Vouch: Preventing Semantic-Evidence Laundering with Structural Artifact Boundaries

Personal author copy. Presentation date: October 6, 2026

Matt Park
Studio Haze Inc.
South Korea

Abstract

Evidence produced by rule engines is often consumed outside the engine that produced it. Auditors inspect logs, users replay decisions, and downstream systems, increasingly AI-assisted, attach semantic claims to generated artifacts. This creates a semantic-evidence laundering risk. We use laundering to mean two confusion failures. In class confusion, external evidence is consumed as native checked evidence. In bound-context confusion, an artifact is reused for a context that contradicts its own byte/hash bindings. We treat full native-shaped fabrication separately as an issuer-binding and authenticity problem.

We present Vouch, a public artifact discipline for Lispex's checked CSK Profile. Vouch separates native differential receipts from external Bridge reports by artifact class, required fields, verifier entrypoints, and boundary declarations. Native receipts bind source, pinned input, intermediate artifacts, evaluation transcripts, diagnostics, and lowered-subset transcript agreement. Bridge reports carry external evidence without promoting it into native agreement. Offline verify checks artifact schemas, hashes, and boundaries, while rule-change replay reports corpus-wide behavioral changes over fixed inputs. Vouch does not claim semantic equivalence, independent witnessing, receipt authenticity, generation honesty, policy correctness, or legal fitness.

We evaluate Vouch on a welfare-shaped decision workload that required zero profile extensions and produced six decision flips over twelve replayed inputs. For laundering resistance, we evaluate class-confusion and bound-context fixtures over source, target, linked-artifact, and expected-context bindings, observing promoted-to-native = 0, bidirectional native/Bridge cross-submission failures, and Bridge read-side canonical rejection. Vouch's contribution is a reproducible, fail-closed trust boundary for semantic-evaluation evidence, not a proof system for correctness or authenticity.

CCS Concepts

• **Security and privacy → Software and application security**.

Keywords

evidence artifacts, trust boundaries, artifact-class separation, decision receipts, software supply chain

1 Introduction

Rule engines increasingly produce evidence that outlives the original execution context. A policy decision may be logged for audit. A pricing rule may be replayed after a rule change. A workflow engine may emit a report that downstream systems treat as proof that a semantic transformation occurred. These artifacts are useful only if their trust boundaries are visible. Without a machine-checkable boundary, an external assertion can be laundered into native verified evidence.

This paper studies two in-scope forms of semantