archives, task-median first-handoff time is $0.157\times$ a compact full snapshot and $0.114\times$ a root-authenticated lazy full ZIP; warm-query time is nearly equal. These are local handoff costs, not whole-Agent speedup or superiority over all lazy stores. Our contribution is a scoped evaluation lifecycle and measured coverage–cost frontier, not arbitrary-code certification, generic-store superiority or ecosystem-wide safety.

1 INTRODUCTION

Agent systems persist heterogeneous tool requests, credentials, workflow configurations, and team state across upgrades, then resume actions from that state rather than merely parse it. A later SDK or not-yet-written auditor may consume an artifact after its source runtime is removed. We target an enterprise boundary where the origin and internal archive may see the full checkpoint, but a cross-tenant auditor is authorized only for one named consumer; full-snapshot decryption would expose unrelated conversation, credentials, or tool outputs. Parsing is insufficient: an omitted `max_rounds`, for example, can mean unbounded under its source runtime but a finite default after upgrade, while the same checkpoint still parses. Defaults, aliases, coercions, unions, and migration create analogous changes. AutoGen exposes `save_state`/`load_state` and warns that a team-state key and field changed across releases (Microsoft, 2026). We therefore allow a later hash-authorized consumer but forbid retrospective source-SDK execution.

The compatibility question is consumer-relative. For a historical source object x_s, current object x_r, admitted consumer P, and explicit input z, the relation is

$$\mathrm{Result}(\mathrm{Eval}(P, x_s, z)) \stackrel{?}{=} \mathrm{Result}(\mathrm{Eval}(P, x_r, z)). \tag{1}$$

Complete-object equality can reject harmless changes; an inferred field set can miss a branch that changes Equation 1. A perfect signature over an incomplete projection only authenticates the wrong relation.

A retrospective oracle can run both SDKs, but not after a source runtime is unavailable or deliberately removed. The decisive baseline is instead to sign the complete source object, reconstruct it later, and

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

run the same admitted consumer. This FULL-SNAPSHOT removes projection error; any selective protocol must justify itself against it, not receiver-only execution or heuristics.

We study *authenticated late-bound consumer evaluation*, organized around provenance-authenticated consumer semantics (PACS). Complete authenticated reconstruction is the baseline when disclosure permits: either FULL-SNAPSHOT or lossless updates to an already authenticated shared base. Dense access alone does not justify retransmitting the full object. Under receiver-scoped selective evidence transfer, **Proof- Backed Consumer Execution** (PACS-PBCE) commits a canonical source view, later opens only facts adaptively read by authorized code, and compares that code's source and receiver results. Unsupported operations, failures, and exhausted budgets abstain. PBCE changes transferred evidence, not accuracy relative to a faithful snapshot. A separate envelope prevents reuse under another receiver, purpose, invocation, or expiry. We measure Merkle facts and authenticated keyed events as co-equal selective implementations; the contribution is the bound source-view evaluation lifecycle and backend frontier, not one Merkle layout. Conventional authenticated queries bind post-issuance (P, z, K) and authorization to both views; we claim no expressiveness separation from generic ADS. Closed-IR replay does not re-execute the source SDK. The empirical native- assessment adapter trusts host helpers; only receiver disclosure is limited.

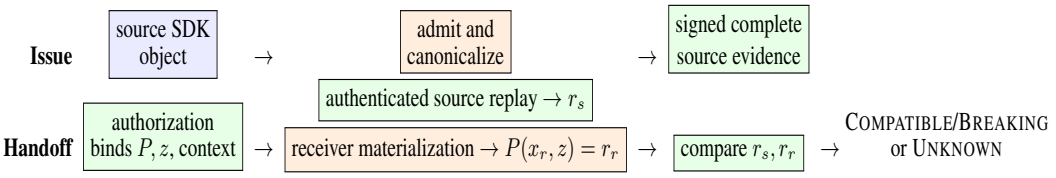

Figure 1: Assurance boundary: green is protocol-verified; amber materialization and current execution are trusted. The certificate is receiver-local. Source-to-P lifting is checked for two fragments and otherwise finite.

Table 4 distinguishes authenticated-data integrity, closed-IR replay, trusted materialization and checked or tested source bridges. Certificates concern receiver-local result equality for bound (P, z), not arbitrary JavaScript, remote execution or unobserved future actions. Evaluation separates fidelity from cost: complete evidence removes projection omission, while workload and representation determine transport savings. Fixed carriers do not estimate ecosystem prevalence.

Our contributions are:

1. a consumer-relative compatibility formulation for executable persistent Agent state whose source runtime has been removed, plus a bounded-profile characterization of sufficient fixed issue-time representations;
2. an authenticated evaluation lifecycle with explicit missing, rejection and resource semantics: closed-IR simulation, conditional result soundness, and authorization of actual evidence openings. Checked AutoGen and corrected LangGraph v2 translation compose with complete-input reconstruction under their stated projection premises; other bridges remain finite;
3. a measured coverage and cost boundary: complete authenticated reconstruction is the default, proof-backed facts and keyed events serve particular sparse regimes, and a complete consumer inventory exposes expression and bridge exclusions. Native continuations and service/fault tests support the stated scope.

2 PROBLEM FORMULATION

Let artifact a have authenticated provenance resolving to source runtime E_s; E_r is the receiver runtime. Source execution yields native object x_s and receiver execution yields x_r. These objects are conceptually distinct from the canonical archive.

Admitted object domain. The admitted domain $\mathcal{J}$ contains finite, acyclic trees of null, booleans, finite IEEE-754 binary64 values, UTF-16 strings, dense arrays, and plain string-keyed records, plus one distinguished finite `StrMap(string,string)` type. It excludes `undefined`, nonfinite

numbers, symbols, accessors, custom prototypes or descriptors, sparse arrays, cycles, shared identity, methods, and user-defined coercion. Issuance rejects a native value outside $\mathcal{J}$.

The encoder preserves exact binary64 bits (including the sign of zero), does not Unicode-normalize strings, and represents each record by its ordered `Object.keys` entry sequence. Encoding followed by flattening produces

$$A_E(a) = \mathsf{flat}(\mathsf{enc}(x_E)) : (\text{JSON pointer}, \text{feature}) \mapsto \text{atom}, \tag{2}$$

where features are `exists`, `type`, `value`, and `shape`. Let $\mathsf{mat}(A)$ be eager reconstruction and $\mathsf{lazy}(A)$ a proof-backed view.

Admitted observations. The certifying program is a closed declarative IR whose opcodes expand into the pure expressions and observations formalized in Appendix C.1: record/array reads, membership and authenticated own-key enumeration; dense-array length and traversal; construction and mutation of fresh local values; and `StrMap.get/size`. Reference identity, general prototype/descriptor reflection, getters, mutation of the input, ambient I/O, dynamic code, nondeterminism, and engine-order- dependent observations are unsupported. Unknown opcodes or parameters are rejected before execution. Enumeration returns the native ECMAScript own-key order committed in the parent `shape`; the Merkle tree may sort fact references independently. A plain-object prototype check and a cycle guard are proof-erased only after value admission establishes plain, acyclic trees. This is an object calculus, not all JavaScript.

For admitted consumer P and auxiliary input z, the domain-result grammar is

$$r ::= \mathsf{Ok}(\mathsf{enc}(v)) \mid \mathsf{DomainReject}(c, \mathsf{enc}(v)).$$

A successful semantic evaluation is

$$\mathrm{Eval}(P, x, z) \Downarrow (r, q), \tag{3}$$

where q is the ordered *query trace* of canonical path, feature, and returned semantic atom. It contains no Merkle siblings, nonces, or signatures. We write $\mathrm{Result}(\mathrm{Eval}(P, x, z)) = r$ for this first component. An interpreter/admission failure, unsupported operation or result, unhandled exception, timeout, or resource exhaustion instead yields $\mathsf{Abort}(d)$ and no comparable r. A protocol *proof trace* ϕ separately contains authenticated openings used to justify a source query trace. Thus q and ϕ are different typed objects.

The ordered handoff is compatible iff both admitted source and receiver evaluations produce domain results and those results are equal; query traces need not be equal across distinct objects:

$$\begin{aligned}\mathrm{Compat}(P, z; x_s, x_r) \iff \exists r_s, r_r, q_s, q_r :\ & \mathrm{Eval}(P, x_s, z) = (r_s, q_s) \wedge \\ & \mathrm{Eval}(P, x_r, z) = (r_r, q_r) \ \wedge\ r_s = r_r.\end{aligned} \tag{4}$$

Any Abort maps to protocol UNKNOWN and is never compared by exception class. A *domain rejection* is comparable only when it is an explicit opcode-level result in r; validator failure, missing proof, cache miss, and runtime exception are aborts, not domain rejections. Thus two unrelated `TypeError`s cannot certify compatibility. This is not observational equivalence for every program; changing P, z, or the explicit domain-rejection policy changes the relation.

After reconstructing the authenticated source result, the verifier returns CERT-COMPATIBLE iff both sides produce equal domain results, CERT-BREAKING iff both produce unequal domain results, and UNKNOWN if either side aborts or for missing evidence, failed admission, unauthorized context, invalid proof, timeout, or resource exhaustion. A *false-safe* is CERT-COMPATIBLE when native tagged outputs differ; an *over-quarantine* is CERT-BREAKING when they agree. Coverage is the fraction receiving either certificate.

Bounded-profile characterization. Although $\mathcal{J}$ is infinite, let profile B bound canonical bytes, nodes, depth, and resources. Its admitted set $\mathcal{J}_B$ is finite because the encoded alphabet and byte length are finite; let $\mathcal{C}_B$ be a finite set of authorized (P, z) pairs. Write $x \equiv_{\mathcal{C}_B} x'$ when every pair returns equal tagged outputs, and let $\Pi : \mathcal{J}_B \to M$.

Theorem 1 (bounded-profile issue-time characterization). For $\mathcal{J}_B$ and $\mathcal{C}_B$ as above, whose evaluations return domain results, an evaluator using only $(\Pi(x), P, z)$ exactly reproduces every source result iff

$$\Pi(x) = \Pi(x') \quad \Longrightarrow \quad x \equiv_{\mathcal{C}_B} x'. \tag{5}$$

Necessity follows because a separating consumer receives identical evidence. For sufficiency, choose one representative of each nonempty projection fiber; Equation 5 makes its result well-defined for the whole fiber. Hence $\mathcal{J}_B/\equiv_{\mathcal{C}_B}$ is the coarsest sufficient representation extensionally. This information-theoretic argument gives no efficient selector, computability, or storage-optimality claim. A system must retain such a representation, provide later authenticated access, or restrict $\mathcal{C}_B$; snapshots and PBCE are sufficient, non-unique designs.

The condition has a simple quantitative consequence. Let the source object contain n independently varying fields over an alphabet of size q, and let $\mathcal{C}$ contain the n consumers that return one designated field. Any universally sufficient issue-time representation must then distinguish all q^n objects. Hence its range has cardinality at least q^n, and a fixed-length binary representation needs at least $\lceil n \log_2 q \rceil$ bits. Adaptive opening does not remove this origin- side information requirement; it changes which authenticated facts are revealed to a particular later consumer.

Two distinct binding properties. Semantic certificate soundness binds source/artifact, P, interpreter, z, and receiver view because each can change Equation 4. Receiver identity, purpose, invocation, and expiry instead enforce *authorization-context non-replay*: substituting them need not change the object-level result, but must invalidate permission to use it. Appendix D defines these separately. Freshness/rollback additionally require a trusted clock and key/registry policy. A concrete defect signed the tree digest but not $|A_s|$; Equation 7 now binds both.

Evidence backends. FULL-SNAPSHOT signs $\mathsf{enc}(x_s)$ and reconstructs it at handoff. PBCE signs a commitment to A_s and proves only adaptively opened atoms. Both are conditionally exact for executions in the formal fragment; their distinction is authenticated access scope and systems cost. When complete transfer is allowed it is the default baseline. PBCE is justified only by an explicit access constraint or declared sparse-access budget; otherwise the system uses the snapshot or abstains.

3 METHOD: AUTHENTICATED LATE-BOUND EVALUATION

3.1 ISSUE AN ADMITTED SOURCE OBJECT

At creation, an authorized attestor resolves provenance to immutable E_s, executes the source adapter, and applies recursive value admission. The canonical encoder rejects duplicate keys and all values outside $\mathcal{J}$. The resulting tree can be issued in either of two forms.

Full snapshot. FULL-SNAPSHOT signs a body containing the canonical source tree, its digest, runtime/adapter closure, format version, issuer, and freshness metadata. At handoff the receiver verifies, reconstructs, and invokes the same closed-IR P. This is the direct baseline when disclosure is unrestricted. If the receiver already holds an authenticated common base, lossless delta reconstruction is another complete representation and must be charged under the same cache assumptions; dense access alone does not make retransmitting the full snapshot preferable.

Proof-backed archive. PBCE flattens the same tree into A_s. For each reference r it samples a nonce and constructs

$$h_r = H(\texttt{LEAF} \parallel \mathsf{Can}(r, A_s[r], \nu_r)). \tag{6}$$

Domain-separated internal nodes form a Merkle tree; the public root binds both tree digest and leaf count,

$$\rho = H(\texttt{ROOT} \parallel \rho_{tree} \parallel |A_s|). \tag{7}$$

The signed receipt binds the same lifecycle metadata as the snapshot. At handoff, the archive service runs hash-bound P against the ideal archive, records adaptive reads, and coalesces duplicate paths into a shared compact multiproof. The receiver verifies that multiproof and deterministically replays P from the proof cache. Acceptance requires equality of the ordered prover and replay query transcripts as well as equality of their tagged outputs; a missing cache read, extra read, reordered read, or output mismatch abstains. This backend proves absence with complete parent `shape` atoms; this disclosure is format-specific, not inherently necessary.

Two-pass compact protocol. The prover computes $(r_s, q_s) \leftarrow \mathrm{Eval}(P, O_A, z)$, forms one compact multiproof for the distinct references in q_s, and binds (P, z, r_s, q_s, ρ) in the decision record. The receiver (i) verifies the receipt and multiproof, (ii) installs only verified atoms in a proof cache, (iii) reruns P with cache misses mapped to Abort, and (iv) accepts the source replay only if $(r'_s, q'_s) = (r_s, q_s)$. It separately evaluates P on the admitted receiver object to obtain (r_r, q_r), and compares only r_s with r_r. For m distinct opened leaves in an n-leaf binary tree, the union proof uses at most $m\lceil \log_2 n \rceil$ sibling hashes; unlike a full snapshot, its byte cost also depends on the accessed values and locality (Appendix E).

Selective backend variants. We also measure compressed multi-fact blocks and hash-chained keyed events with an authenticated latest-head map. Both preserve the same consumer semantics while changing surplus disclosure, proof framing, and repeated-query cost; Appendices E and F give their algorithms and the event log's single-writer model.

Late subconsumer projection. Write $q' \preceq_\pi q$ when each target clause is discharged by either its exact feature or an authenticated first-blocking absence witness in proof π_q. This artifact-relative relation strictly includes textual subsets. An untrusted holder derives ordinary $\pi_{q'}$ without the archive; fresh owner authorization binds target and context, uncovered targets abstain, and the unchanged verifier obtains the direct archive result (Appendix R).

An encrypted snapshot or access-controlled field service is a valid deployment alternative. Encryption protects storage/transport but still reveals the whole snapshot after receiver decryption; an ordinary field service minimizes transfer but asks the receiver to trust each response. Adding authenticated query proofs yields the same generic ADS design class as PBCE. Our contribution is not that primitive, but the late-authorized, same-consumer replay protocol, closed IR boundary, and measured choice against the complete snapshot. Issue-time slicing cannot target an unknown future P; authorization-time slicing still requires authenticated consumer-complete source evidence, so it is an optimization within this lifecycle rather than a distinct trust model.

3.2 Authorize and compare a restricted, checkable consumer

A handoff authorization binds hashes of P, interpreter profile, z, artifact, source receipt, receiver materializer/runtime closure, purpose, invocation and expiry. The receipt itself binds neither P nor the interpreter: both may be chosen after issuance, and a newly admitted opcode needs a newly authorized profile rather than artifact reissuance. The same P executes over the source backend and current receiver object. The final decision record additionally binds the complete tagged receiver outcome and, on acceptance, the canonical digest of the exact materialized receiver view x_r. Only successful canonical results and explicit structured rejections are comparable; every unhandled exception abstains. Decision records bind both inputs—including $H(\mathsf{enc}(x_r))$ when present—domain results, semantic query traces, proof traces, and implementation versions. The proof trace authenticates q_s but is never compared with receiver query trace q_r. This is a receiver-local admission result: current materialization and execution are trusted. A remote party must trust the receiver, independently replay the bound view, or add TEE/verifiable computation; the signature alone is not remote attestation. The semantic certificate concerns the first group of bindings; receiver, purpose, invocation, and expiry enforce a separate authorization-context check. A signature alone does not provide freshness or rollback resistance.

The certifying path does not accept JavaScript. P is a serialized, hash-bound declarative IR whose validator rejects unknown opcodes, parameters, keys, inputs, and excess resource bounds. Closed IR v2 has two modes under one envelope. Its transform fragment contains eight object-transform families (drop, resolve scalar/array/object paths, nested resolution, credential replacement, and conditional set); the evaluated Vapi consumer uses 12 concrete operations. Its observation fragment contains two bounded operations: `observe_tree`, whose expansion must be authenticated and root-to-leaf closed, and `observe_paths`, an ordered, closed list of value/shape observations fixed by a task consumer. Both are finite compositions of the core observations in Appendix C.1 and fresh-local construction. Thus validator acceptance implies syntactic membership in the calculus by structural induction over the operation list; there is no AST-checker premise on the certifying path and no additional open-ended expression language is quantified over by Lemma 2. A restricted fail-closed frontend compiles supported AST patterns from the pinned repository to closed IR and

rejects all others. The IR is the certified policy semantics; repository correspondence is optional and evaluated rather than assumed as a theorem.

A finite-domain bridge gate hash-binds source, compiler, IR, corpus, and native results, then independently revalidates every point; failed or unexercised evidence yields UNKNOWN. Appendix D defines its pointwise lifting guarantee; it is not universal refinement.

For two admitted source fragments we remove the frontend from this empirical bridge. An untrusted compiler emits (P, π_B), where π_B binds the source envelope, complete control flow, configuration, source spans, and target IR. Separately implemented checkers V_B, which import neither compiler nor legacy frontend, recognize five AutoGen termination rules and one LangGraph explicit-ID message reducer. Unknown statements or predicates reject. Appendix P gives the rules and boundaries.

3.3 SEMANTIC BRIDGE AND AUTHENTICATED EXECUTION

The reasoning has two independent layers.

Lemma 1 (native–archive round trip). For every $x \in \mathcal{J}$,

$$\mathsf{mat}(\mathsf{flat}(\mathsf{enc}(x))) \simeq_{\mathcal{J}} x. \tag{8}$$

The proof is structural induction over scalars, dense arrays, records and the distinguished string map. Excluded JavaScript objects are outside the premise.

Lemma 2 (core-operation simulation). Every admitted observation has the same result and successor local store on x, eager reconstruction, and the ideal lazy view of its archive. Container cases follow from complete `type`/`shape` atoms, including the authenticated own-key order; leaf cases use `value`; the two proof-erased guards follow from admission invariants. Induction over an admitted deterministic program therefore yields equal domain results and semantic query traces,

$$\mathrm{Eval}(P, x, z) = \mathrm{Eval}(P, \mathsf{mat}(A), z) = \mathrm{Eval}(P, \mathsf{lazy}(A), z). \tag{9}$$

Authenticated openings. In game AuthOpen, an adversary wins by producing an accepted present atom different from deterministic archive resolution, or an accepted absence witness that is not the first unresolved path step. Absence opens the blocking parent shape/type atom; it is not assumed from failure to find a leaf. Under collision resistance, unambiguous encoding, unique references, canonical path resolution, domain separation, well-formed source attestation, and structural verification, a winner yields a hash collision or violates the explicit attestation premise. This is the conventional authenticated-data-structure guarantee (Miller et al., 2014); mutation tests are not its proof.

Lemma 3 (authenticated source replay). For admitted x_s, P, z, terminating execution, verified openings, and no opening forgery,

$$\mathrm{Eval}(P, O_\rho, z) = \mathrm{Eval}(P, x_s, z). \tag{10}$$

This composes Lemma 2 with AuthOpen correspondence; faithful source attestation remains a premise.

Theorem 4 (IR-relative authenticated certificate soundness). Under Lemma 3, faithful admitted receiver materialization, one hash-bound interpreter profile, a decision record bound to the exact receiver view and materializer/runtime closure, valid context/bounds, and no abort,

$$\textsc{Cert-Compatible} \Rightarrow \mathrm{Compat}(P, z; x_s, x_r), \qquad \textsc{Cert-Breaking} \Rightarrow \neg\,\mathrm{Compat}(P, z; x_s, x_r).$$

Detected validation failures yield UNKNOWN; undetected failures of trusted attestation or materialization invalidate the guarantee. The result is IR-relative; lifting it to repository consumer C requires $\mathsf{Bridge}(C, P, D) \iff \forall x \in D, z : \mathrm{Eval}(C, x, z) = \mathrm{Eval}(P, x, z)$. Signatures/openings establish archive integrity; attestor/materializers remain trusted, and repository-to-IR correspondence is tested, not proved. Appendix D separates the standard opening argument, conditional comparison consequence, trusted premises, and bridge evidence.

Corollary 5 (conditional bridge lifting). If $\mathsf{Bridge}(C, P, D)$ holds, both views belong to D, and P has the archive simulation required by Lemma 2, Theorem 4 lifts to C's observable results. For AutoGen, complete-tree reconstruction followed by the bound predicate interpreter supplies this simulation (Appendix P). Its checker supplies source correspondence on the fresh-instance projection. The corrected LangGraph v2 reducer uses the same reconstruction argument on its explicit-ID input domain; native conversion remains an identity premise. Neither profile adds opcodes to Closed IR v2 or proves arbitrary continuation.

3.4 Admission and backend policy

The prototype applies recursive value admission, closed IR validation, and native/eager/lazy differential tests. A legacy source-closure AST audit is reported only to diagnose what direct JavaScript admission would reject; it is not the certifying interface. Backend selection is policy-driven, not learned. Complete transfer permits FULL-SNAPSHOT or authenticated lossless reconstruction; it does not require retransmitting a full snapshot. Selective transfer permits fact archives or keyed events when their cache, authorization and budget requirements hold. Sparse reads alone do not select PBCE: keyed events win all next-action and final-result comparisons in our retained E9 workload. An infeasible backend abstains; among feasible backends, costs must be measured under the same cache and disclosure assumptions. We evaluate a frontier, not a learned selector or a validated workload-to-backend rule. Section 5 reports the observed crossover.

4 Evaluation Protocol

We separate effectiveness, transfer, coverage, and deployability; constructed carriers are fixed sets, not random ecosystem samples.

Compound omission and direct baseline. Before compound outcomes, we froze 320 Vapi GitOps cases: 40 SHA-256-ranked pairs in each of eight templates, each combining two frozen atomic changes. The manifest fixes consumer commit, policies, code hashes, and gates. Four lossy projections, an issue-time output oracle, signed full snapshot, and PBCE share one serializer and closed consumer; exact backends use 20 repetitions and must reproduce trace and output. The output-union oracle runs all known consumers at issue time and is non-deployable. Construct and finite-domain bridge diagnostics are separated in Appendix K.

Size/locality grid. Canonical 16–16,384-field objects cross five access densities in 30 cells; raw/gzip-9/Brotli-11 are applied symmetrically over ten repetitions.

Semantic and disclosure stress. Independent code compares native/eager/lazy views on ten edges and 2,000 trees; 30 cells measure cumulative disclosure, not privacy.

Official-checkpoint workload frontier. Official AutoGen, CrewAI Flow, and LangGraph persistence APIs cross teams of 2/4/8 with 3/10/30/100 events. Five fresh AutoGen/CrewAI seeds form 120 cells; LangGraph is robustness. Compact PBCE and compressed signed snapshots use four consumer scopes and six tool-value classes; sequencing and invalidated batches appear in Appendix I.

Closest selective event-log baseline. On three fresh seeds, a signed Merkle head map proves the latest hash-chained event per requested key. Raw/Brotli encodings share an ideal persistent cache over four handoffs; all 216 warm-cache verifications are reported.

Multi-ecosystem transfer and statistical unit. Transfer covers 42 AutoGen, 12 Plaid, and 16 Datadog directed release edges, including 30 prospectively receiver-blind AutoGen directions: 70 edge-level units, 28 breaking. The 30 exhaust the complement of 12 prior adjacent directions among all $7 \times 6 = 42$ ordered pairs. A 158-row interface funnel, including four issuance failures, appears in Appendix J.

Executable multi-Agent task continuity. A separate manifest freezes all 42 directed pairs among seven AutoGen releases and a nine-clause consumer over a resumed three-message trajectory. Offi-

cial `save_state`/`load_state`/`run` execute only after all receipts exist; the pair is the unit and complete-tree equality the comparator.

Independent native-assessment transfer. After train-scenario development, we freeze the adapter and all 57 official AppWorld dev tasks (19 disjoint scenarios) before opening their content (Trivedi et al., 2024). Official validation solutions supply initial/final states. Native assessments are compared with detached archive execution; unsupported tasks remain in the denominator. This empirical adapter is outside the closed-IR theorem (Appendix T).

5 RESULTS

We separate result fidelity, certifiable coverage and transfer cost. Snapshot savings do not establish superiority over another selective authenticated store.

5.1 DOES REPLAY PRESERVE THE ADMITTED CONSUMER'S RESULT?

Both full snapshots and PBCE match all 320 compound-challenge decisions; branch-union tracing reaches 229, with 12 false-safes and 79 over-quarantines. The guarantee comes from consumer-complete authenticated evidence, not slicing alone (Appendix Table 18). Source checkers validate 12 configured AutoGen translations and one LangGraph reducer, rejecting 77 substitutions and 13 source mutants. AutoGen's retrospective source audit accepts 32/40 family–release cells across four distinct files.

LangGraph v2's complete-input composition matches all 936 retained-corpus pairs: 483 compatible, 390 breaking and 63 unknown; all 873 certified output pairs and records verify. This is integration regression, not fresh transfer; the v1 suffix failure remains reported (Appendix P).

5.2 COVERAGE AND TASK SCOPE

Table 1 follows all 11 pinned AutoGen termination classes at one-call and two-call horizons; unsupported classes remain included. All 44 official executions complete, but only 5/22 workloads certify correctly: 12 lack expressible semantics and five exceed the checked fresh-instance domain. This single-module inventory uses constructed messages, not sampled production tasks (Appendix P.2).

Table 1: Cumulative coverage on the same 22 inventory workloads. Grants vary over the same tasks. Shape denial concerns the full-tree implementation; it does not establish minimum necessary disclosure.

Stage	Full grant	No shape	One atom
Inventory / JSON objects	22	22	22
Expressible consumer	10	10	10
Checked bridge	5	5	5
Evidence permission	5	0	5
Budget / correct certificate	5	0	0

An earlier 42-pair AutoGen release study tests actual saved-state continuation: the sparse consumer matches all pair-level task outcomes and all 22 compatible pairs resume exactly; full-tree equality over-quarantines two. Its fixed deterministic workflow supports that task oracle, not arbitrary continuation (Appendix J).

On 57 AppWorld development tasks, replay coverage rises from 33/57 to 57/57 after language/relation repairs and trusted complete-index issuance. All 114 phase vectors match, with no prior-task regression. This is empirical correspondence, not checked translation (Appendix T).

Two prospectively selected 36-task test-normal cohorts exclude recorded local exposure. Frozen replay covers 24/36 and 18/36; post-outcome repairs reach 35/36 and 36/36, while restored Agent success is 10/36 and 15/36. The second cohort retains nine network-incident tasks with no provider response. All 36 native checkpoints restore equally; repaired replay matches 72 phase vectors without prior-task regression. Repairs are development, not independent transfer (Appendix T.9).

5.3 When selective backends reduce transfer

Small dense objects favor snapshots: 1.83 KB versus PBCE's 21.05 KB. PBCE wins 17/30 raw and 12/30 compressed size/locality cells. On 27 controlled framework–seed–consumer units, cached lossless DAG reconstruction uses a median 19.581× PBCE's cumulative bytes and loses all comparisons (Table 10). These are not independent production tasks; caches are ideally synchronized and negotiation/WAN latency is excluded.

Selective alternatives can be cheaper. In E9's 27 units, compressed events/PBCE is 0.737 at the median: events win all 18 next-action/final-result units; PBCE wins six CrewAI/LangGraph trace units (Table 11). Inspect's generic map also wins both samples with equal scores and SDK trust (Appendix S).

On 33/57 AppWorld tasks, queries use 0.630× compressed reconstruction bytes; 330/330 replays match, but one state costs 2.5% more (Appendix T.1). Initial encoding wins 9/33 comparisons against the author DAG; digest references win 33/33 with 1.9% median savings after equally optimizing both arms. Uniform-profile public replay supports and wins 12/12 with trace-sized packets: savings are 13.8%, or 10.2% excluding preflight (Appendix T.6). These exposed response-byte results imply neither general superiority nor end-to-end gains.

On 35 restored development checkpoints, selective first-handoff time is 0.157× compact full and 0.114× root-authenticated lazy full ZIP; all 2100 queries agree. Full packages are prebuilt; selective sender work is charged. Warm queries are nearly tied. This is local handoff, not whole-task speedup or generic-store superiority (Appendix T.10).

Appendix N's controlled authoring studies establish no algorithm advantage.

6 Limitations and Threat Model

Soundness is handwritten and profile-relative; only six source schemas have checked translation. Admission covers 8/18 tool observations and 33/130 frontend functions in separate cohorts. Adapters, runtimes, receiver, crypto/OS, registry and freshness remain trusted. Receiver-visible paths, shapes, opened values, sizes and repeated accesses can leak information; this is not a privacy guarantee. Archives see full data/accesses. Resource bounds cause abstention, not denial-of-service resistance. Loopback omits WAN/TLS, multi-host SLOs and rollback/GC. Finite-oracle synthesis and constructed workloads establish neither production prevalence nor broad task coverage. AppWorld's original 11/19-scenario coverage rises to 19/19 only after development, including trusted complete-index issuance. Its source-root binding alone does not prove index fidelity. Empirical evaluator correspondence establishes neither checked translation nor completeness for user intent. Repairing the later cohort from 18/36 to 36/36 establishes neither fresh transfer nor whole-Agent speedup.

7 Related Work

Registries/diffs assume declared changes; attestations authenticate records, not late paired semantics (Confluent, 2026; OASDiff contributors, 2026; Rhodes and Kang, 2026). ADS, proof reuse and translation validation are established (Miller et al., 2014; Brun and Traytel, 2019; Gregersen et al., 2025; Pnueli et al., 1998); our lifecycle applies them to late authorization and paired evaluation (Appendix G). Prior authenticated queries address fine-grained access conflicts (Xu et al., 2018), and TAP combines verification with privacy (Reijsbergen et al., 2023). Evidence-policy conflicts alone are therefore not our novelty; PBCE does not provide their zero-knowledge confidentiality.

8 Conclusion

Late evaluation needs consumer-complete evidence. Full snapshots, cached lossless reconstruction, and selective backends occupy different regimes determined by access scope, shared state, and verification cost. The evaluated savings are conditional; no universal backend advantage or ecosystem claim follows.

REFERENCES

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. arXiv:2407.18901, 2024. https://arxiv.org/abs/2407.18901.

Cheng Xu, Jianliang Xu, Haibo Hu, and Man Ho Au. When Query Authentication Meets Fine-Grained Access Control: A Zero-Knowledge Approach. In *Proceedings of the 2018 International Conference on Management of Data*, pages 147–162, 2018. https://doi.org/10.1145/3183713.3183741.

Daniël Reijsbergen, Aung Maw, Zheng Yang, Tien Tuan Anh Dinh, and Jianying Zhou. TAP: Transparent and Privacy-Preserving Data Services. In *32nd USENIX Security Symposium*, pages 6489–6506, 2023. https://www.usenix.org/conference/usenixsecurity23/presentation/reijsbergen.

Matthias Brun and Dmitriy Traytel. Generic Authenticated Data Structures, Formally. In *10th International Conference on Interactive Theorem Proving*, LIPIcs 141, Article 10, 2019. https://doi.org/10.4230/LIPIcs.ITP.2019.10.

Simon Oddershede Gregersen, Chaitanya Agarwal, and Joseph Tassarotti. Logical Relations for Formally Verified Authenticated Data Structures. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, 2025. https://doi.org/10.1145/3719027.3744801.

Alessandro Chiesa and Eran Tromer. Proof-Carrying Data and Hearsay Arguments from Signature Cards. In *Innovations in Computer Science*, pages 310–331, 2010.

Andrew Miller, Michael Hicks, Jonathan Katz, and Elaine Shi. Authenticated Data Structures, Generically. In *Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 411–424, 2014. https://doi.org/10.1145/2535838.2535851.

Ben Laurie, Eran Messeri, and Rob Stradling. Certificate Transparency Version 2.0. IETF RFC 9162, December 2021. https://www.rfc-editor.org/rfc/rfc9162.html.

Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly Practical Verifiable Computation. *Communications of the ACM*, 59(2):103–112, 2016. https://doi.org/10.1145/2856449.

Robert N. M. Watson, Jonathan Anderson, Ben Laurie, and Kris Kennaway. Capsicum: Practical Capabilities for UNIX. In *19th USENIX Security Symposium*, pages 29–46, 2010.

Confluent. Schema Evolution and Compatibility Types. https://docs.confluent.io/platform/current/schema-registry/fundamentals/schema-evolution.html, accessed 2026.

George W. Dunlap, Samuel T. King, Sukru Cinar, Murtaza A. Basrai, and Peter M. Chen. ReVirt: Enabling Intrusion Analysis through Virtual-Machine Logging and Replay. In *5th USENIX Symposium on Operating Systems Design and Implementation*, 2002.

Daniel Fett, Kristina Yasuda, and Brian Campbell. Selective Disclosure for JSON Web Tokens. IETF RFC 9901, November 2025. https://www.rfc-editor.org/rfc/rfc9901.html.

Vivek Haldar, Deepak Chandra, and Michael Franz. Semantic Remote Attestation—A Virtual Machine Directed Approach to Trusted Computing. In *Proceedings of the Third Virtual Machine Research and Technology Symposium*, pages 29–41, 2004.

Microsoft. AutoGen AgentChat Teams: Saving and Loading Group-Chat State. https://microsoft.github.io/autogen/stable/reference/python/autogen_agentchat.teams.html, accessed 2026.

OpenAI. OpenAI Agents SDK: Sessions. https://openai.github.io/openai-agents-python/sessions/, accessed 2026.

Google. Agent Development Kit: Session and State Tests. https://github.com/google/adk-python/tree/c7ef8cfa8269738033180f49338f160254b8b137/tests/unittests/sessions, accessed 2026.

Pydantic. PydanticAI: Message History and Deferred Capability Tests. https://github.com/pydantic/pydantic-ai/blob/26467f6f3002784265709335a9dc5cb636b15d53/tests/test_capability_deferred_loading.py, accessed 2026.

Ralph C. Merkle. A Digital Signature Based on a Conventional Encryption Function. In *Advances in Cryptology—CRYPTO '87*, pages 369–378, 1988. https://doi.org/10.1007/3-540-48184-2_32.

George C. Necula. Proof-Carrying Code. In *Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 106–119, 1997.

OASDiff contributors. OpenAPI Diff and Breaking Changes. https://github.com/oasdiff/oasdiff, version 1.29.1.

Amir Pnueli, Michael Siegel, and Eli Singerman. Translation Validation. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 1384 of LNCS, pages 151–166, 1998.

Santhosh Kumar Ravindran. Portable Agent Memory: A Protocol for Cryptographically-Verified Memory Transfer Across Heterogeneous AI Agents. arXiv:2605.11032, 2026.

James Rhodes and George Kang. Proof of Execution: Runtime Verification for Governed AI Agent Actions. arXiv:2607.05397, 2026.

Zexun Wang. CAVA: Canonical Action Verification and Attestation for Runtime Governance of Agentic AI Systems. arXiv:2607.13716, 2026.

Adarsh Vatsa, Sachi Shome, Yingming Zhou, and William Eiers. AutoCedar: An Agentic Framework for Verifier-Guided Access Control Policy Synthesis. arXiv:2607.03656, 2026.

Cheng Wen, Junjie Hu, Yikun Hu, Jie Su, Bin Yu, Dugang Liu, Zhiwu Xu, Weidi Sun, Shengchao Qin, and Cong Tian. Formally Specifying the Intended Behavior of the Program: LLM-Driven Neuro-Symbolic Program Specification Synthesis. In *Proceedings of ACL: System Demonstrations*, pages 672–682, 2026. https://doi.org/10.18653/v1/2026.acl-demo.66.

Mark Weiser. Program Slicing. In *Proceedings of the 5th International Conference on Software Engineering*, pages 439–449, 1981.

AI USE STATEMENT

Generative AI tools assisted hypothesis refinement, methodology feedback, implementation and analysis, literature discovery, and manuscript editing. The authors are responsible for verifying the cited literature, frozen manifests and programmatic outputs, and for all claims and artifacts.

REPRODUCIBILITY STATEMENT

Reproduction has distinct levels; a passing ledger audit is not a fresh execution of every experiment. The artifact includes protocol/code hashes, test and analysis scripts, retained results, source-revision identifiers, verification code, and the invalidation history. External timestamps bind byte commitments, not investigator blindness.

Root audit. The script `scripts/verify_pbce_submission.sh`, invoked with Bash, runs the current test suite, saves its exact outcomes in JUnit XML, checks frozen hashes, reconstructs selected counts from retained outputs, compiles the PDF and checks the nine-page main-text boundary. It does not rerun model APIs, native SDK workloads, AppWorld trials or latency studies.

Recomputation from retained evidence. Frozen result rows support reconstruction of reported counts and ratios. Separate transport audits actually decompress retained packets and check their hashes, lengths and byte accounting; native agreement in those audits is checked against execution logs, not independently rerun. After lossless storage archival, some original large input paths require restoration using the recovery index before historical commands can run. Compressed copies retain exact-byte recovery.

Native and external reruns. Native-workload scripts require the pinned repositories, runtimes and input artifacts specified by their manifests. Repeating model-generation studies also requires the corresponding service and model access; these are not exercised by the root command. Loopback and measured-input network scenarios do not reproduce a multi-host WAN deployment.

AppWorld access boundary. The AppWorld studies require a separate authorized local AppWorld installation and the applicable source data. Protected instructions, solutions, test source and state derivatives are not redistributed; only permitted scripts and aggregate results may be packaged. Restricted redistribution is not evidence that an authorized rerun is impossible, but a clean third-party rerun has not been established here. The root audit must not be presented as that reproduction. This empirical adapter is also outside the closed-IR theorem.

A Authenticated Late-Bound Evaluation Architecture

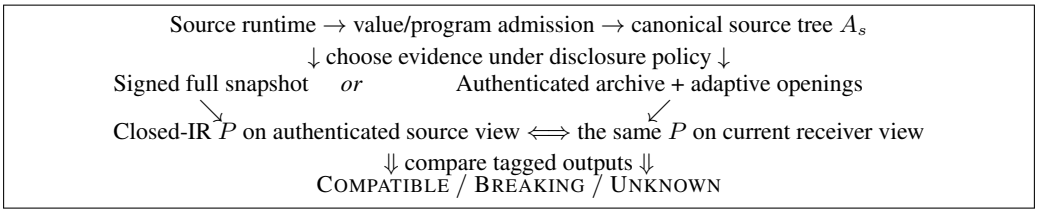

Figure 2: Full snapshots and selective authenticated archives are evidence backends for the same source semantics, not accuracy competitors.

B PBCE Algorithms and Wire Binding

Issue. Given artifact a, source runtime E_s, adapter χ, and issuer policy, the attestor verifies the immutable runtime closure, computes archive A_s, rejects duplicate references or unsupported canonical values, samples an independent 128-bit nonce per fact from the OS CSPRNG, constructs Equation 7, and signs a receipt binding:

$$H(a), \rho, |A_s|, H(E_s), H(\chi), H(\mathsf{Can}), \mathsf{issuer}, \mathsf{key}, \mathsf{time}, \mathsf{format}.$$

The receipt is public; fact atoms, nonces, and tree nodes remain in the archive. The nonce prevents a root-only observer from confirming guesses for low-entropy unopened atoms and prevents deterministic leaf equality across receipts. It provides neither confidentiality from the archive/prover nor access-pattern hiding. Its 128-bit value-hiding parameter is distinct from SHA-256 collision binding; compromise of the archive reveals both atom and nonce and removes this guessing resistance. The prototype relies on CSPRNG independence and keeps no cross-receipt nonce registry; for N sampled leaves the accidental-collision bound is at most $N(N-1)/2^{129}$. Nonces are a hiding salt, not an authenticity key.

Verify and execute. The handoff verifier authenticates receipt, artifact, issuer/key policy, freshness and revocation, source closure, archive format, and current receiver closure. A consumer authorization binds

$$H(P), H(K), H(z), H(a), H(\mathsf{receipt}), H(E_r, \chi_r), \mathsf{purpose}, \mathsf{invocation}, \mathsf{expiry}.$$

It executes P on source oracle O_ρ and receiver materialization A_r. The resulting decision record binds the complete tagged receiver outcome and, when materialization accepts, $H(\mathsf{enc}(x_r))$, the

canonical digest of the exact receiver view produced by χ_r for this invocation. Each source observation appends a semantic row to query trace q_s; its Merkle opening is stored separately in proof trace ϕ_s. Receiver evaluation produces (r_r, q_r) without a proof trace. Any malformed envelope, unauthorized substitution, invalid opening, ambient access, timeout, or unsupported value returns UNKNOWN.

These checks establish two properties. The semantic tuple $s = (H(a), H(\mathsf{receipt}), H(P), H(K), H(z), H(E_r, \chi_r), H(\mathsf{enc}(x_r)))$ fixes the compatibility statement, where K is the interpreter profile. The context tuple $c = (\mathsf{receiver}, \mathsf{purpose}, \mathsf{invocation}, \mathsf{expiry})$ fixes who may use that statement and when. A decision may remain semantically correct after changing c yet be unauthorized; that is not a semantic false-safe.

Canonical fact domain. Internally, a path is a typed sequence of record-key strings and nonnegative array indices. Its wire form is an RFC 6901 pointer; decoding must consume the entire string, accept only canonical RFC 6901 escaping, and re-encoding the typed path must reproduce the received pointer. Component interpretation is fixed by the authenticated container type, so an object key "0" is not silently treated as an array index. References are unique pairs of canonical pointer and feature tag. Scalars have `exists`, `type`, and `value`; containers have `exists`, `type`, and complete `shape`. Record shapes preserve ECMAScript own-key order, while Merkle leaf references are sorted independently. Arrays preserve order; binary64 atoms commit exact bits (so $-0 \neq +0$); nonfinite numbers are rejected. Strings are committed without Unicode normalization. A child absent from an authenticated parent shape is absent in the committed archive. The receipt binds the source runtime/encoder profile. Replay consumes the committed source order rather than recomputing it in the receiver engine; receiver-side enumeration uses the receiver's admitted order, so an observable engine-order difference remains a compatibility difference.

Path resolution and absence witnesses. Let $p = (c_1, \ldots, c_k)$. Resolution starts at the authenticated root and follows components in order. At the first unresolved component c_j, an absence witness is $w = (p_{<j}, c_j, u)$, where u is an authenticated atom for the longest existing prefix: (i) an object `shape` omitting string key c_j; (ii) an array `shape` of length n with integer $c_j \notin [0, n)$; or (iii) a scalar `type`, proving that no child can exist. Successful resolution instead requires membership openings for every queried terminal atom. The verifier checks canonical pointer decoding, container/component kind, every opening, and that w is the first failed step. Multi-level missing paths reduce to this first unresolved prefix; a later descendant claim is ignored. Wrong-container reads and array bounds therefore have explicit, distinct witnesses. This property is conditional on the trusted attestor having built a well-formed archive from admitted x_s; the flat Merkle root alone does not prove cross-atom tree well-formedness.

The native admitted value is a finite acyclic JSON tree plus one distinguished string-to-string map. Issuance also rejects `undefined`, symbols, accessors, nondefault descriptors/prototypes, sparse arrays, cycles, shared object identity, class instances, methods, and non-string map entries. These checks are semantic premises, not conveniences of serialization.

C ADMITTED OBJECT SIMULATION

Write $x \simeq_{\mathcal{J}} x'$ when two admitted values are recursively equal, including binary64 bits and record own-key order, after distinguished-map normalization. Structural induction gives

$$x \simeq_{\mathcal{J}} \mathsf{mat}(\mathsf{flat}(\mathsf{enc}(x))).$$

For records, authenticated `shape` fixes the complete key set and each child follows by induction; arrays additionally fix dense length/order; scalar values are direct atoms. The string map is encoded as a distinguished record and exposes only `get` and `size`.

The admitted rules are pure primitive computation, member read, membership, authenticated own-key enumeration, array length/index/traversal, local construction/update, and string-map operations. Each rule is simulated by the eager and lazy views. General identity/prototype/descriptor operations are not rules of this calculus. The evaluated Vapi closure contains a plain-record prototype test and a `WeakSet` cycle guard; admission proves their answers in advance (plain and acyclic), so they are proof-erased. This argument would not admit a consumer whose output depends on alias identity.

C.1 CORE OBSERVATION CALCULUS

Values are

$$v ::= \mathsf{null} \mid \mathsf{bool}(b) \mid \mathsf{f64}(h) \mid \mathsf{str}(u) \mid [v_1, \ldots, v_n] \mid \langle (k_1, v_1), \ldots, (k_n, v_n) \rangle \mid \mathsf{strmap}(K),$$

where h is an exact finite binary64 bit string, u is a UTF-16 code-unit sequence, record keys are unique, and $(k_1, \ldots, k_n)$ is the native `Object.keys` sequence. Input observations are

$$o ::= \mathsf{get}(p, k) \mid \mathsf{has}(p, k) \mid \mathsf{keys}(p) \mid \mathsf{len}(p) \mid \mathsf{index}(p, i) \mid \mathsf{mapget}(p, k) \mid \mathsf{mapsize}(p).$$

Their rules return the corresponding child/value, membership bit, committed key sequence, or length. Missing record reads return a language-level missing token, which cannot be stored as an admitted value. There is no additional open-ended expression language in the theorem premise: admitted programs are exactly the validated Closed IR v2 operation lists below. Their pure tests are the type guards, exact tagged equality, mapped-key lookup, and explicit nullish predicate stated in the transition table; none can inspect identity, prototypes, descriptors, or ambient state. A result outside the tagged output grammar abstains.

Total observations and missing. Let M be a distinguished result token, not an admitted value. For a record $r = \langle (k_1, v_1), \ldots, (k_n, v_n) \rangle$, $\mathsf{get}(r, k_i) = v_i$, $\mathsf{get}(r, k) = \mathsf{M}$ for absent k, $\mathsf{has}(r, k) = [\mathsf{get}(r, k) \neq \mathsf{M}]$, and $\mathsf{keys}(r) = (k_1, \ldots, k_n)$. For a dense array $a = [v_0, \ldots, v_{n-1}]$, $\mathsf{index}(a, i) = v_i$ when $0 \leq i < n$ and M otherwise, while $\mathsf{len}(a) = n$. Applying an observation to the wrong container kind returns M. String-map lookup and size are analogous. Nullish is defined exactly as $\mathsf{nullish}(v) \equiv (v = \mathsf{null} \vee v = \mathsf{M})$; thus null and missing are distinguished until this explicit test.

Machine state, domain results, and aborts. An execution configuration is $\langle P, i, y, z, j, b, t \rangle$: admitted immutable input i, fresh local output $y = \mathsf{clone}(i)$, explicit auxiliary string maps z, next operation j, remaining budget $b = (b_{op}, b_{depth}, b_{node}, b_{byte})$, and ordered observation transcript t. Local update $y[p \leftarrow v]$ allocates only fresh records or arrays and never mutates or aliases i. Validation failure, a non-admitted input/auxiliary value, missing required program input, an unhandled exception, or exhaustion of any budget steps to $\mathsf{Abort}(d)$; at the protocol layer every abort maps to UNKNOWN, never CERT-COMPATIBLE or CERT-BREAKING. Otherwise the finite operation list steps deterministically to $\mathsf{Ok}(v)$ or an explicitly specified $\mathsf{DomainReject}(c, v)$. Only the latter is a comparable domain result; it is not a synonym for interpreter rejection. The implementation caps operations at 64, depth at 128, nodes at 10^6, and canonical output at 64 MB; each authorized profile binds smaller limits in the program hash. The operation counter charges top-level IR instructions: `observe_tree` is one instruction, while its recursive work is charged independently to node, depth, and byte budgets.

Closed IR v2. The JSON envelope contains exactly `version`, `mode`, `limits`, and `ops`. Unknown keys, opcodes, or parameters reject. Safe path components exclude `__proto__`, `prototype`, and `constructor`; section identifiers range over the six bound state maps. The transform fragment is defined by the following exhaustive table. Here $R_s(z)$ is the right-biased UUID-to-name reverse map for section s, $\widehat{R}_s(v) = R_s(v)$ for a mapped string and v otherwise, and a type guard that does not match makes the operation a no-op.

Opcode	Exact state transition on fresh output y
`drop_top`(F)	Retain, in order, exactly top-level entries whose key is not in F.
`resolve_array`(p, s) `rename_resolve_array` (f, q, s)	If $y[p]$ is an array, replace each string element v by $\widehat{R}_s(v)$. If top-level $y[f]$ is an array, write its resolved copy to $y[q]$ and delete f.
`resolve_array_objects` (p, f, s)	In array $y[p]$, replace string field f of each record by $\widehat{R}_s(v)$.
`resolve_nested_` `array_objects`(p, n, f, s)	Apply the preceding field rule to each record in nested array n of every record in $y[p]$.
`resolve_scalar`(p, s) `replace_credentials` (f, F)	If $y[p]$ is a string, replace it by $\widehat{R}_s(y[p])$. Finite depth-first traversal; replace string field f and string elements of array field F using explicit credential map z_c.
`set_if_nullish` (p, q, true)	If immutable input read $i[p]$ is null or M, set fresh top-level $y[q]$ to true.

The observation fragment has two opcodes. `observe_tree`(p) rejects a missing root and otherwise emits a deterministic preorder transcript: an object emits its native own-key sequence and recurses over those keys; an array emits its length and recurses over indices $0, \ldots, n-1$; a scalar emits its exact tagged value. Compilation to proof-package format v2 (PPF2) opens exactly one `shape` fact per container and one `value` fact per scalar. The independent receiver checks root-to-leaf closure before comparison. In contrast, `observe_paths`$((p_i, f_i)_{i=1}^{k})$ is a nonempty, ordered, duplicate- free list of value/shape observations fixed in the authorized program. The independent receiver requires the authenticated source transcript to match that list exactly, then evaluates the same clauses on the receiver. Thus its completeness is relative to a declared task consumer rather than the entire checkpoint tree.

For both modes, structural induction over the exact-key validator and the finite operation list gives

$$\mathsf{validateIR}_{v2}(P) = \mathsf{true} \Longrightarrow P \in \mathcal{C}_{core} \ \wedge \ \mathsf{steps}(P, x) \leq c_P\,|x|,$$

where c_P is fixed by the authorized program; the bound is additionally cut off by b_{node} and b_{byte}. This implication concerns the IR itself. Equivalence between repository TypeScript and a manually authored transform IR is not certified; it is measured and mutation-tested separately.

For each container path p, `shape` stores the ordered key sequence or dense-array length. For each scalar path, `value` stores the tagged scalar bits. Eager materialization and the ideal lazy oracle therefore implement every rule above identically; induction over deterministic evaluation proves Lemma 2. This calculus and closed IR are the theorem premise. The legacy AST checker is only a diagnostic filter; arbitrary TypeScript does not enter the certifying path.

D Authenticated-Opening Game and Factorization Proof

D.1 Game definition

For security parameter λ, challenger $\mathcal{C}$ and PPT adversary $\mathcal{A}$ play $\mathsf{AuthOpen}$:

1. $\mathcal{A}$ selects, or is given, finite canonical archive A.
2. $\mathcal{C}$ samples leaf nonces, constructs the domain-separated tree, and gives ρ, $|A|$, and public format parameters to $\mathcal{A}$.
3. $\mathcal{A}$ adaptively requests valid openings for references in A.
4. $\mathcal{A}$ returns a canonical path/feature query together with either $\mathsf{Present}(x^*, \pi^*)$ or $\mathsf{Absent}(w^*, \pi^*)$. In the latter, w^* identifies the first unresolved component and π^* opens its blocking parent shape/type atom.
5. $\mathcal{A}$ wins iff the independent verifier accepts under ρ but the claim differs from deterministic $\mathsf{Resolve}(A, p, f)$: a present atom is absent/different, or an absence witness is not the first failed step under the authenticated parent atom.

Proposition 1 (opening binding). Assume collision resistance of H, unique unambiguous fact encoding, domain separation, and a verifier that checks leaf index/count, exact depth, ordered sibling directions, root, canonical path resolution, and absence witnesses. Then the probability that a PPT adversary wins $\mathsf{AuthOpen}$ is negligible.

Proof sketch. Compare an accepted forged leaf-to-root computation with the honest path at the claimed position. If distinct leaf encodings have one digest, they collide in H; otherwise the first level at which distinct ordered child pairs produce one parent is a collision. Invalid count, depth, direction, index, or pointer encoding either fails a structural check or creates one of these divergent inputs. For absence, the opened parent atom is an ordinary bound member: object-key omission, array bound, or scalar type deterministically establishes the first failed step. A contradicting present child would disagree with that committed atom or violate the well-formed-archive premise. Binding $|A|$ in Equation 7 is necessary: an invalidated engineering format that bound only the tree digest admitted count changes when the implied depth remained equal.

D.2 Execution statement

Let O_A be the ideal fact oracle and O_ρ the proof-checking oracle.

Supporting lemma (query/proof-trace correspondence). Conditioned on no successful AuthOpen forgery, each response in an accepted adaptive semantic query trace q_s justified by proof trace ϕ_s equals O_A at the same reference. The proof trace itself is not an input to semantic compatibility.

Proof of Lemma 3 (authenticated source replay). Suppose (i) native x_s passes $\mathcal{J}$ admission; (ii) P is deterministic, terminates, and uses only admitted operations; (iii) all other input is in content-addressed z; (iv) code, capsule, archive root, auxiliary input, query trace, proof trace, and domain result are bound; and (v) every opening verifies. Conditioned on no opening forgery, Equation 10 holds with the same domain result and semantic query trace as native execution.

Proof sketch. First apply native–archive round trip and operation simulation in Appendix C, obtaining equality among native, eager, and ideal lazy execution. Then induct over deterministic lazy execution. At an oracle step, trace correspondence gives the same fact and successor state. Any unsupported/ambient access or invalid proof is outside the premise and abstains.

Proof of Theorem 4 (IR-relative authenticated certificate soundness). Additionally assume faithful admitted receiver materialization x_r, a decision record bound to $H(\mathsf{enc}(x_r))$ and the authorized materializer/runtime closure, identical hash-bound interpreter semantics/profile for source replay and receiver evaluation, valid authorization/context, and no abort. Write source replay as $(\hat{r}_s, \hat{q}_s)$ and receiver evaluation as (r_r, q_r). The verifier first requires $(\hat{r}_s, \hat{q}_s) = (r_s, q_s)$ for the proof-bound source record; Lemma 3 identifies r_s with native source result. It then returns CERT-COMPATIBLE exactly when $r_s = r_r$ and CERT-BREAKING exactly when $r_s \neq r_r$. Consequently,

$$\begin{aligned} \textsc{Cert-Compatible} &\implies \mathrm{Compat}(P, z; x_s, x_r), \\ \textsc{Cert-Breaking} &\implies \neg\, \mathrm{Compat}(P, z; x_s, x_r). \end{aligned}$$

Detected proof, admission, authorization, and resource failures return UNKNOWN. Faithful attestation and materialization are trusted assumptions: an undetected violation need not abstain, and neither implication is then guaranteed. This is soundness relative to admitted P. For repository consumer C, lifting the result additionally requires translation validity $\forall x \in D, z : \mathrm{Eval}(C, x, z) = \mathrm{Eval}(P, x, z)$; finite differential/mutation tests do not prove this universal premise.

For finite declared $D \subseteq \mathcal{J} \times \mathcal{Z}$, define

$$\mathsf{TV}_D(C, P) \iff \forall (x, z) \in D : \mathrm{Result}(\mathrm{Eval}(C, x, z)) = \mathrm{Result}(\mathrm{Eval}(P, x, z)).$$

Finite-domain instance of Corollary 5. If $\mathsf{TV}_D(C, P)$ is valid, P satisfies Lemma 2's archive simulation, and both evaluated sides belong to D, the two conclusions of Theorem 4 also hold after replacing admitted-P native results by C's results. No conclusion follows outside D.

Proof. For $(x_s, z), (x_r, z) \in D$, $\mathsf{TV}_D(C, P)$ substitutes equal tagged repository results for the two admitted-P results. Equality and inequality are preserved by this substitution, so Theorem 4's two implications lift pointwise. The validation record establishes only the enumerated equalities relative to its trusted native-result corpus. Mutant detection demonstrates that each emitted operation is exercised but is not needed for the logical implication and does not extend D.

Trusted boundary. The proposition does not establish faithful origin materialization, signing-key authenticity, registry freshness, VM correctness, receiver-side materialization, availability, or equivalence to different native code. These are explicit TCB assumptions and empirical endpoints, not epsilon terms hidden inside a probability inequality.

Semantic binding counterexamples. The semantic tuple has simple indistinguishability counterexamples. If the source receipt or artifact identity is omitted, let artifact A have source value 0, artifact B have source value 1, and the receiver view be 0 under a field-reader P. Replaying A's valid evidence at B yields CERT-COMPATIBLE although B is breaking. If P, z, or the interpreter profile is omitted, choose source/receiver objects separated only by a missing field and two authorized profiles that respectively map missing to 0 or to a tagged missing value; a decision valid under one profile is false-safe under the other.

Authorization-context non-replay. Let an authorization signature cover canonical (s, c). In game CtxReplay, an adversary obtains valid authorizations of its choice and wins by making the verifier accept under (s, c'), $c' \neq c$, without an authorization for (s, c'). Under EUF-CMA security of the signature, collision resistance of the canonical hashes, exact tuple comparison, and an honest key registry, its advantage is negligible: acceptance reuses a signature on different bytes, forges one, or collides a bound hash. This covers receiver, purpose, invocation, and expiry substitution; it does not make them semantic inputs to Equation 4. Receipt substitution under an accepted issuer signature reduces to the same assumptions. The 44 fault-injection trials exercise these substitutions and return UNKNOWN; they are regression evidence, not the proof.

Freshness and rollback boundary. Expiry assumes a trusted clock. Rejecting an old but validly signed receipt or authorization additionally requires a monotone registry epoch, revocation state, or external transparency/timestamp policy. The prototype binds registry snapshots and checks configured expiry/revocation data, but does not prove that this external state is current. Historical key compromise is outside CtxReplay absent that policy. Authorization names an issued receipt and the decision names that authorization; this hash ordering alone does not imply freshness.

Corollary (finite field-reader lower bound). Let $X = \Sigma^n$, $|\Sigma| = q$, and let $P_i(x) = x_i$ for every $i \in \{1, \ldots, n\}$. Consumer completeness makes Π injective: if $x \neq x'$, some coordinate i differs and P_i separates them, so $\Pi(x) \neq \Pi(x')$. Therefore $|\mathrm{range}(\Pi)| \geq q^n$. A fixed-length binary encoding of that range requires at least $\lceil \log_2 q^n \rceil = \lceil n \log_2 q \rceil$ bits. This is an information bound for this explicit consumer class, not an optimality theorem for PBCE.

E COMPACT MULTIPROOF REPLAY

Let Q contain m distinct queried leaves in an n-leaf binary Merkle tree. The compact proof is the union of their authentication paths after deleting nodes derivable from opened leaves or other included nodes. Since each path has at most $\lceil \log_2 n \rceil$ siblings, the proof contains at most $m\lceil \log_2 n \rceil$ sibling hashes. This is a safe upper bound rather than a claim of optimal encoding; shared ancestry can make the union smaller. The transmitted size is therefore the signed receipt, opened atom encodings, and at most that many hashes. The full-snapshot backend instead transmits the complete canonical source tree. Neither dominates without assumptions on object size, value lengths, access density, and locality, which motivates the measured grid in Section 5.

After multiproof verification, the receiver exposes only verified atoms through a read-only cache oracle. It records every ordered cache query while rerunning the hash-bound consumer and accepts only if both the ordered trace and tagged output equal the prover-bound values. Thus an unproved extra observation is a cache miss, while a path-divergent replay is rejected even when it happens to produce the same final output. A valid but semantically unused opening does not change the result: the implementation permits it only inside the signed path/feature capability, charges it to the disclosure ledger and transmitted bytes, and reports it as surplus disclosure. It is not evidence minimality and cannot enlarge the accepted semantic query trace.

F SEQUENTIAL KEYED EVENT BACKEND

The measured event backend has a deliberately narrower state model than the fact archive. A single writer serializes complete admitted checkpoints $A_0, A_1, \ldots$ by ordinal t. For each changed reference u, it appends

$$e_{u,t} = H(\texttt{EVENT} \parallel t \parallel u \parallel \mathsf{atom}_{u,t} \parallel h_{u,t^-}),$$

where h_{u,t^-} is the previous per-key head; deletion appends a tagged tombstone. After every changed key is processed, the writer atomically signs $(\mathsf{sequence}, t, |M_t|, \mu_t)$, where $M_t[u] = h_{u,t}$ and μ_t is a domain-separated Merkle root over the complete sorted head map. This signed root, rather than append-only membership, defines the committed checkpoint. Current missing paths are justified by the current authenticated parent `shape`; tombstones prevent a deleted value event from remaining the latest head but are not themselves semantic values.

A warm receiver stores events by hash and a previously verified head $\bar{h}_u$. For requested u, a package must (i) open $M_t[u]$ under μ_t, (ii) provide a hash-valid suffix from $h_{u,t}$ back to $\bar{h}_u$, or to genesis

after cold start, and (iii) bind every event to u. A stale head that is merely present in the log cannot pass the current head-map proof.

Proposition 2 (sequential head correspondence). Assume collision resistance, receipt authenticity, faithful atomic construction of M_t from admitted A_t, a trusted choice of current ordinal t, and no fork. Every accepted requested head and event suffix resolves to the latest committed atom for that reference in A_t; replay over those atoms therefore satisfies Lemma 3 under the same closed-IR premises.

Proof sketch. Merkle binding fixes $h_{u,t}$ at the requested head-map position. Induction over the predecessor hashes either reaches the receiver's verified cached head or genesis; changing, omitting, reordering, or splicing a suffix changes an event hash or fails to reach that anchor. The trusted atomic issuance premise relates the resulting per-key heads to one checkpoint A_t. The proposition does not supply that premise: multiwriter serialization, fork resolution, durable commit, cache eviction, checkpoint truncation, garbage collection, and rollback/freshness are outside the implemented model and fail closed or require an external policy. Thus the event result is a warm-cache sequential backend comparison, not a concurrent-store theorem. Component tests cover current-head replacement, omitted suffixes, wrong cache anchors, and deletion tombstones; the 216 measured handoffs exercise successful sequential replay.

G Expanded Mechanism Comparison

Independent authenticated-computation baseline. Gregersen et al. (2025) mechanize security and correctness for Authentikit, including proof reuse and linking optimized operations with generic code. These mechanisms are prior work, not PBCE contributions. We compiled and ran their unchanged OCaml examples at commit `24c41ec37e5f12f75a40b00c65b787dea63810b0`; our environment uses OCaml 4.13.1 rather than their recommended 4.14.2, and we did not rebuild their Rocq proofs.

In a separate development-profile comparison, 72 public AgentDojo action replay states yield 288 predeclared existing-field query batches: one, four, sixteen, or all scalar leaves per state. These are designed membership queries, not 288 independent Agent tasks or version-skew decisions. Each of three author-library variants, a typed content-DAG prototype, and a full snapshot returns the correct values on all batches. Authentikit uses its own proof machinery over path-labelled binary Merkle leaves; our adapter checks the returned path before releasing a value. Faithful flattening and trusted roots are assumed; absence, object-shape equivalence, and restricted evidence authorization are outside this component comparison.

Compressed response (median B)	One	Four	Sixteen	All
Author naive	561.5	1638	4285	6704.5
Author buffered	561.5	1629	4217	6377
Author buffered/optimized	351	1079	3060	6482.5
Typed DAG prototype	874	2264	4734.5	6605
Full snapshot	1579.5	1579.5	1579.5	1579.5

All response sizes include their framing, zlib-6 compression, and 32 root bytes; common query paths are excluded. Caches are cold across batches. The author's optimized variant is smaller than the typed DAG in every one/four/sixteen-field batch; dense access favors full snapshots in median paired size. Thus selective reads alone do not establish an efficiency advantage over independent ADS implementations. The comparison does not establish equivalence of the full lifecycle protocols. An earlier RestrictedPython control shared our DAG and equality code, so its equal proof sizes support interpreter substitution, not independent prior-system reproduction. Neither component comparison upgrades the primary claims.

Complete consumers under a linked development profile. A subsequent implementation experiment extends the bounded record language with source-bound helper calls, literal receiver constants, explicit default and keyword binding, numeric multiplication, and string-prefix tests. Calls share a fuel budget and a depth bound of 32; unsupported transitive bodies are rejected before execution.

This is application-language support, not a new authenticated-data-structure primitive or an extension of the earlier IR theorem by assumption. The same 72 previously observed states remain the development denominator: supported states increase from 37 to 49 (26 to 32 task identities), with no loss of previously supported states. All 49 results match native utility decisions. On the existing 2697 cross-state stress pairs, 1211 execute correctly (500 True, 711 False), compared with 1007 previously; fully supported task identities increase from 41/97 to 51/97. These are not fresh-task or independent population observations.

We then execute the complete admitted utility programs through the author's unchanged naive, buffered, and optimized proof backends, our typed DAG, and full snapshots. The author adapter authenticates complete typed node facts, including container shapes, and uses ordinary short-circuit structural equality. All five arms support the same 49/72 states with 49/49 native matches; the 23 unsupported states are retained. The independent backend therefore confirms neither a PACS-specific coverage advantage nor an accuracy advantage.

Comparator	Warm paired ratio	Cold paired ratio	DAG smaller
Author buffered/optimized	0.8682	0.9243	35/49
Full snapshot	0.8094	0.8834	36/49

Entries are medians of per-state DAG/comparator byte ratios, not ratios of marginal medians. Warm accounting includes zlib-6 response bytes and 64 bytes for archive/program digests; cold accounting additionally transmits the compressed source package common to all arms. The 13.2% warm reduction against the author optimized variant becomes 7.6% when the source package is included. The author's response-only accounting is optimistic: its source-collected logical query list is excluded and recorded separately. These are application-byte measurements, not end-to-end latency, network, issuance, or deployment results. A different author adapter could improve costs further. Together with the scalar-query negative result above, this supports workload-dependent representation tradeoffs, not general superiority over Authentikit. Native utility matching is finite differential evidence; faithful package export and refinement to the native consumer remain separate obligations. General callbacks, SDK methods, and several other language constructs remain unsupported.

Authenticated access is not a new privacy problem. Xu et al. (2018) study authentication of relational queries with fine-grained access control, using access-policy-preserving signatures and zero-knowledge techniques to avoid exposing inaccessible information. Reijsbergen et al. (2023) combine integrity, privacy, and verifiable data services. These precedents rule out novelty claims based solely on recognizing that verification evidence can exceed a user's data permissions. PBCE instead measures explicit openings and abstains when their closure is ungranted; it does not hide those facts using zero knowledge. Its narrower research question is how a later consumer's paired historical/current evaluation can be bound to an Agent artifact lifecycle. Establishing that this specialization supplies additional task-level value requires evidence beyond authorization regression tests; we do not claim superiority over these privacy-preserving systems.

Capability systems reduce ambient authority and replay systems preserve prior machine executions (Watson et al., 2010; Dunlap et al., 2002). PBCE is not an OS confinement replacement; it removes the historical SDK and later executes reviewed hash-bound consumer code over authenticated materialized facts. Semantic remote attestation can attest high-level VM behavior (Haldar et al., 2004); PBCE instead binds the compatibility predicate and consumer at handoff and does not attest the remote platform. Selective- disclosure credentials and Merkle commitments are underlying data-plane mechanisms, not contributions (Fett et al., 2025; Merkle, 1988).

In a separate 20-artifact credential microbenchmark (not the Vapi workload in Table 14), RFC 9901 SD-JWT over 157 fixed disclosure policies verifies all 157 exact presentations and rejects all 157 altered disclosures. Its median presentation is 2.25× the PPF2 proof and 2.49× the signed full snapshot in this small-object workload. This is not a general SD-JWT efficiency claim: claim layout, batching, and workload differ, and SD-JWT remains a viable backend when its claim model matches the consumer.

General verifiable computation could evaluate P over committed private source state, potentially avoiding disclosure of the opened path, but moves trust and cost into circuit translation and proof generation (Parno et al., 2016). A TEE can retain or replay the old runtime but preserves a historical

execution environment and adds hardware/attestation trust (Haldar et al., 2004). Vector commitments or other authenticated dictionaries could replace the Merkle data plane. None alone specifies the source/artifact, late consumer, interpreter, and invocation bindings studied here.

Proof-carrying code makes a code consumer check a safety proof, while proof-carrying data attaches recursively checkable properties to distributed messages (Necula, 1997; Chiesa and Tromer, 2010). PBCE is narrower: its theorem is relative to a closed consumer IR and authenticated source observations. It does not certify arbitrary producer computation, consumer safety, or a general inductive invariant. The carried record proves which source result the admitted consumer obtained and binds that result to the later handoff context. Agent-action attestations bind an intended action, authority, or execution record (Wang, 2026; Rhodes and Kang, 2026); they do not reconstruct historical state for a consumer chosen after issuance or compare that consumer on two runtime views.

Program slicing removes code irrelevant to a preselected behavior criterion (Weiser, 1981). PBCE instead retains the complete source-side information at issuance and discovers an evidence slice when the later consumer executes. This difference enables a consumer unknown at issuance, but also requires a retained archive and reveals its access pattern.

Portable Agent Memory uses a content-addressed Merkle-DAG, access control, and rehydration to move structured memory across heterogeneous agents (Ravindran, 2026). Our object is a consumer-relative compatibility certificate rather than transferred memory: source and receiver results of one authorized consumer are compared. Certificate Transparency authenticates an append-only history and consistency between log heads (Laurie et al., 2021); it does not by itself prove that one selectively disclosed event is the latest event for a semantic key. E9 therefore adds a signed authenticated head map and shows that this event representation can outperform the fact archive.

Table 2: Protocol increment over the closest mechanisms. "Late P" means the consumer need not be known when the artifact is created; "app." means an application can build the property but the primitive does not define it.

Mechanism	Late P	Paired semantic statement	Cumulative receiver cap
Replay historical runtime	yes	application	no
Signed complete snapshot	yes	application	no
Signed static/trace projection	limited	application	application
SD-JWT claim bundle	claimized	no	application
VC/SNARK execution	circuit-bound	application	application
TEE historical replay	yes	application	policy
Generic authenticated query system	yes	application	application
This work's lifecycle	yes	defined	enforced

H Trust and Disclosure Boundary

The origin adapter observes the source runtime and full native object, validates the admitted value, and signs either a snapshot or archive root. The archive/ prover stores the complete fact archive and observes every authorized query; it can deny service but cannot make an altered fact pass the independent opening verifier without breaking the commitment assumptions. The receiver sees the public receipt, opened facts, parent shapes, access transcript, current object, and decision record. The hash-authorized consumer author and registry/operator are trusted to bind the intended P, z, keys, revocation, and runtime identities. The certificate is a local admission record for this trusted receiver boundary, not remote attestation of an untrusted receiver. A third party must trust the receiver, independently replay the bound current view, or compose an orthogonal TEE/verifiable-computation mechanism.

More explicitly, let V_o contain the native source and issued archive, V_a the full archive plus every authorized (P, q_s), and $V_r^{(i)}$ the receipt, current object, and openings returned to receiver i. A coalition learns $\bigcup_i V_r^{(i)}$; repeated authorizations can therefore approach the full snapshot. Authorization binds the exact consumer hash and budgets, but budgets alone are not a sensitivity policy: a deployment must additionally review P or allow-list readable path prefixes. The prototype provides no path hiding, coalition privacy, or confidentiality from origin/archive. Historical-key acceptance is registry policy: authorization expiry is checked, but distinguishing a signature made before versus

after an undisclosed key compromise requires an external timestamp/transparency and revocation policy.

The intended bounded-disclosure deployment separates an internal archive from a less-trusted compatibility auditor or cross-tenant receiver. The receiver is authorized to evaluate a particular resumption predicate but is not authorized to receive unrelated conversation history, credentials, or tool outputs. A signed encrypted snapshot remains preferable when that receiver is entitled to the complete plaintext. PBCE does not protect data from the origin or archive, and repeated authorized queries compose by set union; path allow-lists and a cumulative disclosure ledger are therefore explicit deployment mechanisms, not consequences of the commitment scheme.

"Late-bound" means that the serialized consumer may be chosen after issue but must validate under an authorized interpreter profile. Because issue-time evidence commits the full canonical source tree, adding an observation opcode does not itself require artifact reissuance. The receiver must possess an independent implementation of the new profile, and authorization binds its hash. A security upgrade revokes the vulnerable profile; requests under it become UNKNOWN rather than being reinterpreted. If an upgrade changes canonical encoding or source materialization, old receipts remain meaningful only under their historical profile and cannot be silently migrated.

Proposition (consumer-profile agility). Let a valid receipt commit a complete admitted source view independently of P and interpreter profile K. For any later (P, K) that validates under the authorized bounds, the same receipt can instantiate Theorem 4 without reissuance. This follows because receipt/opening verification depends on the committed view, while the late authorization and decision record bind $H(P), H(K)$ for that invocation. The proposition does not admit semantics outside the canonical source domain: changing encoding or materialization is a new historical profile, not a retroactive reinterpretation of the receipt.

Table 3: Actor visibility and trust. "Full" means no confidentiality from that actor.

Actor	Source visibility	Trusted for	Not provided
Origin attestor	Full native object	faithful materialization/signing	truth of external world
Archive/prover	Full fact archive	availability only	archive confidentiality
Receiver	openings, shapes, trace	verifier/runtime execution	hidden access pattern
Consumer author	declared P, z	program intent/determinism	hostile-code safety
Registry/operator	identities, keys	freshness/revocation policy	attestor honesty

Table 4: Theorem-to-deployment assurance ledger. Runtime checks and semantic assumptions are separate; checking a bridge does not establish materializer fidelity.

Layer	Evidence/assumption	What it does not establish
Archive binding	Signature, domain-separated multiproof, independent verifier	Truth of attested source object
Closed-IR replay	Hash-bound P, z, exact source trace/result replay	Repository-program equivalence
AutoGen profile	Checked bridge, complete-tree reconstruction, bound local predicate evaluator	Stateful horizons, arbitrary Python or projection fidelity
LangGraph v2 profile	Checked source bridge and complete-input reconstruction	Native projection/conversion fidelity or arbitrary continuation
Authorization	Signed program/capability, actual opening scope and cumulative ledger	Privacy against the archive or forked ledger signer
Materialization	Trusted source/receiver adapters, projections and runtimes	Fidelity under an undetected faulty adapter/loader
Repository frontend	294/294 differential sides and 24/24 detected mutants	Universal refinement $\mathsf{Bridge}(C, P, D)$

Table 5: Implementation independence in the principal checks.

Check	Shared component	Independent component
320 native/exact backends	Canonical source encoder and closed P	Native repository execution oracle
2,010 semantic cases	Test values and admitted semantics	Separately implemented eager/lazy evaluator
PPF2 proof/decision	Wire vectors and public keys	Receiver parser, materializer, proof and closure checks
Cross-language confirmation	Serialized receipt/proof vectors only	Node.js receiver; no Python/repository imports

I Detailed Experimental Sequencing

I.1 Timestamped objects

Only SHA-256 message imprints were sent to timestamp services; no artifact, schema, private archive, or consumer output was uploaded.

Table 6: RFC 3161 timestamps issued before corresponding outcomes.

Object	SHA-256	UTC issuance
64 source-root manifest	`2f1cf335fc43f5dc...10af7289`	2026-08-31 16:24:06
Stage-B protocol v2	`3cc2d9d99abef82b...44d8aa9`	2026-08-31 16:24:06
Compound omission v2	`1945c07b97a6f3e4...110150b`	2026-08-31 16:24:04

Offline verification uses the archived TSA leaf and CA certificates. Matching OpenTimestamps files were submitted to four public calendars and may be upgraded after Bitcoin inclusion. The RFC 3161 tokens are the immediately verifiable sequencing evidence. They prove that these bytes existed by the stated time. They do not prove investigator blindness: related atomic outcomes and native transfer outputs predated the tokens. We use them only as evidence of manifest/implementation freeze before the named PBCE executions.

I.2 Invalidated batches

Stage-B v1 is excluded because the implementation scored compatibility-decision agreement where the manifest required exact side-level output agreement. The corrected code and protocol were re-frozen and timestamped before a full rerun. Compound v1 is excluded because a CLI scientific-status parsing fix changed the frozen PBCE implementation hash before any compound outcome was evaluated; v2 was then frozen and timestamped. No partial batch contributes results.

J Per-Stratum Fixed-Set Results

Table 7: Exact outcomes on the fixed transfer set. Shared versions and mechanisms preclude an ecosystem-i.i.d. interpretation.

Stratum	Exact / n	Breaking
AutoGen	42/42	22
Plaid	12/12	2
Datadog	16/16	4
Overall	70/70	28

The observed cluster-level false-safe count is 0/28. AutoGen's 126 raw rows are 42 edges repeated over three seeds and are never treated as 126 independent observations. The 30 non- adjacent directions were prospectively frozen before receiver execution; the 12 adjacent directions were prior evidence. Opposite directions, releases, and failure mechanisms are dependent, so we report finite counts without an i.i.d. population interval.

Fresh task-continuity gate. The separate manifest fixes all 42 directed AutoGen release pairs and three fresh seeds before execution. The endpoint is the exact deterministic workflow transcript: user input, planner reply, and executor reply. Its nine clauses are the Cartesian product of those three positions with the framework's three typed message observables (source, content, type), rather than fields selected from version differences or complete-tree disagreement. The manifest records this consumer and its hash before the fresh source/receiver stage; it is not an externally timestamped pre-registration. At the directed-pair reporting level, 20/42 are breaking and 22/42 compatible; PBCE matches all 42 labels and all 22 compatible pairs execute the exact continuation oracle. The median opening is 9/414 (2.17%) feature leaves. Complete-tree equality labels two of those 22 task-compatible pairs breaking because of state representation drift. The 126 seed rows are repeatability checks, not independent samples.

Closest authenticated-chunk baseline. The E5 v1 harness produced zero comparison rows because it applied `len()` to an integer byte count and is invalidated in full. Fresh v2 evaluates all 324 cells in the predeclared 3 chunk sizes $\times$ 3 codecs grid. On the nine 100-event task-consumer rows, median chunk/one-fact-PBCE ratios for 4/16/64-fact Brotli chunks are 0.909, 0.998, and 1.108; their wins are 8/9, 5/9, and 4/9. The four-fact layout was then frozen before five new seeds.

Table 8: Fresh four-fact Brotli confirmation on 100-event official checkpoints.

Framework	Rows	Wins vs one-fact	Median blocked/one-fact
AutoGen	15	10	0.882
CrewAI	15	15	0.909
LangGraph	15	15	0.912
Overall	45	40	0.909

All 45 task oracles and both authenticated verifiers pass; blocked PBCE beats the compressed snapshot 45/45 times (median ratio 0.0199). Compression trades proof bytes for surplus disclosure: median disclosed/requested fact multipliers are 2.67 for next action, 4.0 for final result, and 3.4 for resumed trace. The corresponding median blocked/one-fact ratios are 0.951, 0.909, and 0.840. This is an empirical granularity frontier, not an optimality or privacy result. AutoGen's 10/15 win count, versus 15/15 for the other frameworks, is entirely its five next-action rows: those request only two facts, so four-fact packing adds surplus disclosure before it amortizes enough proof framing. Final-result and resumed-trace rows are 5/5 wins in every framework.

Table 9: Median disclosure accounting for the 45-row blocked-archive study. Package bytes exclude the 1,064-byte signed receipt; handoff bytes include it.

Consumer	Req.	Opened	Value B	Shape B	Package B	Handoff B
Next action	3	8	368	0	705	1,780
Final result	2	8	877	0	1,039	2,100
Resumed trace	20	68	4,746	320	5,010	6,085

Cached incremental-snapshot baselines. E7-v1 produced no complete comparison rows: restored CrewAI/LangGraph virtual environments referenced deleted temporary Python installations, and the blocked-verifier harness read a nonexistent diagnostic key after successful verification. The batch, protocol/runner hashes, and raw summary are retained and invalidated in full. E7-v2 uses three new seeds and reconstructs all 288 fixed/content-defined incremental snapshots exactly. Its best predeclared configuration (16-KB content-defined Brotli chunks) is 39.752$\times$ blocked PBCE in median cumulative bytes and loses all 27 comparisons.

E8 removes E7's complete ordered-manifest retransmission. A signed constant- size root addresses a Merkle DAG; the ideal synchronized receiver cache obtains only unseen compressed leaves and internal nodes. All 576 reconstructions and 108 blocked replays verify over 36 new official checkpoints. Table 10 reports the best of the complete predeclared 16- configuration grid; the full grid remains in the artifact.

Table 10: Fresh four-handoff comparison with the best content-addressed Merkle- DAG snapshot (4-KB CDC, Brotli-11, fanout 64). Ratios exceed one when blocked PBCE transmits fewer cumulative bytes.

Stratum	n	Median DAG/PBCE	PBCE wins
Next action	9	22.264	9
Final result	9	19.581	9
Resumed trace	9	6.777	9
AutoGen	9	20.104	9
CrewAI	9	18.763	9
LangGraph	9	18.364	9
Overall	27	19.581	27

Closest selective event-log baseline. E9 addresses a stronger alternative: append immutable change events and fetch only those relevant to the late consumer. A bare transparency-log membership proof is insufficient because it cannot prove that an old event is still the latest event for its key. The frozen baseline therefore signs a Merkle map from every path–feature key to its latest hash-chained event. Under an ideal persistent cache, the receiver verifies the requested head-map multiproof and the chain since its cached head. All 36 fresh official checkpoints, 216 log handoffs, and 108 paired blocked-PBCE replays verify exactly.

Table 11: Fresh four-handoff comparison with the Brotli authenticated keyed event log. Ratios below one favor the event log; min–max is over the nine framework–seed or consumer–seed units in each row.

Stratum	n	Median log/PBCE	Min	Max	PBCE wins
Next action	9	0.736	0.609	0.738	0
Final result	9	0.736	0.718	0.738	0
Resumed trace	9	1.093	0.973	1.096	6
AutoGen	9	0.718	0.609	0.974	0
CrewAI	9	0.737	0.735	1.096	3
LangGraph	9	0.738	0.736	1.094	3
Overall	27	0.737	0.609	1.096	6

Across individual Brotli handoffs, median components are 388 receipt bytes, 181 authenticated head-entry bytes, 494 proof-hash bytes, 559 compressed event bytes, and 21 framing bytes. This result rejects representation-specific dominance: a keyed event log is another selective PBCE-class backend, not a non-selective comparator. Its ideal-cache assumptions exclude negotiation, durability, garbage collection, concurrency, and network latency. With a cold or evicted cache it must transmit each requested key's chain back to genesis (or a trusted checkpoint), so Table 11 is not a cold-start bound. Concurrent writers additionally require a serialized or fork-resolving head policy, which the prototype does not implement.

Durable local-store boundary. A post-review protocol places the keyed event store, current heads, and signed checkpoint roots in SQLite with WAL journaling, full synchronous commits, and foreign-key checks. Nine fresh official checkpoints span AutoGen, CrewAI, and LangGraph at 3/30/100 events. Before each commit, closing a transaction after event insertion, head replacement, or checkpoint insertion preserves the prior logical database in all 27/27 trials. Six readers that establish a snapshot before an overlapping writer observe the complete prior authenticated root; fresh readers observe the complete new root. All 54 cold/warm selective packages verify. Median commit time is 19.44 ms; median local query time is 64.58 ms cold and 59.19 ms warm, while median package bytes fall from 2,092 to 1,517. The final database is 27.7 MB for 52,856 immutable events and 52,722 current heads. Query latency includes reconstructing the complete current head map to build a proof, exposing a concrete indexing bottleneck rather than claiming production readiness. These checks support transactional behavior of this named local implementation only; they do not cover multiwriter ordering, distributed consensus, Byzantine storage, remote latency, rollback, or GC.

The first authenticated-index batch incorrectly reconstructed historical heads by pooling globally content-addressed events across framework sequences; its self-consistent packages did not match the previously committed roots, so the frozen root-equality gate failed and the batch is retained as invalid. The corrected frontend starts from each sequence's committed final heads and walks per-key predecessor links to the target ordinal. It materializes leaf indices and authenticated internal nodes for all nine checkpoints. Every materialized root now equals the E10 committed root and all 54 cold/warm packages verify without scanning the head table. Median local proof-query latency is 2.57 ms cold and 1.00 ms warm (16 metadata reads; 4/2 event reads), versus 64.58/59.19 ms when rebuilding the head map. The tradeoff is 43.1 MB, or 1.55× the durable store, and 192.37 ms median index construction per checkpoint. This is a conventional authenticated-index optimization, not a new cryptographic primitive.

Model-generated request gate. The frozen manifest selects 12 source-schema request types over three immutable Vapi SDK revision edges and evaluates both directions (24 cells). DeepSeek V4 Flash receives the source schema but no receiver schema, diff, compatibility label or retry feedback. It emits 24 artifacts using 35,578 prompt and 6,381 completion tokens; 23 parse

and 21 execute repeatably under both endpoint SDKs. The direct wire oracle labels six breaking and 15 compatible directions. Receiver-only acceptance is correct on 16/21, while the separately archived source/receiver execution certificate agrees on 21/21. We use only the former comparison to establish that model-generated artifacts instantiate the problem; neither this certificate nor the two-runtime oracle is presented as PBCE/IR evidence. The gate manifest hash is `f6d9425a74bae0353ee8a24b5ef0870011083ce604ad0b58a1269da26fd14e5e`.

The separate downstream-action manifest is frozen after that localization but before any Qwen generation. It fixes four operational tasks for each of five mechanism classes and hashes the deterministic consumer implementation. The consumer ignores non-contract wire aliases and applies frozen defaults, as a tolerant JSON endpoint would. Qwen sees the operational task and public source schema, but no receiver schema, diff, consumer implementation, or outcome. All 20 outputs parse and execute under the source runtime; all 20 are wire-breaking under three-repeat official-SDK replay; all 20 change the frozen operational trace; and all 20 certificates verify independently.

Table 12: Targeted downstream operational-consequence stress. Selection is mechanism-targeted and therefore does not estimate natural prevalence.

Operational family	Source-executable	Wire-breaking	Trace changed
Twilio authentication mode	4/4	4/4	4/4
Supabase fallback routing	4/4	4/4	4/4
Output-tool dispatch/timeout/retry	4/4	4/4	4/4
Hume voice-cache policy	4/4	4/4	4/4
OpenAI voice-cache policy	4/4	4/4	4/4
Total	20/20	20/20	20/20

The precommitted gate required at least 18 parsed outputs, 16 source-executable artifacts, ten wire-breaking cases, impact in at least four families, at least 80% operational impact conditional on wire breakage, and verified certificates for every eligible case; every condition passes. The manifest hash is `70c1890b1660585f446001ff0340086e303db71752196ef67c213e82d86f125c`.

Compound-projection baselines. All policies were frozen in the predecessor manifest before compound outcomes. *Manual* contains ten author-selected source path families with existence/type/shape features. *Single trace* is the exact read set from one pinned assistant fixture. *Branch union* unions native read traces over all eight pinned fixture families and is the dynamic-coverage baseline. *Conservative full facts* contains every source fact appearing in those fixtures. *Consumer-output union* contains every fact in the canonical consumer outputs and is an issue-time oracle: it requires executing the already known consumer and therefore cannot serve late authorization. The latter two are boundary references, not deliberately weak competitors. No claim is made that the three deployable projections exhaust static or dynamic slicing.

Table 13: Compound errors by template for branch-union traces.

Template	Breaking / 40	False-safe	Over-quarantine
assistant	31	0	9
credential nested	25	0	15
generic	26	1	13
handoff tool	29	0	11
simulation	33	3	7
simulation suite	31	2	9
squad	31	0	9
structured output	34	6	6

The 91 errors have two distinct causes. All 79 over-quarantines arise because the static union includes a changed feature from a branch not taken in the paired execution. The 12 false-safes arise when two changes jointly activate a dependency absent from either atomic trace (six structured-output, three simulation, two simulation-suite, and one generic case). The exact consumer itself takes different source/receiver query sets in 201/320 pairs; the challenge therefore exercises data-dependent reads rather than only a frozen path list.

K Direct Baseline, Scale, and Admission Details

Direct baseline. Median results are in Table 14; times are local prototype costs rather than network latency. On the same 320 cases and 20 repetitions, full-snapshot issue and receiver execution

Table 14: Direct authenticated-backend comparison on the same 320 cases (20 repetitions; medians).

Backend	Issue ms	Prover ms	Receiver ms	Handoff B	Private B
Signed full snapshot	0.036	–	0.068	1,829	1,230
Compact one-fact PBCE	0.378	0.952	0.270	21,050	25,145

p95 are 0.044 and 0.079 ms. PBCE issue, prover trace, and receiver compact-verify/replay p95 are 0.432, 1.068, and 0.314 ms. P95 handoff sizes are 1,994 bytes for the signed snapshot, 23,069 bytes for the compact multiproof form, and 116,530 bytes for the uncompressed repeated-path form. Compact replay reproduces native output and the ordered prover query trace on 320/320; both decisions remain 320/320 correct. The private serialized PBCE archive is a median 25,145 bytes versus a 1,230-byte canonical source payload. A fresh descriptive rerun leaves all decisions unchanged and decomposes the compact handoff into per-case medians of 480 receipt bytes, 18,775 authenticated atom bytes, 1,786 multiproof-node bytes, and 12 framing bytes (component medians need not sum to the median total). The dense-read penalty is therefore mainly repeated typed atom payload, not signatures or proof hashes.

All local timing experiments use one Apple M4 Pro host (14 cores, 24 GB RAM), macOS 15.7.5, and Python 3.14.5; they exclude common authorization-service and sandbox startup costs. A post-review deployment envelope combines the paired wire sizes of 120 official AutoGen/CrewAI checkpoint cells with the measured local medians above and the measured 1.00/2.57 ms warm/cold authenticated-index lookup. It assumes one request–response RTT for both methods, charges zero snapshot storage latency, and charges lookup latency only to PBCE.

Table 15: Conservative handoff-time envelope for the 30 long-history cells. Ratios are median PBCE/snapshot completion time; wins count paired cells.

Cold profile	Network	PBCE wins	Median ratio
Datacenter	1 ms, 1 Gbps	0/30	2.536
Regional WAN	40 ms, 100 Mbps	30/30	0.915
Constrained WAN	100 ms, 10 Mbps	30/30	0.594

Across all 120 cells, cold PBCE wins 0, 40, and 75 under the same profiles. The calculation is a measured-input scenario analysis rather than an emulated service or production latency claim; its purpose is to show that byte savings do not imply latency savings on a fast local network.

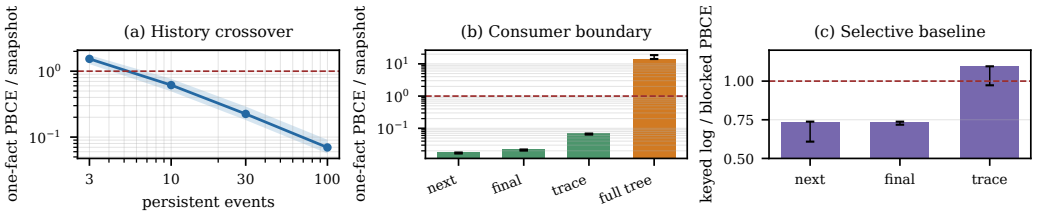

Figure 3: Byte ratios (red dash: equality). (a) Median and min–max over six predeclared framework–team coordinate medians (five seeds each); (b) median and range over three framework carriers at 100 events; (c) median and range over nine framework–seed units for four cumulative handoffs.

Scale protocol. The 30-cell grid contains six object sizes and five access densities. PBCE is smaller in 17 raw cells and full snapshot in 13; after symmetric gzip-9 or Brotli-11 it is smaller in 12 and full snapshot in 18. At 16,384 fields, gzip PBCE/snapshot ratios are 0.026, 0.189, 1.229, 4.425, and 5.122 across increasing read densities (Brotli: 0.027, 0.198, 1.280, 4.537, 5.160). Full-snapshot

issue medians range from 9.6–9.9 ms and execution from 5.8–8.2 ms. PBCE archive issue medians range from 148–153 ms; proof construction plus verification ranges from 5.8 ms at 16 reads to 3.12 s at 16,384 reads. These figures include local serialization, hashing, signatures and proof checking but exclude transport, registry lookup, durable storage and production sandboxing.

Restricted source compiler, IR sensitivity, and admission funnel. The fail-closed frontend compiles the pinned repository consumer into 12 closed IR operations. It parses the canonicalization and credential-walker ASTs, requires every statement to match a supported structural pattern, extracts field/path parameters, and rejects unknown statements instead of ignoring them. Fourteen prespecified unsupported source mutations (deleted resolution branches, extra or expression-hidden calls, unknown helpers, and removed safety checks) are rejected 14/14; four supported semantic mutations (excluded, nested, rename, and nullish fields) compile 4/4 to a different IR. Validation admits it, rejects 8/8 malformed/unknown programs and all four depth/node/byte/missing-root trials, and IR-native execution agrees with repository TypeScript on 294/294 source/receiver sides of the 147-pair carrier. On the 320 compound carrier, IR source, receiver, PBCE-lazy output, and compatibility decision each agree on 320/320. These checks validate the compiler output empirically; certification thereafter uses the hash-bound IR itself. A separately implemented finite-domain validator binds hashes of the two repository sources, compiler, emitted IR, 147-pair manifest, and native-result corpus. It recomputes all 294 equalities and emits a validation record only when every check below passes; a changed source, IR, manifest, oracle domain, expected summary, or absent file returns UNKNOWN. Substituting a different admitted IR while updating its file hash is still detected by native-result disagreement. Deleting each concrete operation and mutating one admitted parameter per operation yields 24 mutants; all 24 are detected, with 8–294 changed side executions per mutant. This demonstrates that every emitted operation is causally exercised. The resulting TV_D record permits Corollary 5 only on this finite D; it is not mechanized semantic refinement of the source language.

To avoid making translation evidence depend on one repository or on field projection alone, a second frontend targets the pinned AutoGen 0.6.4 termination-condition source. It recognizes the exact control-flow shapes of `StopMessage`, `TextMention`, `Handoff`, `SourceMatch`, and `TextMessage` termination consumers and emits a separate closed predicate IR; unknown classes, source shapes, fields, or parameters are rejected. Twelve parameterizations cross 50 deterministic message batches, including empty, source-filtered, type-conflicting, substring, and handoff-target cases. A hash-bound validator replays the official package in its immutable environment and obtains 600/600 native–IR equalities. It then changes each program's predicate family and one family parameter: all 24 mutants disagree with the trusted native corpus, with at least 13 detecting executions per mutant. Source, interpreter, worker, manifest, and native rows are bound in the validation record; missing or altered evidence returns UNKNOWN. This broadens finite translation evidence to repository control flow, but remains TV_D, not universal Python refinement.

A third frozen bridge targets LangGraph 1.2.11's official `add_messages` state reducer. Its separate IR admits explicit-ID human, AI, system, and removal messages, including ordered replacement, deletion, append, and remove-all behavior; UUID assignment, chunks, general coercion, and output formatting are rejected. Five scenario families cross 80 cases each. The v1 corpus matched 400/400 native executions but could not detect an order-sorting mutant and was invalidated in full. Before v2 native outcomes, we froze shuffled non-lexicographic IDs and all six semantic mutants. V2 again matches 400/400 native executions and detects all six mutants in 80–320 cases each. The record binds the official source, protocol, worker, interpreter, manifest, and native rows; altered evidence yields UNKNOWN. This adds a third repository and state-reduction semantics, but remains finite translation validation rather than arbitrary LangGraph refinement.

The legacy syntax inventory is all 130 exported function declarations/arrow bindings in 45 pinned Vapi source files. The conservative closure scanner admits 33 (25.4%) and rejects 97. The target consumer is admitted through seven transitive functions. Frequent rejection causes are async control (46 functions), filesystem/external calls, time and network. The scanner may reject programs that could be safely expressed in a richer calculus; it is designed to avoid silent fallback. Separately, all 16 prespecified unsupported snippets and all 13 inadmissible values are rejected, while 10/10 valid domain examples pass.

Official checkpoint-domain audits. A protocol frozen before audit outcomes selects AutoGen, CrewAI, and LangGraph checkpoints at 11/25/75/120 events (12 prespecified units, no replacement). Official creation and two identical official replays pass on every unit; all 12 source-adapter artifact materializations and 12 replay observations lie in the canonical JSON domain, survive exact serialize/parse round-trip, and issue source receipts (4/4 per framework).

A subsequent protocol freezes six representative user-defined tool values (nested JSON, datetime, bytes, tuple, Pydantic model, and set) in each framework, with no replacement and no omnibus gate. Of 18 cells, 17 save, 11 restore through the tested durable representation, 11 persisted artifacts and 8 restored observations lie in $\mathcal{J}$, and 5 preserve original type and value. CrewAI's JSON provider normalizes several types and rejects bytes; LangGraph preserves datetime/bytes/set but those restored observations lie outside $\mathcal{J}$; raw AutoGen state contains its own datetime and is not directly JSON-domain even for the nested-JSON control. Thus the first audit measures the declared source adapter, not natural coverage of arbitrary checkpoint values. Two implementation-invalidated full batches are retained and excluded.

The cross-ecosystem funnel starts from 158 frozen raw candidates: 36 adjacent AutoGen, 90 non-adjacent AutoGen, and 32 Plaid/Datadog. Four held-out-provider rows fail source receipt issuance and are not replaced. All 154 issued rows compile to the closed observation IR, all 154 authenticated source transcripts pass exact closure checking, and all 154 receiver decisions agree with the previous native outcomes. The authorization is issued after the source receipt and verification performs zero source-runtime executions. Thus end-to-end issuance coverage is 154/158 (97.5%): every uncovered row is an explicit issuance failure, while no issued row returns UNKNOWN or aborts at receiver verification. The task-continuity carrier is fully covered (126/126 repeated executions, reported as 42 directed pairs); its scored set contains no source-runtime exception versus receiver domain-rejection case.

L LEGACY SOURCE-CAPSULE AUDIT

The Vapi consumer's commit is `4c3630212f6fa558b01b7be5c8af0249a713a505`. The capsule binds Node `v22.23.2`, V8 `12.4.254.21-node.56`, Darwin/arm64, disabled string/WASM code generation, a 250-ms timeout, deny-list, consumer source hash, oracle bootstrap hash, and independent-verifier hash.

This capsule predates the closed IR and is retained as engineering evidence, not as the certifying interface. Across 147 pairs, ten executions per side preserve the exact output and ordered fact/proof transcript. All capability probes listed in Section 5 are denied. Constructor escape is tested separately from ordinary dynamic-code generation. These results do not imply that a general Node VM is a security boundary against an arbitrary hostile program. The threat model is hash-pinned, reviewed repository code with accidental or unexpected ambient dependencies. Adversarial consumer code requires a process/container/WASM boundary not implemented here.

M EVIDENCE ROLES AND CLAIM BOUNDARY

Official-checkpoint and composed-verifier detail. At 100 events, median PBCE/snapshot ratios across AutoGen, CrewAI, and LangGraph are 0.018 for next action, 0.022 for final result, 0.069 for the complete resumed trace, and 14.162 for the complete checkpoint tree; each task-conditioned consumer wins in 3/3 frameworks and the full-tree consumer in 0/3. The 16-consumer cached-receipt ratios are 0.192–0.214. A separate 16,384-field accounting grid opens 1.0% of values for one 1% consumer and 16.0% for 16 disjoint consumers (8.5% when half the paths are shared); root key names and access patterns remain visible.

The frozen composed-verifier study includes four semantic controls and 40 faults spanning provenance, adapter/registry, proof, authorization, interpreter, receiver, resource budgets, and absence resolution. Its first run exposed a mismatch: producer execution applied whole-input node/depth budgets to sparse path observation while the independent receiver did not. That run is invalidated; after fixing the receiver and adding a regression test, the complete unchanged protocol passes 44/44, and no injected fault returns CERT-COMPATIBLE. Five absence cases cover multilevel missing objects, scalar descendants, and invalid array positions. This is implementation-level fault containment only.

The receiver-side local footprint is 1,174 nonblank, noncomment Python lines plus cryptographic and OS dependencies.

Table 16: What each study establishes.

Study	Role	Supported conclusion
320 compound cases	Fresh timestamped stress	Admitted replay removes sampled projection omissions; not prevalence
42 directed AutoGen task units (126 rows)	Fresh executable continuation	Sparse task consumer matches 42/42 pair-level outcomes; all 22 compatible pairs resume exactly; full-tree equality over-quarantines two
70 directed state units (158 candidates; 154 issued rows)	Mixed prior/prospective transfer	One late-authorized closed observation IR preserves all issued native decisions; four issuance failures retained; seeds are repeatability only
147 Vapi pairs	Engineering assurance	Adaptive lazy execution, confinement, transcript repeatability
Closed IR v2, 294 sides + 320 units	Admission/translation bridge	Finite-domain record revalidates 294/294 and 24/24 mutants; universal repository refinement is not proved
AutoGen predicate IR, 600 executions	Repository control-flow bridge	Five families match 600/600; checked derivations reject 72/72 substitutions and 6/6 semantic-source attacks
LangGraph reducer IR, 400 executions	Repository state bridge	Reducer matches 400/400; checked derivation rejects 5/5 substitutions and 7/7 semantic-source attacks
SQLite keyed log, 9 checkpoints	Durable local-store diagnostic	27/27 crash rollbacks, 6/6 reader–writer snapshots, 54/54 authenticated queries; local single-writer only
SQLite authenticated index, 9 checkpoints	Sparse proof-construction diagnostic	54/54 queries verify without full-head scans; 1.55× storage/build tradeoff; first root-reconstruction batch invalidated
320 full-snapshot comparison	Additional frozen direct baseline	Both match native; full is cheaper on small dense objects
2,010 semantic cases	Independent differential stress	Own-key order, -0, Unicode, and container observations match; not translation proof
30-cell cumulative disclosure	Information accounting	Value disclosure accumulates; key names/access patterns remain visible
30-cell size/locality grid	Deployability stress	Raw/gzip/Brotli crossover; not natural workload prevalence
130-function syntax audit	Legacy checker stress	Rejection causes in one repository; not certification or coverage
20 Qwen downstream-action units	Targeted prospective mechanism stress	Fixed operational traces change in five selected families; not prevalence or task utility
512 opening mutations	Regression testing	Independent verifier rejects specified corruptions; not cryptographic security evidence

N Late-Bound Policy Synthesis and Evidence Closure

Closed policy language and prospective design. The synthesis IR is a total boolean expression language with conjunction, disjunction, negation, equality/inequality, substring, existence, and bounded cross-path comparisons. Every observation names one canonical path and one of `exists`, `value`, or `shape`. The independently frozen 36-intent population contains 12 tasks per AutoGen, CrewAI, and LangGraph checkpoint schema and 484 hidden finite-domain states. Both models receive the same syntax, catalog, intent, and logical capability. Each one-shot candidate is copied before either repair treatment: blind refinement sees only its prior candidate, whereas CEGIS sees accumulated failing states with candidate and expected decisions. Neither arm sees the gold program. At most two repairs are permitted. The primary protocol and source hash are archived before calls.

Counterexamples improve seven cells and regress none versus one shot (descriptive one-sided exact paired sign test $p = .0078125$), and improve five with no regression versus blind refinement ($p = .03125$); blind self-refinement improves only two of the eight initial errors. Total usage is 155 calls, 74,856 prompt tokens, and 13,639 completion tokens. All admitted candidates obey logical path, feature, and atom budgets. The gate nevertheless requires absolute gains of at least 10 points over one shot and 8 points over blind refinement; observed gains are 9.72 and 6.94 points. It is therefore recorded as **FAIL**, not changed because one model is at 36/36 ceiling.

Evidence-closure-aware admission. The first end-to-end audit reveals that logical observation sets and proof requirements can differ. Four AutoGen intents test whether an optional `target` path exists. In the actual frozen checkpoint it is absent, so the proof opens its parent object's `shape`. The frozen logical capability allows `exists`/`value` but not `shape`; a surface-only authorization

Table 17: What each study establishes (continued): authorization, authoring, and profile composition.

Study	Role	Supported conclusion
27 official checkpoints; 171 admitted projections	Three frozen protocol confirmations	Existing proofs derive textual and artifact-relative witness-closed proofs without archive access; a separate Node receiver reproduces the final 57; uncovered targets abstain
432 disclosure-capability trials	Authorization/accounting stress	108 authorized requests accepted; 324 scope, budget, context, concurrency, and audit-tamper trials rejected; no coalition privacy or fork prevention
180 capability-integrated handoffs	End-to-end integration	Actual proof leaves are charged before replay; 36 valid handoffs certify and 144 capability or receiver-record substitutions fail closed
72 official-checkpoint capability trials	Deployment integration	24 authorized first/repeated handoffs certify; 36 scope/feature/budget violations and 12 receiver-view substitutions fail closed; median opening 0.191%
36-intent late policy synthesis (72 model–intent cells)	Capability-constrained consumer authoring	Counterexamples repair 7/8 initial semantic errors versus 2/8 blindly; absolute-gain omnibus remains failed
72 synthesized-policy PBCE cells	Evidence-closure integration	64 capability-covered policies replay exactly; eight ungranted absence proofs abstain; 72/72 proof mutations abstain
18 fresh repository-derived policies (36 model–intent cells)	Prospective hybrid transfer	23/36 one-shot to 36/36 repaired; frozen gate passes; this alone does not isolate LLM value
18-task no-LLM comparison	Post-review descriptive ablation	All starts solve 18/18; counterexamples total 79 without a model, 37/25 with model starts; no final-accuracy advantage
600 authenticated predicate pairs	Retrospective native-record regression	Complete-tree adapter matches all retained pair results, including 274 breaking; no new external sample
22 workloads / 11 official classes	Single-module inventory diagnostic	22 objects, 10 expressible, 5 checked and correctly certified; grants removing shape or limiting one atom certify none; constructed messages
36 fresh synthesized-policy PBCE cells	Prospective evidence-closure integration	32 covered policies replay exactly; four ungranted absence witnesses abstain; zero false authorization; 36/36 proof mutations abstain
6 official-checkpoint canary cells	Controlled value exclusion	Full views contain the public canary, proofs/records exclude it, and all next-action decisions certify; not production privacy
Formal IR/bridge/game/theorem	Conditional reasoning	AutoGen and corrected LangGraph v2 compose with complete-input reconstruction; native projection/conversion remain premises

Table 18: Compound omission challenge. Complete snapshots and PBCE match decisions; the distinction between them concerns evidence transfer cost.

Method	Correct / 320	False-safe	Over-quarantine
Manual paths	117	203	0
Single trace	211	30	79
Branch-union traces	229	12	79
Conservative full facts	240	0	80
Consumer-output union	320	0	0
Signed full snapshot + closed-IR P	320	0	0
Compact PBCE + closed-IR P	320	0	0

would therefore approve evidence it did not grant. Those first-audit requests correctly return UNKNOWN and are not widened post hoc.

The post-diagnostic v2 protocol freezes the evidence-closure relation $\mathsf{EC}(P, x_s) \subseteq_{\downarrow} \Pi \times F$ and atom budget B as the admission rule. Across 72 final policies it authorizes 64/64 whose compiled closure is in scope and abstains on the eight parent-shape cases. All authorized PBCE replays produce exactly the source decision; all eight abstentions are required by the signed capability; there are zero false authorizations and zero incorrect abstentions. A byte mutation of every proof makes the independent verifier return UNKNOWN in 72/72 trials. Median/max authenticated opening fractions are 0.0754%/0.1519% of the official checkpoint feature tree.

Proposition (untrusted synthesis composes with selective replay). Let an arbitrary synthesizer emit P. Assume closed-IR validation accepts; compilation faithfully enumerates $\mathsf{EC}(P, x_s)$; the capability and nonforking ledger satisfy the confinement proposition in Appendix O; and PBCE

Table 19: Frozen 36-intent synthesis results. The omnibus remains failed even though paired repair endpoints are positive.

Arm	Exact / 72	DeepSeek / 36	Qwen / 36
One shot	64	28	36
Blind self-refinement	66	30	36
Counterexample-guided	71	35	36

satisfies Theorem 4. If evidence-closure admission authorizes, every delivered source atom is inside the signed prefix/feature set, the cumulative distinct-atom total remains at most B, and any returned compatibility certificate has the IR-relative meaning of Theorem 4. If closure exceeds capability, no evidence is delivered and the result is UNKNOWN.

The proof is composition: compilation fixes the complete set of terminal and first-blocking observations; set/budget admission establishes the capability premise before delivery; ledger serialization establishes the cumulative bound; and authenticated replay supplies the conditional semantic result. The claim is deliberately independent of whether P matches natural-language intent. A synthesis error can implement the wrong authorized policy, but cannot enlarge its evidence authority or forge a source result.

This establishes selective implementation correctness for the frozen policies, checkpoints, and capabilities. It does not establish unrestricted prose-to- policy equivalence, and deployment needs owner-authored tests/properties or another authoritative intent oracle. Generic LLM–verifier iteration is prior work (Wen et al., 2026; Vatsa et al., 2026); the protocol result is that authorization must apply to the artifact-dependent proof closure generated after compilation, not merely to a syntactic field list. Because v2 was designed after observing the v1 closure mismatch and reuses the same 72 policies, it validates the repaired composition but is not an independent replication of the closure phenomenon.

Prospective repository transfer after freezing bounded repair. The first repository-derived transfer set exposed two general hypothesis-class failures: redundant observations could exhaust evidence budgets, and unordered fan-out/join policies require nested Boolean structure. We therefore froze a repairer with unique-suffix path canonicalization, test-preserving pruning, a finite atomic library, flat conjunction/disjunction search, and a two-term bounded-DNF fallback. Equivalence queries reveal one failing state and its expected Boolean decision at a time; the repairer sees neither the gold program nor unrevealed labels. Paths and scalar constants come from the owner-declared finite catalog. Earlier repository-derived sets remain failed development experiments and are not reclassified.

We then froze three new upstream repositories and commits—OpenAI Agents SDK `f1f3503`, Google ADK `c7ef8cf`, and PydanticAI `26467f6`—before defining 18 new official-test-derived predicates and before observing any model output (OpenAI, 2026; Google, 2026; Pydantic, 2026). The set contains 186 hidden finite-domain states, six policies per repository, and zero gold-program overlap with either earlier repository-derived set or the original 36-intent population. The primary gate requires at least 90% exact overall, at least 80% per repository, zero regression and capability violation, and median repair within eight counterexamples.

Fresh repository-derived synthesis after freezing bounded repair.

Population	Cells	One shot	Hybrid final
OpenAI Agents	12	8	12
Google ADK	12	9	12
PydanticAI	12	6	12
Overall	36	23 (63.9%)	36 (100%)

The prospective gate passes: all 13 initial errors are repaired, none of 23 correct programs regress, and no accepted program exceeds its path, feature, or atom capability. DeepSeek improves from 10/18 to 18/18 and Qwen from 13/18 to 18/18. Initially wrong cells require median four and at most

eight revealed counterexamples. This is evidence for the hybrid within a finite declared domain, not for generic CEGIS novelty or arbitrary natural-language synthesis.

A separately frozen integration compiles logical existence queries into authenticated value openings or first-blocking parent-shape witnesses. It authorizes all 32 capability-covered policies, abstains on the four closures requiring ungranted shape evidence, and makes no false authorization. Every covered authenticated projection reproduces the source decision; all 36 single-byte proof mutations return UNKNOWN. Median/max opened fractions are 3.57%/7.14% of these compact carrier feature trees. The initial integration attempt is invalidated before outcome collection because it passed logical `exists` directly to an observation IR admitting only concrete `value`/`shape` facts; the corrected compiler and runner were re-frozen before this audit.

Post-review initialization ablation. We rerun the identical frozen repairer on the same 18 tasks with an empty initial candidate and with each saved model proposal. Candidate library, objective-based lexical ranking, finite domain, oracle ordering, capability, and the 24-counterexample budget are identical. This is a descriptive ablation on already observed tasks, not a new confirmatory gate. The terminal equivalence check is counted separately from returned counterexamples.

Initialization	Exact / 18	Counterexamples	Oracle checks
No LLM	18	79	97
Text-ranked conjunction	18	55	73
DeepSeek	18	37	55
Qwen	18	25	43

Representation-dependent authorization diagnostic. The four closure refusals above are not intrinsic to the logical grants. A post-review baseline builds a conventional 256-bit sparse authenticated map of the same complete feature population at issuance. Non-membership of an existence key proves absence without opening its parent's shape. On the same 36 saved cells, all decisions agree with original execution, and the existing signed capability and cumulative-disclosure ledger admits 36/36 under unchanged logical grants. All 108 wrong-receiver, expiry, and wrong-receipt checks reject. These are dependent implementation checks, not new tasks or a security theorem. The map completeness and fixture issuer remain trusted; a new root and bound capability are issued rather than retrofitting old receipts. A missing scalar value key alone is not taken as path absence: existing containers remain unsupported scalar values.

Among the 32 cells authorized by both formats, median paired sparse-map/PBCE proof-payload ratios are 1.233 uncompressed and 1.279 with identical zlib-9 compression. Sparse packets are smaller in 4/32 uncompressed comparisons. The baseline shares sibling digests but does not implement a minimal sparse multiproof. These counts exclude receipt, capability, request and audit transport, issuance, storage and networking; they do not establish end-to-end cost superiority. The local harness checks grants after constructing packets, not as an audited pre-disclosure server. Thus this diagnostic supports a format-dependent size/authorization trade-off, not novel cryptography, minimum necessary disclosure or production coverage. The original 32/36 and its gate are unchanged.

The no-LLM run is counted once: model arms are paired over the same 18 tasks, not 36 independent tasks. All arms reach exact finite-domain behavior. Model initialization reduces feedback on this population, but final accuracy does not establish its necessity. A stronger post-review control initializes from a text-ranked conjunction without inspecting expected labels. Its 73 checks improve on the empty start, but exceed the model starts' 55/43. Text-ranked single-atom and disjunction starts each require 97 checks; all reach 18/18. These are additional retrospective controls, not independent confirmation or a per-task oracle selector.

We further give all starts identical per-task memoization of pure program/state outputs and use lazy text initialization, preserving all previous oracle traces. The conjunction start executes 4,362 candidate/state evaluations, versus 2,268/1,680 for saved DeepSeek/Qwen starts. Execution counts do not determine latency: in 15 randomized timing blocks, the empty start is faster than the conjunction start (median cohort totals 188 versus 212 ms), despite executing more candidates. Saved model starts take 148/116 ms locally. These timings include seed construction, enumeration, cache-key construction, and terminal checks, but exclude generation and proposal parsing. Paired median local savings against the empty start are only 37/74 ms over 18 tasks. Historical generation used 18

calls per model and did not record API latency; therefore neither these timings nor the feedback reductions establish an end-to-end speedup. Timing repetitions are not additional tasks. The scripts and full traces preserve all tasks and all tested starts.

O Cumulative Receiver-Disclosure Capability

For deployments that authorize only a bounded semantic view, the prototype issues an Ed25519-signed capability

$$\gamma = (\mathsf{receiver}, \mathsf{purpose}, H(\mathsf{receipt}), \Pi, F, B, \mathsf{expiry}, \mathsf{capability_id}),$$

where Π is a canonical component-wise RFC 6901 prefix set, F is the set of permitted fact features, and B is a maximum number of distinct atoms over the capability lifetime. Prefix matching is performed on decoded components, not strings, so `/messages/1` does not authorize `/messages/10`; noncanonical escapes reject. A SQLite-WAL ledger uses `BEGIN IMMEDIATE` to atomically add newly disclosed atoms. Reopening an already charged atom is free, whereas concurrent requests cannot each spend the same remaining budget. Every accepted update extends a signed hash chain over the prior head, capability, opened set, sequence number, and cumulative total.

The enforced invariant is

$$\forall t: \quad U_t \subseteq \{(p, f) : p \in\downarrow \Pi,\ f \in F\}, \qquad \left|\bigcup_{i \le t} U_i\right| \le B,$$

for each capability and ledger view.

Proposition (capability confinement and cumulative accounting). Assume EUF-CMA security of the capability signature, collision resistance of the bound hashes, injective canonical path decoding, and a nonforking ledger whose transactions serialize updates for each capability. Every accepted opening is within its signed path/feature scope and the union of distinct atoms accepted under that capability never exceeds B. Furthermore, altering an entry in an externally anchored audit chain is detected.

Proof sketch. Signature verification and exact context comparison fix Π, F, B and the capability identity. Canonical component-wise prefix checking establishes $U_t \subseteq\downarrow \Pi \times F$ before any update. Induct on serialized ledger commits: the base union is empty; a commit computes $U_{\le t} = U_{<t} \cup U_t$ inside the same transaction and accepts only when $|U_{\le t}| \le B$. Thus concurrent attempts cannot both rely on one prior balance. Each accepted row signs its predecessor head and new state, so a modified anchored chain requires a signature forgery or hash collision. A malicious signer can still create two valid successors, hence the explicit nonforking/transparency premise.

A frozen component matrix spanning AutoGen, CrewAI, and LangGraph path shapes accepts all 108 authorized requests and rejects all 324 outside-prefix, feature-escalation, cumulative-budget, authorization-context, concurrent-overspend, and audit-tamper trials. A separate 180-trial end-to-end gate authorizes actual proof leaves before replay: 36/36 valid handoffs certify, 108/108 scope/feature/budget violations abstain, and 36/36 receiver-view substitutions invalidate the bound decision record. Its v1 output is retained but invalidated because the view field hashed the tagged envelope; v2 freezes the corrected separate outcome/view digests. The fresh E15 deployment gate composes the same verifier and ledger with checkpoints created and resumed through official AutoGen, CrewAI, and LangGraph persistence APIs. Its locally frozen manifest fixes two unseen seeds, 30/100 history rounds, four agents, and six trials for each of 12 checkpoint cells. All 24 authorized first/repeated handoffs certify; reopening the same atoms adds zero distinct-atom charge. All 36 path-prefix, feature, and cumulative-budget violations abstain, and all 12 exact receiver-view substitutions invalidate the decision record. The authorized task consumer opens a median 0.191% of the committed feature atoms (maximum 0.437%); median restored observation and proof sizes are 348,637 and 6,642 bytes. The former is the receiver-local admitted current view, not transmitted source evidence; the latter is the compact source proof payload. This is an executable delegated-auditor scenario, not evidence that production checkpoints contain sensitive data or that the origin/archive learns nothing. In a separately frozen E16 control, a public credential-like canary is embedded in

30/100-round tool outputs for the same three official workflows. All six complete restored views contain it; all six compact proofs and decision records exclude it while the next-action consumer remains certc. Median/max opening is 0.0253%/0.0569% of committed feature atoms. The canary is synthetic and public: this checks receiver-evidence value exclusion, not real-secret handling or a cryptographic privacy definition. This is an authorization and accountability result, not cumulative privacy: receivers may collude, paths and access patterns are visible, and a malicious ledger signer can fork views unless an external transparency service anchors heads.

P Checked Semantic Bridge for Late Consumers

Target-language boundary. Closed IR v2 contains only the transform and observation operations in Appendix C.1. The AutoGen predicate target and LangGraph message-reducer target below are separate profiles, not additional opcodes admitted by that validator. Their checkers establish source-to-target correspondence on the specified projections. Applying Theorem 4 additionally requires a verified lowering to the core or a separate archive-simulation argument for the target profile; a profile hash alone supplies neither. We initially reported these as semantic-bridge components. The complete-tree construction below now supplies an explicit input-simulation path for the AutoGen predicate profile. A subsequently corrected LangGraph v2 profile uses the same complete-input construction, as described below; the original v1 certificate is not silently upgraded.

The Boolean policy experiment likewise uses a separate evaluator. Its PBCE runner compiles observed path/features into Closed IR v2 `observe_paths`, verifies those openings, reconstructs a projected object, and executes the Boolean policy locally. The 32 covered decisions test projection sufficiency at the evaluated objects; they do not prove universal projection sufficiency for that language or complete workflow continuity. Such lifting requires both projection sufficiency and evaluator correspondence in addition to authenticated observations.

The repository-to-IR bridge is generally a semantic refinement obligation, not something signatures or authenticated openings can establish. We therefore separate the existing finite-domain validators from two proof-producing source fragments. The first covers fresh instances of five official AutoGen termination conditions over an admitted projection of finite JSON messages. The observable native result is whether the call returns a stop message; its human-readable formatting is outside the relation.

Let m range over admitted messages with type, source, textual content, and an optional handoff target. The checked predicate schemas are

$$\begin{aligned}
&\mathsf{Stop}(m) && \equiv \mathsf{type}(m) = \texttt{StopMessage},\\
&\mathsf{Mention}_{t,S}(m) && \equiv (S = \bot \vee \mathsf{source}(m) \in S) \wedge t \sqsubseteq \mathsf{text}(m),\\
&\mathsf{Handoff}_{u}(m) && \equiv \mathsf{type}(m) = \texttt{HandoffMessage} \wedge \mathsf{target}(m) = u,\\
&\mathsf{Source}_{S}(m) && \equiv \mathsf{source}(m) \in S,\\
&\mathsf{Text}_{u}(m) && \equiv \mathsf{type}(m) = \texttt{TextMessage} \wedge (u = \bot \vee \mathsf{source}(m) = u).
\end{aligned}$$

Each class rule requires exactly the published bases and class member envelope, a constructor whose complete body binds the displayed parameters to the exact instance attributes and initializes `_terminated` to false, and a complete `__call__` body of the form "already-terminated guard; iterate messages; on the family predicate set the flag and return a stop message; otherwise return none." The mention rule additionally admits its source filter and `to_text` assignment; the message projection premise requires that this call returns the admitted textual content. Extra methods, unknown statements, altered Boolean connectives, unrecognized effects, and constructor substitutions are outside the grammar.

The untrusted producer emits target P and certificate π_B containing the rule identifier, exact class/constructor/call AST digests, source spans, configuration, source digest, and target IR. The independent checker reparses the source, checks the complete rule schema and configuration domain, and reconstructs the unique target family. It imports neither the producer nor the legacy frontend.

Bridge-checker lemma. For a class C accepted under one of the five rules, a fresh instance with configuration θ, and any admitted message sequence M,

$$V_B(C, P, \pi_B) = 1 \quad \Longrightarrow \quad \mathsf{terminated}(C_\theta(M)) = \mathsf{terminated}(\mathsf{Eval}(P_\theta, M)).$$

For each production, the constructor check gives equality of the parameters read by the predicate. The initial guard is false for a fresh instance. Loop induction then shows that source and IR select the same first matching message; both return false if none matches and true otherwise. State assignment and return checks establish the source observable in the matching case. The five predicate equations above discharge the family cases. This is a paper proof of the checked rules, not a mechanized proof of Python.

E17-v1 is retained but invalidated because it missed an additional class-level `__getattribute__`; v2 closed that route but was then invalidated because it missed module-level rebinding of a predicate class. The fresh v3 checker verifies relevant imports and unique module bindings in addition to the class envelope. On pinned AutoGen 0.6.4 it checks all 12 configured instances (five families); 72/72 recomputed-digest source/rule/AST/span/configuration/target substitutions and 6/6 semantic source mutations accepted by the legacy frontend are rejected. Median certificate size is 1,018 bytes and median checking time over 101 checks per instance is 4.24 ms. A retrospective release audit spans eight package versions and four distinct source files: 32/40 family–release cells verify, while eight older multimodal, absent, or import- binding-incompatible schemas abstain. AutoGen 0.7.5 was acquired after the checker freeze but its relevant source is byte-identical to earlier releases, so it is not counted as an independent source replication.

E19 applies the same trust separation to LangGraph 1.2.11's official `add_messages` reducer. Its independent checker recognizes the complete two-argument wrapper and reducer control flow, verifies the relevant module bindings and constant, and accepts only the explicit-ID, already-typed, unique-left-ID, `format=None` domain. Conversion is an identity premise on that admitted projection; UUID assignment and formatting branches are unreachable. The source derivation verifies, all 5/5 recomputed-digest certificate mutations reject, and all 7/7 semantic source mutations still accepted by the old token-presence frontend reject. The checker is also hash-bound to the earlier 400/400 native–IR record, although the rule proof, not that corpus, supplies the lifting premise. The certificate is 1,078 bytes; median checking time over 101 repetitions is 4.26 ms.

Reducer-bridge lemma. Let L, R be finite lists of already-typed messages, every message have an explicit string identifier, identifiers in L be unique, and `format=None`. For the v1 target, additionally require that the suffix after the last remove-all sentinel contain no removal message. Assume the two library conversion calls preserve each admitted message. If the LangGraph checker accepts (C, P, π_B), then

$$C(L, R) = \mathsf{Eval}(P, \{\texttt{left} : L, \texttt{right} : R\}).$$

The wrapper delegates because both arguments are non-null lists; coercion assignments are identities by premise, and missing-ID branches are unreachable. The first loop records the greatest index containing the remove-all sentinel, so both programs return the same strict suffix when it exists. Otherwise, induction over R maintains equality of the ordered merged list, ID-to-position map, and removal set: an existing ID replaces in place or is tombstoned; a fresh non-removal ID appends; a fresh removal rejects. The final filter removes exactly the tombstoned IDs, and the checked `format=None` branch is the identity. Complete normalized-AST equality to the checker's source schema excludes every additional executable statement or altered predicate. The library-conversion premise, not its implementation, remains in the TCB.

Subsequent boundary correction. A bounded exhaustive diagnostic exposed a missing premise in the original v1 domain: native `add_messages` returns the suffix after the last remove-all sentinel verbatim, including removal messages, whereas v1 rejected such suffixes. Across all right-hand words of length zero through three over five fixed symbols and three left lists, v1 matches 429/468 native outputs. The prior 400/400 corpus did not exercise this boundary; it is retained but does not justify the original unrestricted lemma. A separately versioned v2 target preserves the suffix and matches 468/468 outputs on this diagnostic. These correlated constructed cases alone establish neither population accuracy nor an authenticated handoff guarantee. The v1 checker and certificates are not silently rebound to v2.

A separate v2 source checker binds the corrected target to the same complete source grammar. In the reset branch, the target now returns the suffix verbatim; the remaining stable-ID loop is unchanged. Complete authenticated tree reconstruction supplies the exact admitted input, so deterministic evaluation commutes with reconstruction. Together with the source-schema correspondence

and identity-conversion premise, this supplies a handwritten composition argument for the v2 projection, not a mechanized Python refinement proof. The receiver-local wrapper checks signed origin and program bindings, the bridge, evidence permissions and the cumulative ledger before returning a bound decision record. Either-side execution failure returns UNKNOWN; undetected failures of trusted materialization premises need not do so.

An integration diagnostic reuses all 468 exposed inputs, with identity and fixed cyclic-next pairings frozen before execution. All 936 decisions match the retained native outputs: 483 compatible, 390 breaking and 63 unknown; all 873 certified output pairs and decision records verify. Thirty regression tests pass, including seven reissued semantic-source mutations. This is retained-corpus integration evidence with public fixture keys and trusted projections, not 936 independent tasks, fresh native execution, or general workflow-continuation validation.

These bridges close the compiler-trust gap only for the stated source grammars. They do not certify arbitrary Python, message-projection/coercion fidelity, or the Vapi frontend; that frontend remains in the finite-assurance rows of Table 4.

P.1 COMPLETE-TREE COMPOSITION FOR THE AUTOGEN PROFILE

Let $T(x)$ be Closed IR v2's root-complete observation transcript and R the receiver reconstruction function. For admitted finite JSON, $R(T(x)) = x$: scalar rows contain their exact values; array shape fixes length and each indexed child; object shape fixes ordered keys and each escaped child path. Structural induction establishes the claim; exact closure excludes missing or extra descendants. Thus, for the deterministic predicate evaluator E, $E(P, R(T(x))) = E(P, x)$. Composing authenticated opening correspondence, this reconstruction identity and the checked source bridge yields the two result comparisons of Theorem 4 for the admitted fresh-instance projection. This is a handwritten composition argument. The additional predicate interpreter is explicitly trusted; no predicate opcode is silently added to Closed IR v2, and no selective-projection sufficiency is assumed.

The new receiver entry point checks origin context, a signed program envelope binding the predicate/profile and bridge, the bridge derivation, and complete input evidence. It then enforces the signed capability against actual proof openings and commits the cumulative ledger before returning a view-bound decision record. Detected failures return UNKNOWN, including identical invalid inputs; differing message trees can still yield CERT-COMPATIBLE when their predicate outputs agree. The trusted service sees the proof before granting output; this is not a transport guarantee against a client already possessing that proof. Origin fidelity and current materialization remain assumptions. The full-tree route may be more expensive than a full snapshot.

Retrospective regression rotates pairs within the existing 12 configured 50-case native blocks: all 600 pairs agree with the archived official results, including 274 breaking pairs. This reuses observed cases and makes no new external-transfer claim. Integration tests cover program/bridge substitution, wrong receiver, expiration, origin mismatch, missing capability, damaged proof, closure-denying permissions, budget denial and receiver-record substitution.

P.2 COMPLETE OFFICIAL-CONSUMER INVENTORY DIAGNOSTIC

Before new native outcomes, a manifest includes every direct `TerminationCondition` subclass in the pinned AutoGen 0.6.4 module: 11 classes, one explicit configuration each, and one-call/two-call horizons. All 22 tasks remain in the denominator. Message sequences and three grant policies are constructed and frozen; this is an unfiltered class inventory, not a random production-task sample. A separate SDK worker invokes the official conditions and records each call until termination, independently of the predicate IR. All 44 source/receiver executions complete normally.

Cumulative stage	Full grant	No shape	One atom
Inventory / JSON objects	22	22	22
Expressible consumer family	10	10	10
Checked bridge domain	5	5	5
Evidence permission	5	0	5
Budget	5	0	0
Correct certificates	5	0	0

The five certificates include four breaking and one compatible pair. Twelve tasks have unsupported consumers; five more require stateful semantics outside the fresh-instance bridge. Permission and budget arms repeat the same tasks, not 66 independent samples. The no-shape ledger returns `outside-capability`; the frozen runner's textual classifier calls this reason unexpected, but its raw decision and closure-stage count are correct. The no-shape and one-atom arms test this full-tree route, not the existence of a smaller sufficient proof for the same predicate. They do not establish minimal disclosure or an advantage over every authorization architecture. No rates are extrapolated beyond this inventory. This experiment connects the pipeline stages on one cohort while showing that broad task coverage remains unestablished.

Q Loopback Multi-Service Verification Plane

E20 composes three logical services over actual TCP/HTTP on loopback: an immutable provenance/policy registry, a compact-proof archive, and a receiver that resolves a content-addressed, pre-positioned application artifact and runs authorization, opening, capability, cumulative-ledger, and result checks. One SQLite-WAL ledger is shared across requests for each framework. The frozen v5 configuration uses the official-framework seed-611, team-4, 100-event, 512-byte-payload checkpoints, ten warmups, 101 measured requests, 16 concurrent requests, and ten trials for each of seven failure modes. A downstream transport operation has at most two one-second attempts; the injected archive delay is 1.5 seconds, so persistent delay still exhausts both attempts.

Table 20: E20 loopback verification-plane results. Latency is descriptive and includes local service composition; control bytes exclude the pre-positioned application artifact.

Framework	Healthy	Concurrent	Fault → Unknown	p50/p95 ms	Control KB
AutoGen	101/101	16/16	70/70	17.5/20.7	13.7
CrewAI	101/101	16/16	70/70	15.0/16.0	15.8
LangGraph	101/101	16/16	70/70	450.4/498.5	15.8

The faults are registry unavailable, archive unavailable, archive timeout, proof tampering, registry substitution, revoked origin, and expired authorization. All 303 healthy sequential requests certify, all 48 concurrent requests certify, and all 210 faults return UNKNOWN. LangGraph's receiver-local 100-event application artifact is 49.5 MB, versus 0.36/0.27 MB for AutoGen/CrewAI; receipt validation rehashes and admits that local object on every measured request. This explains the dominant local cost but is not an optimization result. The large latency remains a system boundary rather than an omitted result.

The retained iteration ledger matters. v1 used an AutoGen-only 20-ms timeout and failed on the first healthy LangGraph transfer. v2 increased the downstream timeout without changing the test population, passed every sequential/fault trial, but certified only 43/48 concurrent requests because it placed a 49.5-MB artifact in registry control-plane responses. v3 moved the artifact to each handoff request and exhausted host socket buffers under 16 duplicate uploads. v4 introduced the content-addressed data/control-plane split and reached 47/48; v5 added one bounded retry and passed the unchanged gate. These are observed architecture repairs, not independent confirmatory replications. E20 does not benchmark bulk artifact delivery, WAN, TLS, multi-host operation, Byzantine services, or production latency.

R Authenticated Evidence-Closed Projection

For path policies with unique clauses, write $q' \preceq_\pi q$ when every clause of q' can be discharged from valid base proof π_q by either the exact authenticated feature or a deepest first-blocking witness: an object shape omitting the next key, an array shape excluding the next index, or a scalar type forbidding descendants. This proof- and artifact-relative relation strictly contains textual clause inclusion but does not decide static program implication; q' retains its own output order. A base capsule consists of origin receipt ρ, ordinary consumer authorization α_q, and valid PPF2 multiproof π_q. A later authorization β binds $H(\alpha_q)$, $H(q')$, context, invocation, and expiry. The projector receives no source object, private archive, or signing key. It verifies π_q, computes all Merkle nodes

derivable from its opened leaves and boundary hashes, retains the witnesses required by q', and replaces discarded base leaves with their computed subtree hashes. The output is an ordinary PPF2 proof checked by the existing independent verifier.

Proposition (evidence-closed attenuation). Suppose π_q verifies under authenticated root ρ, $q' \preceq_\pi q$, and α_q, β verify for the stated programs and context. Canonical projection returns $\pi_{q'}$ such that PPF2 verification under (ρ, q') returns the same ordered results as a fresh archive opening for q'. Consequently the derived compatibility decision is identical. If a target is present but undisclosed, or coverage, authorization, or the base proof fails, the protocol returns UNKNOWN.

The proof follows level by level. Validity of π_q authenticates every selected value, shape, type, or first-blocking absence witness. Exact witnesses return their committed result; the three absence rules are precisely the PPF2 direct-opening rules, so each yields the same missing result. At each Merkle level, two authenticated children determine the parent; a discarded opened subtree can therefore become the sibling digest of a retained path. Induction reaches the same root ρ. PPF2 reconstructs each target result from one valid base witness, while β prevents substitution of a different target or context. This inherits the opening guarantee and signature unforgeability; it is not a new Merkle assumption. For b base-proof nodes, t retained leaves, and height h, the implementation uses $O(b + th)$ time and $O(b)$ memory, independent of archive size except through h. A persistently indexed archive can also answer in $O(th)$, so the measured timing is not an asymptotic advantage over an optimized online service.

The initial capsule signal reused four later task consumers across three 100-event checkpoints. Under symmetric receipt/proof/authorization accounting, bytes fell from 37,100 to 26,638 (28.2%); all four decisions matched independent PBCE, and 15/15 attacks abstained. The broader frozen projection study gives Table 21.

Table 21: Archive-free proof projection. Times are medians across frameworks; all 57 admitted target proofs are byte-identical to fresh proofs.

Events	Median source leaves	Fresh proof μs	Projection μs	Ratio
3	744	3,192	909	0.287
30	5,280	21,941	1,199	0.055
100	17,040	71,608	1,910	0.027

The nine workloads admit 57/63 targets: all 45 deterministic subsets and 12/18 named task consumers. Six CrewAI/LangGraph next-action policies are outside the resumed-trace capsule and reject rather than reuse incomplete evidence. The unchanged verifier accepts 57/57 projected proofs with transcripts identical to fresh archive construction and rejects all 36 prespecified base-proof, root, policy-order, and coverage attacks. Timing comprises 1,140 repetitions per arm. This confirms explicit path-subconsumer projection, not arbitrary program containment or minimal semantic evidence.

The disjoint 7/40/80 gate adds message-array/record shapes and one scalar type to the base. Per workload, five nontextual consumers query missing keys, an out-of-bounds message, a scalar descendant, and a mixed audit. Textual projection rejects all 45, whereas $\preceq_\pi$ admits them; all 57/63 admitted targets are fresh-proof byte- and transcript-identical. Six uncovered targets abstain, present-but-undisclosed values never become absence, and all 27 attacks reject. This spans every first-blocking PPF2 witness class, not arbitrary semantic containment.

Cross-language receiver confirmation. A third pre-outcome protocol uses disjoint histories 9/50/90 for all three frameworks. A standalone Node.js receiver with no Python/repository imports authenticates receipt, artifact, canonical PPF2 atoms, root, witnesses, and receiver evaluation. It accepts all 57/63 valid projections (45/45 nontextual) and exactly matches both Python transcripts and decisions on 57/57; all 45 proof, artifact, key, receipt, and policy attacks return UNKNOWN. This removes producer/verifier language sharing for one receiver path, not every JSON-number or language-runtime interoperability risk.

The source archive can still be false if the authorized attestor lies. PBCE prevents archive substitution and projection omission inside the admitted execution boundary; it does not certify the truth of the world or the safety of the consumer itself.

S NATIVE INSPECT RESCORING: INTEGRATION BOUNDARY

This exploratory diagnostic uses Inspect commit `40821b49e57b` (full revision in the artifact) and two one-sample logs from official local tests using a mock model. Both samples are rescored with the same later `includes` consumer; this is not a reproduction of the second sample's original store-reading scorer, a held-out task study, or a cross-version evaluation. The second score uses empty completion and target and provides no evidence of task-solving quality.

All arms authenticate historical evidence, reconstruct complete output and target subtrees, use the same native SDK hydration, and reproduce the complete native Score object. A conventional sparse Merkle map retains every top-level sample field before the query; it is not a consumer-specific projection. The common fixture commitment binds the complete-object digest and both representation roots. Correct construction of those representations remains trusted issuer work. Fixture signatures are retrospective, not evidence of original creation-time attestation or independent deployed authorization.

Table 22: Framed application bytes for two retained native log samples. All three arms reproduce both native scores. Parentheses cache the common source/profile package; other request and representation costs remain charged.

Representation	Sample 1	Sample 2
Signed full sample	3,839 (3,103)	2,744 (2,008)
PBCE subtree reconstruction	4,262 (3,526)	2,703 (1,967)
Generic authenticated map	2,055 (1,319)	1,835 (1,099)

The generic map wins both comparisons under either cache condition. PBCE alone pays for its expanded shape/leaf request manifest; all arms pay the common request, envelope, framing, and compressed source/profile where uncached. This is local application transfer, not WAN latency, verifier installation, storage amortization, or a production authorization-service benchmark. An earlier scalar-only comparison was cheaper but did not establish SDK hydration from that projection and is not substituted for this integration. An initial unsupported object-opening attempt and an intermediate accounting correction remain in the artifact; neither is a confirmatory gate.

Consumer admission is still narrow. The fixed inventory contains all 15 publicly exported scorer factories in `scorer.__all__` whose return annotation is `Scorer`; these are library factories, not independent tasks. Only the `includes` composition is admitted by the current structural bridge. SDK hydration, module binding, and Score construction remain trusted; whole SDK-to-result correspondence has not been independently established. Suspending consumers, helper dependencies, callbacks, and additional semantics remain outside the checked fragment. In particular, the seven normalization helpers shared by `exact` and `f1` identify a development dependency, not additional verified coverage. These results establish an executable integration path with a strong-baseline loss, not a new compiler contribution or a general PBCE transport advantage.

T APPWORLD: INDEPENDENT NATIVE-ASSESSMENT TRANSFER

Question and population. AppWorld provides state-based programmatic task assessments (Trivedi et al., 2024). We ask whether those native assessment outcomes can be reproduced using an authenticated archive instead of live application-model access. This is not an evaluation of an LLM solving AppWorld tasks, nor a test of compatibility between different AppWorld releases. The later acquisition cohorts below separately measure Agent task success. The runtime is pinned to repository revision `42b5bcf3cd33`, AppWorld 0.2.0.dev0, Python 3.13.13 and SQLAlchemy 2.0.52; full revisions and hashes are in the local manifest.

After development on training scenarios, an ID-only audit retains all 57 official dev IDs, covering 19 scenarios with three variants each. Neither IDs nor scenario prefixes overlap the 90-task, 30-scenario training split. Execution order is fixed by hashing a declared prefix and ID, with no outcome-based selection or replacement. The adapter, scripts, inventories and dependencies are hash-frozen before dev content is opened. This is local provenance, not independent timestamped registration. Test splits are not inspected.

Execution and trust boundary. Official validation solutions create the final states; native initial and final assessments give requirement identities, pass/fail vectors, counts and success flags. All ordinary model tables and both states are issued, not just tables observed in a query. The separate bounded Python-AST frontend uses authenticated record/ID access, a fixed forward-relation inventory and five admitted equality- count getters. Getter-target tables are re-encoded completely; no native database fallback is permitted. Re-encoding fidelity, profile bindings and native `TestTracker`/collection-helper semantics remain trusted or empirically checked assumptions. This frontend is *not* an instance of the finite closed IR proved in the main text, and its larger language does not inherit that theorem by association. Expected roots are trusted local commitments and evidence grants are unrestricted. This isolates content-hash-verified replay, not creation-time signing, permission enforcement or full deployment of the lifecycle protocol.

The query allowance remains 50 million combined raw-byte/step units, separate from the 500-million-unit per-state reissuance allowance. Native trials use two workers and 240 seconds per task; issuance and detached replay use two workers and 900 seconds per stage. Before evidence-only re-execution the stores close; audit hooks deny native-data/DB access and network operations. This is an instrumented process boundary, not OS-enforced isolation or a remote-execution proof. First, middle and last selected openings are removed in separate checks for each completed state.

Table 23: Complete official-dev denominator. Each scenario has three task variants; state evaluations and variants are not independent replications.

Stage or outcome	Count
Frozen tasks / scenarios	57 / 19
Native tasks available	57/57
Whole tasks replayed and matching	33/57
Fully matching scenarios	11/19
Unsupported tasks / scenarios	24 / 8
Native requirement vectors matching	66/66
Evidence-only vectors matching	66/66
Selected missing-block aborts	198/198

No completed native-vector mismatch is observed. All three variants agree on support within each scenario; reporting a binomial safety interval over 66 states or 57 variants would therefore overstate independent evidence. First unsupported constructs include arithmetic/other binary operations, explicit loops, and unadmitted helper, search and string-method calls. Unsupported does not mean the task was correctly answered. A final auditor recomputes native agreement from saved requirement identities and counts; proof-replay and missing-block counts remain execution-log evidence rather than independent reruns by that auditor.

Endpoint and coverage diagnostics. All 57 official-solution trajectories go from unsuccessful initial assessment to successful final assessment. Phase alone therefore predicts every aggregate success flag; this does not reproduce the requirement vector. As a retrospective diagnostic, assigning all initial requirements false and all final requirements true matches only 36/66 complete pass/fail vectors on the 33 covered tasks, even when the native requirement inventory is supplied. This oracle-assisted shortcut is not an executable authenticated baseline. All 33 covered tasks change their requirement vector between states. The median initial requirement count is two for covered tasks and six for unsupported tasks (ranges 2–10 and 5–8). Counts are not a validated difficulty metric, but coverage should not be treated as representative. These checks neither redefine the frozen endpoint nor establish performance on arbitrary intermediate or failed-solution states.

Final-state evaluation-order audit. A subsequent native diagnostic on a separate training task exposed sensitivity to evaluating the initial state before running its reference solution. We therefore reran all 57 development tasks in fresh per-task processes, executing each reference solution before any evaluation. All 57 native final requirement vectors were unchanged, and frontend final-state agreement remained 33/57. This retrospective check supports the reported final-state results; it is not a new held-out cohort and does not independently revalidate initial-state vectors or cryptographic cost measurements.

Development history is not independent confirmation. The first frozen train-scenario detached study matched 4/12 tasks. A second study matched 5/11 and included two executed native mismatches, not merely abstentions: an unavailable scalar normalizer was converted into a test failure. Repairs to primitive binding, exception scope and the supported language were made after observing those outcomes. The final development regression matches 19/23 exposed tasks; it is not held-out evidence and does not replace either earlier frozen result. The subsequent 33/57 dev result uses the frozen repaired method without tuning on dev outcomes.

What follows. This adds scenario-held-out, same-ecosystem evidence for selective native- assessment correspondence on a third-party task population. It does not establish arbitrary Python support, whole-workflow behavioral equivalence or ecosystem-wide safety. Native SDK hydration and semantic helpers are not removed from the trusted computing base merely because database access is detached. Full cost comparisons must separately include complete original state plus receiver reconstruction, not handicap the baseline by transmitting redundant derived tables. The separate cost follow-up below tests that comparison.

Protected benchmark instructions, solutions, test source and state derivatives remain local. Reproduction requires obtaining the benchmark through its official distribution; the paper reports aggregates and does not redistribute those materials. No model calls are used in this study.

T.1 Conditional warm-cache transfer cost

Design and strong baseline. The follow-up retains all 57 slots and 19 scenario clusters from the behavior study, without replacement or cost-based exclusion. It reuses slot zero as an explicit implementation preflight, so the entire cost batch is not outcome-unseen. Each of the 33 executable tasks contributes initial and final views. Five arms receive the same fixed training-state cache and patch dictionary: query CAS, complete derived JSON, complete derived CAS, complete original JSON and complete original CAS. Original-source receivers reconstruct the fixed relation/count metadata and check the same derived root before executing the same assessment. The common recipe is preinstalled for every arm; no native database fallback is allowed. This avoids charging only the complete baseline for redundant derived tables.

Each arm uses zlib6 and Zstd patch levels 3, 9 and 19. All 40 combinations per task must restore exactly; the smallest restored candidate for each arm is replayed, giving ten native-vector comparisons per task. The complete baseline is the minimum across all four complete arms for each state, selected by bytes, not correctness. Byte counts include the same 64-byte authentication allowance, not a measured full application handshake. Task ratios sum initial and final bytes before division. Variants and states are not independent samples.

Table 24: AppWorld conditional cost follow-up. Lower ratios mean fewer query bytes than the strongest tested complete-reconstruction transport. Missing costs are not assigned zero.

Outcome	Value
Frozen tasks / scenario clusters	57 / 19
Completed cost tasks / covered clusters	33 / 11
Unavailable tasks	24
Median task byte ratio, best codecs	0.630
Median task byte ratio, fixed Zstd9	0.655
Tasks with fewer query bytes, either comparison	33/33
Selected transport/native-vector agreement	330/330

Limits and accounting. One task has a best-codec ratio of 0.9985, almost no benefit. One individual final-state query is 1.0248 times the complete baseline even though its task sum is lower. Thus neither a universal state-level win nor a correctness gain follows. All comparisons condition on the 33 supported tasks; the other eight scenario clusters contribute no useful-query cost measurement.

The fixed dictionary occupies 390,277,919 local bytes, the cache archive 184,356,864 bytes, and the shared recipe 38,804 bytes. These are stored representation sizes with overlapping information, not measured initialization traffic, minimum memory, or a cold-start requirement to sum together.

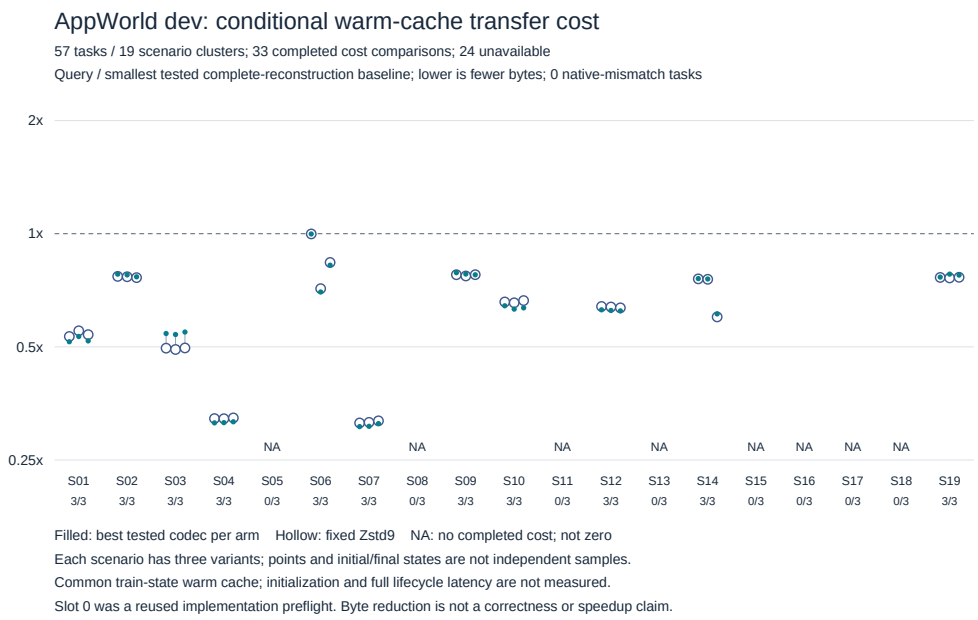

Figure 4: All 19 dev scenario clusters, including eight with no supported cost measurement. Each point aggregates both states of one task variant. Filled and hollow points compare best tested codecs and fixed Zstd9, respectively. The dashed line marks equal transfer bytes; the baseline may choose any tested complete-reconstruction representation. This is a warm-cache conditional result, not an estimate over independent task draws or of full-lifecycle speedup.

Setup amortization is unmeasured. Sequential shared-host timings separate query discovery, export, compression and receiver reconstruction/evaluation where instrumented; reconstruction is included in the receiver total, not added twice. They do not establish latency superiority or full-lifecycle speedup. A separate auditor checks the full denominator, frozen hashes, actual compressed sizes, codec grid and native-agreement ledger; byte re-decoding is audited separately. Fresh re-decoding of all 1,320 logged codec candidates reproduces every original byte stream and matches all recorded compressed byte counts. This does not rerun native assessments or constitute an independent implementation of the codec.

T.2 Empty-cache deployment diagnostic

After observing the warm-cache experiment, we freeze a separate diagnostic on the same 57 slots. This is not another independent behavioral confirmation. The only deployment change is that the receiver has no data blocks or patch dictionary. Its common interpreter and recipe remain installed. Every arm uses plain Zstd9; the complete baseline again chooses the smallest of original or derived JSON/CAS, not an uncompressed representation. Each phase independently starts empty. Adding the two phase costs within a task is therefore not a multi-query session with evolving cache reuse.

Producer-side discovery reconstructs the original opening set using the saved warm packets and immutable training archive, checking its original count, bytes and native vector. Cold query packets include *all* opened blocks, including previous cache hits. The producer store then closes; query replay uses only the restored packet and matches 66/66 native vectors. Complete CAS packets are whole-root exports, not query projections; their materialized JSON hashes match the prior complete-arm artifacts. All 330 cold transports are exactly restored. This links complete transports by byte equality to previous full-arm execution; it is not a fresh native replay of all five arms. An independent accounting pass re-decodes all 330 cold files without a dictionary, matching every original hash, length and reported compressed size.

Table 25: Same supported dev tasks under two data-cache conditions. Both rows fix Zstd9 for every arm. Ratios are per-task ratios, whereas byte columns are separate medians, so their quotient need not equal the median ratio.

Condition	Median query B	Median complete B	Median ratio
No data cache/dictionary	90,794	62,400,533	0.001455
Training-state cache/dictionary	45,756	117,651	0.654925

All 33 cold task ratios are below one, ranging from 0.001402 to 0.014187; the minimum complete representation is original JSON for all 66 states. The 24 unsupported tasks remain without useful-query cost observations. The large cold reduction follows from sparse access compared with sending the full state; strong cached complete reconstruction substantially narrows that gap. Neither comparison establishes a new authenticated-data primitive, superiority to all selective representations, or end-to-end latency improvement. The fixed codec diagnostic does not optimize all possible cold compression algorithms.

T.3 Original-author selective-proof comparison

To separate the benefit over complete transport from a comparison with selective authentication, we apply the original author's naive, buffered and buffered-magic proof engines to complete logical-node exports of the same frozen dev roots. The authentication/proof implementation and local container image are fixed; our adapter assigns logical paths to leaves and replays the unchanged v9/v5 consumer on the returned facts. Internal nodes carry container shape rather than descendant payload. Exports contain the complete tree, not only queried nodes. This is a post-outcome layout diagnostic: slot zero reuses its exposed implementation preflight, and neither the consumer semantics nor support pool is adapted to these results.

Both responses use dictionary-free Zstd9 and the same 64-byte accounting convention. The author response includes rank framing; logical request-path metadata is omitted, giving that comparator an optimistic response-only account. This does not compare full protocol traffic, archive issuance or latency. Buffered-magic is the prespecified primary author arm, not a winner selected after observing the other two arms.

Table 26: Original-author logical-node proof comparison on all 33 supported dev tasks (11 scenario clusters). Ratios are DAG query bytes divided by author response bytes; each task sums its independently cold initial/final states.

Author arm	Median task ratio	Lower-byte tasks	Lower-byte states
Naive	0.325	33/33	66/66
Buffered	0.332	33/33	66/66
Buffered-magic	0.448	33/33	66/66

All 198 author-arm/state replays match the corresponding native requirement vectors. The primary task ratio ranges from 0.437 to 0.892, with median 0.4484 (55.2% lower paired response bytes); no individual state reverses the byte ordering. A separate file audit re-decodes 198 author transports and 66 DAG transports, verifies lengths/hashes/accounting and frozen source identities, and retains all 57 slots. That audit does not rerun native assessment or cryptographic verification. The other 24 tasks remain unsupported, not zero-cost successes.

The original batch completed 13 tasks before the 20 GiB disk reserve prevented the remaining 20 supported tasks from starting. A separately logged continuation completed nine; a second completed the final eleven. All original infrastructure failures remain in the audit history. No completed task was repeated or replaced; algorithms, proof arms and limits were unchanged. Redundant raw JSON was removed only after verification against retained lossless archives with recovery hashes.

This result supports a transfer advantage over the measured complete logical-node adapter of an original-author implementation. It does not establish that generic authenticated data structures can-

not use the DAG layout, that this is an optimal selective comparator, or that authentication expressiveness is new. The earlier small-object and event-log losses remain part of the backend frontier.

T.4 Direct-DAG attribution diagnostic

To test whether the preceding advantage depends on the logical-node adapter, we fix the already exposed slot zero and both historical states. The original library's unchanged authentication and buffered-proof engines directly open DAG nodes; there is no outer Merkle tree. Child commitments are read only from authenticated parent nodes, starting from an externally trusted root. The complete exports contain 484,021 and 484,039 nodes; each consumer access list contains 804 terminal-node observations. Our adapter uses a custom object, array and scalar string encoding and the library's string hash convention. Consequently this is not a wire-identical factorial comparison, nor a test of an optimal authenticated data structure.

Table 27: Exposed single-task direct-DAG diagnostic. Response bytes use dictionary-free Zstd9 plus the common 64-byte convention.

Representation	Initial	Final	Total
Our typed DAG	45,206	45,683	90,889
Author library, direct DAG, naive	33,072	34,767	67,839
Author library, direct DAG, buffered	30,496	30,921	61,417

All four author-arm/state runs reproduce all 804 expected node observations; all four compressed proof files independently re-decode to their recorded hashes. These are node-level checks against the retained access lists, not new native assessment executions. Buffered proofs contain 317/320 records, versus 4,824/4,824 without buffering. Modified-first-record, missing-first-record and wrong-root tests are rejected in each run (12/12 targeted checks). Appended unused records are accepted by the underlying API; thus this adapter does not establish canonical proof consumption or rejection of every malformed stream. An initial container mount failure occurred before execution and was retained; the corrected mount did not change frozen inputs or the algorithm.

The direct-library buffered response is 32.4% smaller than our typed DAG on this task. This counterexample rules out attributing the preceding 33-task advantage to authentication alone. It does not estimate a 33-task reversal: only one exposed task was measured, encoding differs, and total protocol traffic, preparation cost and native consumer replay were not compared here. We retain the original layout result but make no generic selective-backend superiority claim.

Retained-population direct-DAG expansion. We subsequently freeze the same adapter, author-library binary, Zstd9 setting, access lists and complete input packets for all 57 original dev slots. The 24 unsupported slots remain excluded from cost comparisons but included in the coverage denominator. Slot zero reuses the diagnostic above; the other 32 supported tasks receive new initial/final executions of both naive and buffered author modes, with no outcome-based task replacement or tuning. This is a post-development baseline expansion, not held-out task transfer.

Table 28: Direct-DAG expansion, summing initial/final response bytes per task. Ratios are our typed DAG / buffered author DAG; lower favors ours.

Population	Tasks	Ours wins	Author wins	Median ratio
All supported slots	33	9	24	1.473
Excluding reused slot zero	32	9	23	1.473

All 33 supported tasks complete; none fails or is replaced. The 132 author- mode/state runs match all 129,338 requested node observations, and their compressed proof hashes and record counts revalidate. All 396 modified, missing and wrong-root checks reject. All 132 appended-unused-record checks remain accepted, so canonical proof consumption is still not established. These repeated observations are not independent task samples or new native assessment executions. The byte ratio ranges from 0.820 to 1.485. Our nine wins lie in three of the 11 supported scenarios; the other eight

scenarios favor the author adapter in all three task instances. This descriptive clustering is not a prospective selection rule or population-level estimate.

The expanded result narrows the earlier logical-node advantage to that representation choice. Neither implementation uniformly dominates, but the direct author baseline is cheaper on most retained tasks. Different encodings still preclude attributing the difference to authentication alone. Runs use the pinned offline image with two task workers; retained source hashes remain unchanged and no scientific parameter or code was changed mid-run. The 64 new container jobs take a median 16.37 seconds per state, excluding host conversion and including process startup and verification; this is not an end-to-end latency comparison against our backend.

T.5 Lossless wire compaction of the typed-DAG adapter

The original wire packet explicitly carries each node's 32-byte digest even though its bounded decoder recomputes that digest from the canonical node. We implement an alternative `DHE1` packet with a four-byte version tag, node count and length-prefixed raw nodes. The decoder recomputes content addresses, rejects duplicate nodes, truncation and trailing bytes, and returns the same hash-to-raw map. The existing DAG reader still requires the trusted root and validates the nodes it reads: parsing alone authenticates neither the issuer nor a task result. Nodes are not re-encoded, source roots do not change, and any signature bound to packet bytes must bind the new format rather than reusing a signature over old bytes.

For n nodes, the uncompressed size satisfies $L_{\text{elided}} = L_{\text{explicit}} - 32n + 4$. This identity does not guarantee a compressed-size improvement. We additionally test first-access ordering, followed by hash-sorted unvisited nodes; no node is omitted. A fixed 2×2 diagnostic on the same 33 supported tasks crosses explicit/elided digests with hash-sorted/access order, using unchanged dictionary-free Zstd9 and the 64-byte response convention. Elided-plus-access is specified as the candidate before these costs are measured. The original 24 unsupported slots remain in the denominator. This is development on exposed data, not independent transfer or a new cryptographic primitive.

Table 29: Wire compaction on 33 retained tasks; summed initial/final bytes. Ratios below one favor the listed arm. All four arms are retained.

Digest / order	Median / old	Median / author	Wins vs author
Explicit / sorted (old)	1.000	1.473	9/33
Elided / sorted	0.706	1.037	15/33
Explicit / access	0.996	1.469	9/33
Elided / access (candidate)	0.702	1.032	15/33

All 264 arm/state packets reconstruct the exact old block map and preserve requested node observations. A separately implemented parser, importing no candidate codec, checks all elided packets; the explicit/sorted arm reproduces the saved baseline packet and compressed bytes exactly. The tested decoder limits are 16 MB per packet, 100,000 nodes and 8 MB per node. Malformed-input tests cover truncation, invalid versions, duplicate blocks, byte/node budgets and a changed root node that remains parseable but fails trusted-root lookup. The codec is an additional transport, not an automatic replacement for old signed messages or a new native task assessment.

The candidate is smaller than the old transport in all 33 tasks, with a median reduction of 29.8%; hash elision accounts for most of this improvement. It is smaller than the direct author baseline on 15 tasks but larger on 18, with median ratio 1.032 and range 0.704–1.036. Thus practical wire redundancy explains much of the earlier deficit without establishing algorithmic superiority over generic authenticated stores. The remaining gap is not used to retune encoding or select favorable tasks within that diagnostic. Codec-only timings do not establish an end-to-end latency advantage.

T.6 Backward references and an equally optimized author baseline

A subsequent development extension, `DHR1`, eliminates another wire redundancy: child digests repeated inside parent records. Included blocks are ordered deterministically in postorder; occurrences of an already reconstructed block's hexadecimal digest may be replaced by a backward index. Other

digests remain literal. The bounded decoder reconstructs each original raw node and content address exactly. The trusted root, DAG reader, and authentication assumptions are unchanged. This is lossless transport engineering, not a new authentication primitive, and packet-bound signatures must bind the new bytes.

On all 33 supported development tasks, DHR1 uses a median 0.681 times the original-author response bytes and wins 33/33. This comparison alone overstates the evidence for a representation advantage: applying the same digest-reference optimization to the author proof narrows the median ratio to 0.981, still with 33/33 wins. The author adaptation preserves the exact original binary proof, including order and repeated records; its ordering indices are charged. Both arms use file-based Zstd9 without dictionaries and the same 64-byte per-response convention. All 66 author proofs are recovered byte-for-byte. Different record layouts and interfaces remain, and an even better generic store implementation is not excluded. The original 24 unsupported tasks are retained in the 57-task denominator.

Public fixed-edit development extension. Twelve public-mode tasks outside the preceding 33-task comparison were selected, one per scenario, before this transport comparison. Their lifecycle and admission outcomes subsequently became visible. The initial profile supports six; successive exposed-task extensions support seven (v11), ten (v12), and all twelve (v13 with archive adapter v6). All twelve receive the same local note creation/update and named checkpoint save/load procedure, not a solution to the original task. This is not untouched held-out task-solving validation.

We replay all twelve complete declared model-domain archives with the identical v13 evaluator and v6 adapter: all 24 initial/resumed requirement vectors match the retained native references, including packet-only replay. Historical archive provenance remains unchanged; the new evaluator profile is separately bound in the replay manifests. No native database fallback is used by detached replay. The original-author verifier was run on full-root conversion for each task. Its existing proof evidence is reused in the uniform summary only after root, requested paths, node-map packets and compressed wire bytes match exactly; independent parsers recheck proof identity and costs. The original-author container is not represented as freshly rerun by this consistency check.

The extensions admit bounded arithmetic, grouping, sorting, loops over admitted collections, locally owned list appends, and safe YAML parsing. One task additionally requires native query observations: issuance records the complete email lookup and ordered file relation for all 151 users and 23,310 files in each state, not only the eventual consumer's read set. The receiver reads these authenticated observations without invoking the SDK. Faithful observation remains an issuer premise, not a proof of native SQL semantics. Retained archives have heterogeneous issuance versions; uniform execution does not imply uniform issuance. A separate six-run diagnostic resolved SDK database-routing contamination by resetting the official routing context after each interpreter attempt. Historical failures are retained; no original requirement was dropped. These host-profile extensions remain outside the closed-IR theorem.

Table 30: Exposed public-trajectory development results under uniform v13/v6 replay. Bytes sum initial and resumed responses; the optimized comparator receives the same digest-reference optimization. All twelve original candidates are supported.

Selection index	DHR1 bytes	Optimized author bytes	Ratio
0 (preflight)	280,104	374,371	0.748
1	186,306	232,551	0.801
2	70,822	78,297	0.905
3 (support extension)	335,517	464,137	0.723
4	69,559	77,420	0.898
5	61,273	66,302	0.924
6	233,270	238,780	0.977
7	41,083	41,622	0.987
8	117,318	142,249	0.825
9	841,072	1,150,667	0.731
10	40,689	41,184	0.988
11	168,833	207,660	0.813

DHR1 wins all twelve task-level comparisons: median savings are 29.9% versus the unoptimized author transport and 13.8% versus the equally optimized arm. Excluding the reused slot-zero preflight leaves eleven wins with 10.2% median savings against the optimized comparator. States are paired within tasks, not independent replicates; no IID population interval is inferred. The earlier six-task result is retained (8.6% median savings, or 7.6% excluding preflight), as is the seven-task result (9.5%, or 8.6% excluding preflight). Uniform replay fixes the mixed-evaluator-version issue but does not erase the adaptive development history, increase task success, or prove general superiority.

T.7 Symmetric response caching and online cost

The following development studies preserve the original-author verification worker and give both methods the same BFS 256 KiB/2048-node page membership, optimized path reader, codec, fresh receiver state, and Linux 2-CPU/3-GiB environment. Delay is added service sleep, not measured WAN RTT. An initial all-page cache study on exposed task 1 preserves 44/44 assessment vectors. At zero added delay, candidate median online time decreases from 0.299 to 0.115 seconds and author time from 0.307 to 0.121 seconds; cached medians at 20 ms/request are 0.438/0.445 seconds. However, response preparation costs 13.25/7.38 seconds and about 40.1 MB per backend, separately from source loading and indexing. This generic optimization does not establish an exclusive algorithmic advantage.

Before new outcomes, we freeze exposed initial tasks 1, 6, and 9 for a symmetric lazy-cache replication. Each task has four warmups and forty interleaved measured replays: four arms, two delays, five repeats. The 64-MiB payload-bounded LRU cache is bound to root/layout/wire/codec. Warmup state is discarded; each delay series starts empty. Miss encoding is timed inside requests. Receivers reverify evidence on every replay. All 132 complete native vectors match; exact schedules, frozen hashes, evidence counts and cache lifecycle checks pass. No API is called.

Table 31: Post-cold online medians in seconds (four repeats per cell). Tasks are exposed development cases, not independent confirmation.

Task	Delay (ms)	Candidate	Candidate+LRU	Author	Author+LRU
1	0	0.14286	0.05498	0.16712	0.05940
6	0	0.30364	0.10101	0.31868	0.11515
9	0	0.23333	0.09788	0.24379	0.10377
1	20	0.46666	0.36648	0.45724	0.36156
6	20	1.16517	0.93681	1.15464	0.96131
9	20	0.71357	0.55425	0.73014	0.56406

Only 12/30/17 pages out of 1052/1069/1052 are cached (1.008/2.200/1.681 MB). No eviction occurs; unit tests separately exercise eviction and oversize cases. Cold zero-delay times are 0.13582/0.30551/0.22306 seconds for the candidate and 0.14138/0.31863/0.25403 for the author. Cumulative five-replay times retain all outliers: 0.35793/0.71100/0.63490 versus 1.99714/0.84834/0.67507 seconds. The latter task-1 total includes a 1.6809-second observation with only 0.0274 seconds in the verifier bridge; its cause is not established. We do not treat this as an 82% algorithm advantage. At 20 ms/request, five-replay candidate savings versus the author are 0.70%, 7.09%, and -0.16%. Source loading still costs 87.7–103.6 seconds; per-arm indexing 1.22–1.91 seconds, and source-format conversion 2.72–4.65 seconds. These are not included in online timers. The result supports reduced repeat encoding, not universal speed superiority, newly solved tasks, or a task-population confidence claim.

Shared-harness follow-ups. A separate instrumented task-1 repetition localized 0.7359 seconds of a 0.8167-second author replay to parent-process garbage collection. This does not identify every earlier outlier. Applying the same bounded-lifetime heap policy to both arms preserves 44/44 vectors: collecting/freezing prepared objects costs 0.5815 seconds, cleanup 0.0931 seconds, and measured replay including warmups 11.5679 seconds. Source loading, indexing, conversion and worker startup remain additional costs. The largest per-replay GC time is 0.0164 seconds. This sequential diagnostic is not a randomized latency-improvement estimate.

Complete-source preparation can also avoid materializing unused JSON while still checking every reachable unique DAG node, logical depth and unfolded-visit budgets. On the same three exposed tasks, six old-first/graph-first paired comparisons preserve every complete-map fingerprint, block count and raw-byte total. Descriptive median preparation-wall and CPU savings are 4.49% and 9.44%. Both backends can use this optimization; repeated measurements on three tasks do not establish an exclusive algorithm advantage or an Agent speedup.

Matched-page bandwidth frontier. A further frozen grid on the same exposed tasks uses two cached arms, unlimited/1/10/100 decimal Mbps response pacing, 0/20 ms added request delay, and five repeats per cell (one cold, four warm), plus two warmups per task. Every receiver starts without evidence and independently verifies received pages. The same static-parent GC policy applies to both arms. All 246 vectors match; source hashes, exact randomized schedules, cache reset/continuity/budgets, matched query/evidence counts and serialization bounds pass the audit. Pacing delays actual framed loopback response chunks; this is not a WAN/TCP model, and requests are unthrottled.

Table 32: Equal-task median latency savings over the author backend, in percent. Negative means the candidate is slower. Warm refers to the four post-cold repeats; cumulative includes all five. Three exposed tasks are the units, not the 246 repeated observations. All grid cells and outliers are retained.

Response rate	Delay (ms)	Cold	Warm	Cumulative
Unlimited	0	-1.27	7.63	-5.25
Unlimited	20	3.34	1.19	-0.43
1 Mbps	0	0.13	0.17	0.44
1 Mbps	20	-0.40	0.16	0.09
10 Mbps	0	1.68	0.34	0.55
10 Mbps	20	-1.40	0.41	1.15
100 Mbps	0	-1.40	0.22	0.04
100 Mbps	20	1.86	1.80	2.66

Framed response bytes are 1,008,244/2,199,779/1,680,683 for the candidate versus 1,007,589/2,200,266/1,680,657 for the author, invariant across regimes. Thus the earlier 13.8% trace-packet saving does not transfer to matched pages. Unlimited zero-delay warm savings are 4.94%, 7.63%, and 12.23%; cold and cumulative results are mixed (Table 32). Preparation, conversion, startup and GC costs remain separately available. Converting a shared candidate-native archive into the author's representation is harness preparation, not an intrinsic author-backend deployment penalty. No population-level speedup, non-inferiority or end-to-end Agent gain is inferred.

Complete exposed public-cohort replication. After the three-task diagnostics, we freeze all twelve existing public-trajectory slots, without selecting by timing, and retain the same implementations and matched-cache environment. Each task has two warmups and ten interleaved repetitions at each of 0/20 ms added request delay: the first is cold and the remaining nine retain server cache, while receivers always reverify evidence. Task order is fixed; delay and within-pair arm order use seed $2026091919 + i$. There is no response-rate limit. This post-diagnostic expansion is descriptive, not a fresh holdout or the separate 33-task development cost cohort.

All 12 processes exit normally, and all 504 native initial-state assessment vectors match. Independent verification is invoked on every request; source hashes, schedules, query/evidence counts, and cache lifecycle checks pass. Twelve audit implementation tests include deliberately corrupt results, skipped verification calls, changed source hashes, missing rows, and changed bindings; these are not extra benchmark cases.

At zero added delay, warm task-level savings range from 2.87% to 16.62%. Slot zero's cumulative time instead increases 8.29%, including two slow candidate observations; both remain included. Response bytes do not explain the warm advantage: the candidate wins only 3/12 byte comparisons and its paired-median response size is 0.038% larger. Source loading alone costs 95.49–102.73 seconds; indexing, representation conversion, worker startup and GC preparation/cleanup are additionally re-

Table 33: Complete exposed 12-task matched-cache study. Savings are equal-task medians; wins count tasks, not repeats. Warm medians use nine post-cold replays; cumulative includes all ten. No observations or outliers are removed.

Delay (ms)	Endpoint	Median saving	Candidate wins
0	Cold	6.12%	10/12
0	Warm median	9.14%	12/12
0	Cumulative ten	7.16%	11/12
20	Cold	−0.35%	5/12
20	Warm median	2.08%	9/12
20	Cumulative ten	1.69%	12/12

ported. Consequently the experiment supports a conditional loaded-assessment replay advantage, not cold whole-system deployment savings, Agent task-solving gains, or broad latency superiority.

Restored-state replication. After the initial-state audit, we freeze a second run on the same twelve task slots, replacing only the authenticated root and native reference with their previously retained restored-state counterparts and using seed $2026091920 + i$. Implementations, page layout, compression, verification, cache and resource limits remain unchanged. No Agent is rerun and the actual restoration operation is not timed. All twelve processes terminate normally and all 504 restored-state vectors match. The same schedule, hash, verification and cache audits pass. A supplementary identity audit binds both runs to the original task manifest and phase-specific DHR1 roots; AHR1 has different representation roots. All twelve state roots differ, but native assessment vectors change in only three tasks. Thus this is a state-conditioned performance replication, not 24 independent tasks or a broad test of detecting changed task outcomes.

Table 34: Restored-state replication on the same twelve exposed tasks. Definitions match Table 33.

Delay (ms)	Endpoint	Median saving	Candidate wins
0	Cold	2.91%	8/12
0	Warm median	10.44%	12/12
0	Cumulative ten	8.49%	12/12
20	Cold	0.82%	7/12
20	Warm median	1.75%	10/12
20	Cumulative ten	1.71%	9/12

Zero-delay warm savings range from 5.81% to 14.82%; the paired-median response-byte reduction is only 0.064%. At 20 ms/request three tasks lose on cumulative cost; all observations remain. Timing includes connection setup, transfer, decoding, verifier-worker IPC and assessment, but excludes source preparation, worker startup, final connection teardown and Agent execution. The two implementation languages also differ; no language-independent cryptographic or asymptotic superiority is established.

T.8 Retrospective full-population capability extensions

After the original development-split study, all 57 task sources and outcomes are exposed. Subsequent language extensions therefore test development coverage, not independent generalization. Each new batch retains all task identities, both states, original archives, native references and query budgets; no new tables are selected from observed reads. All 114 native initial/final vectors remain unchanged in fresh native reruns. Native-only and archive-only support must be distinguished: v15 matches 54/57 with live model access, but the same frontend with the unchanged archive adapter matches 45/57.

V15 adds bounded flattening, unique scalar selection, grouping into sets, exact string affix operations and scalar membership. Its 102 unit/regression tests include 200 deterministic random helper comparisons against the pinned SDK; these are not independent tasks. Twelve tasks still abort: six require Spotify user lookup, three require filesystem user lookup, and three need nested JSON traversal not supplied by this adapter. No prior supported task regresses. An independent audit

Table 35: Complete-pool replay development; three variants per scenario. The original cost studies still use their original populations.

Profile	Tasks	Scenarios	Vectors	Missing-block aborts
Original frozen v9	33/57	11/19	66/66	198/198
Retrospective v14	39/57	13/19	78/78	234/234
Bounded collections v15	45/57	15/19	90/90	270/270
V15 + bounded JSON traversal	48/57	16/19	96/96	288/288
V15 + unique-query bridge	51/57	17/19	102/102	306/306
V16 + order-free relations	54/57	18/19	108/108	324/324
V16 + complete partitions	54/57	18/19	108/108	324/324
V17 + invariant reductions	57/57	19/19	114/114	342/342

recomputes full native vectors and rehashes source files and immutable archives; replay and attack execution counts remain log evidence, not a second independent cryptographic implementation. A subsequent frozen adapter adds only bounded traversal of exact dictionaries inside authenticated record fields, retaining direct-field precedence and abstaining on missing keys or unsupported objects. Eighteen lookup/relationship tests include 200 random comparisons against the official getter and packet-only replay with each opening removed in turn. Its full-population replay matches 48/57 tasks across 16/19 scenarios, including all 96 initial/final vectors and 288/288 selected missing-block aborts, with no prior-task regressions. Nine user-lookup tasks remain unsupported; no native database fallback is introduced.

A further development bridge scans complete authenticated tables for unique equality matches, aborts on ambiguity, and supports a unique player relation and bounded queue-cursor observation. This avoids assuming a database's first-match order, at the cost of full-domain evidence. Forty combined tests include native Spotify/filesystem ORM checks and 200 randomized SQLite equality fixtures. The full 57-task replay reaches 51/57 across 17/19 scenarios: all 102 native vectors match and 306/306 selected missing-block trials abort. None of the prior 48 tasks regresses. Six tasks still lack supported multi-record relations. SDK source hashes and empirical differential checks do not establish a general SQL refinement theorem; faithful archive issuance remains a premise.

V16 exposes reverse relation evidence as an opaque order-free collection, not a guessed SQL sequence. The frontend permits scalar set projections and unique-key map projections, but rejects positional access, ordered iteration, list conversion and duplicate map keys. Maps admit lookup/membership and set-valued key observation, not ordered views. The complete target table and join targets are authenticated; missing records, type mismatches and budget exhaustion abort. Thirty-five adapter/primitive tests include real official Spotify/filesystem ORM fixtures; 45 frontend regression tests also pass. The full population reaches 54/57 and 18/19 scenarios, with 108/108 matching native vectors, packet-only replay agreement and 324/324 selected missing-block aborts. All prior 51 tasks remain supported. Three filesystem tasks exceed the unchanged relation budget; they are not dropped or rescued by raising limits. This is order-insensitive observational correspondence, not equivalence of arbitrary ORM list order. A preliminary batch failed before any assessment because a protocol filename was wrong; it is retained as a launch invalidation. The corrected, separately named batch reran all 57 tasks with identical algorithm modules and original archives. Full-domain scans establish no new efficiency claim.

A separate complete-partition extension scans the authenticated source tables at construction, checks their entire archived ID domains and commits equality groups. Queries open the selected group and verify each original record, without raising the 10,000-row relation budget or 50M proof budget. The 23,310-row file table thus no longer requires a full query-time scan. Index-only replay still covers 54/57: the same three tasks reach an unsupported set comprehension. V17 additionally admits a pure membership comprehension only when immediately consumed by a trusted literal `all is`/`any is` boolean reduction; ordered list observation, filters and callbacks remain unsupported. Ninety-four component/regression tests pass, including 512 finite native reduction comparisons; a different Python hash seed repeats the reduction tests. Empty-input exception types follow the pinned SDK, not a replacement oracle. All 57 tasks then match both native vectors, with packet-only agreement, 342/342 selected missing-block aborts and no prior-task regression. A frozen hash-seed-41 repeat of the same 57 tasks preserves all 114 reference vectors and 342 aborts. Native SDK references were

not regenerated under that seed; this checks detached repeatability, not additional tasks or native refinement.

Derived-index completeness remains trusted: root/profile binding and selected-row checks cannot detect a dishonest reissuer's omissions, as a negative test demonstrates. Original archives are unchanged. Index-only construction costs a median 1.30 seconds and 2.53 MB of raw derived data per state before deduplication, opening 16.94 MB of source data. These two-worker observations exclude replay cost. Across 108 previously supported states, median proof bytes increase 0.089%; only six improve. Added coverage is neither overall speedup nor new cryptography. Original coverage and index-only abstentions remain reported; this is not independent transfer.

An intermediate v13 batch scored six missing-method AttributeErrors as task failures (45 returned, 39 matched). V14 aborts these cases; all six remain unsupported at that stage. A separate initial-state harness failed on SDK traceback formatting; a physical-source rerun retained all 57 with unchanged scoring. Both failed batches remain archived. Neither repair extends closed-IR coverage, Agent task-solving gains or transfer-cost results.

T.9 Frozen acquisition and post-outcome replay repairs

Before model execution, we fixed 36 public test-normal tasks in twelve scenarios, excluding exposure in local development records. This does not establish absence from model training data. Acquisition used 320 HTTP attempts under a 432-attempt cap. All planned tasks remain in the denominator, including one checkpoint whose native assessment changes after restoration. No additional model calls were used to repair or reevaluate the replay frontend.

Table 36: Distinct endpoints on the 36-task cohort. Repaired results are development evidence on already-observed tasks, not a second held-out test.

Endpoint	Observed count
Agent task success before restoration	11/36
Native checkpoint vector preserved	35/36
Agent task success after restoration	10/36
Frozen frontend: verified tasks	24/36
Frozen frontend: fully covered scenarios	8/12
Repaired frontend: verified tasks	35/36
Repaired frontend: fully covered scenarios	11/12
Repaired frontend: solved and verified tasks	10/36
Repaired frontend: native phase vectors matched	70/70
Repaired frontend: detached repeated checks	700/700
Repaired frontend: selected missing-block aborts	210/210

The frozen frontend abstains on eleven restorable tasks: three stop at field resolution, three at string title conversion, three at aggregation and two at Boolean evaluation. Subsequent fixes add bounded primitives, short-circuit semantics, interpreter-owned dictionary updates and an explicitly trusted `admin.MainUser.name` bridge checked against the native getter. All 24 previously verified tasks remain supported. These extensions do not inherit the closed-IR theorem or establish arbitrary Python translation correctness. The separate checkpoint repair is not substituted into the frozen cohort.

The independent funnel audit reconstructs native requirement-vector identities; the repaired-run audit checks both phases and detached outputs. Repeated queries and selected block-removal checks are implementation tests, not independent tasks. Variants share scenarios; we make no IID population-confidence claim. The accompanying artifact includes acquisition, frozen replay, repaired replay and their audit manifests as separate records.

Second prospective cohort and separate repairs. After the first repairs, we froze the frontend, adapter, acquisition harness and checkpoint binding on 36 further tasks (twelve scenarios), selected by the existing outcome-free hash order from the recorded unexposed remainder. This is same-benchmark scenario transfer, not an unseen ecosystem or a claim about model-training contamination. The run used 280 request attempts and saved 271 provider responses; nine tasks in three

entire scenarios received no response during a documented DNS incident. They were not retried or replaced. Thus response-conditional rates are not estimates under random missingness.

Table 37: Second cohort: frozen transfer versus post-outcome development. Task success and replay support are distinct. Repeated checks are not samples.

Endpoint	Frozen	Repaired
Planned tasks / scenarios	36 / 12	36 / 12
Native restoration equality	36/36	unchanged
Agent solved tasks	15/36	unchanged
Verified replay tasks	18/36	36/36
Solved and replay-verified	10/36	15/36
Matching initial/final vectors	36/36	72/72
Separate receiver repeat checks	360/360	720/720
Selected missing-block aborts	108/108	216/216

The frozen failures comprise six groups of three variants: two splitlines call sites, starred unpacking, nested grouping, division, and absent relation evidence. Among the 27 tasks with a provider response, 15 are replay-verified and 15 solved, with ten in their intersection. Three of the nine no-response tasks replay successfully; these are not successful Agent trajectories.

Separately versioned development adds bounded string and numeric operations, immediately consumed generator unpacking, structurally admitted grouping factories, a pinned month/year bridge, and reverse joins over complete authenticated partitions. Relations remain explicitly unordered; list projection is admitted only into order-insensitive membership observations. Native relation declarations, SQLite differential joins, resource rejection and proof removal are checked; 148 component/regression tests pass. All 18 prior passing tasks remain supported. Independent audits bind archive roots, native requirement vectors, source hashes and separate receiver outputs. These empirical bridges do not extend the closed-IR theorem. Neither the repaired 36/36 nor the implementation-check counts provide a fresh generalization or whole-Agent latency result.

T.10 Root-authenticated lazy full transport comparison

We separately freeze a zero-API cost comparison on all 35 restorable checkpoints of the first 36-task development cohort; the previously reported restoration failure is not replaced. The evaluator and source/receiver roots are unchanged. Each checkpoint has initial and final phases, three rotating-order arms, and ten queries per arm/phase. All 2100 queries match their native assessment and within-phase query sequence. Repeats and task variants are not independent statistical samples.

Compact full uses a single gzip block stream. Root-lazy full transports the complete per-block-compressed ZIP, but treats its index as an untrusted locator: only requested blocks are authenticated against trusted DAG roots. It does *not* require a receiver-side whole-file digest pass. Missing requested blocks abort; semantic absence comes from authenticated object shape, not index absence. Unrequested corruption may remain undetected. Duplicate index names are rejected. This removes the earlier indexed-full comparison's whole-file verification prerequisite without claiming that ZIP is an optimal lazy store.

Table 38: Selective/full ratios on 35 development checkpoints. First handoff charges selective sender execution, packet construction, receiver load and first query; full arms receive prebuilt packages for free. Warm queries exclude loading. These are local costs, not Agent lifecycle or measured network latency.

Full comparator	First handoff	Task wins	Warm query	Bytes
Compact gzip	0.157	35/35	0.985	0.003893
Root-lazy ZIP	0.114	35/35	0.998	0.001101

First-handoff ratios average the two phase ratios within each task and report their median; warm and byte ratios are medians across phases. Selective wins all 70 phase comparisons against both implementations. A post-hoc metricwise best-full comparator gives 0.157 first-handoff and 1.006

warm-query ratios; it is descriptive, not a single deployable baseline. Thus the observed advantage is initialization/transfer scope, not faster steady-state consumer execution.

The run retains all failures (none among eligible checkpoints). Separate audits check timing identities, 1406 source hashes, and all 140 transport packages (10.58 GB). Root provenance, faithful index issuance and interpreter assumptions remain shared premises. Local scheduling is not certified isolated; OS caches are not flushed. ZIP central-directory parsing and metadata checks still scale with archive size; a different on-disk index can have different loading costs. Source acquisition, common preparation, actual network transfer and Agent execution are outside the primary endpoint. Package bytes characterize these implementations, not an optimal compression bound. The results support a conditional local first-handoff benefit, not universal backend dominance.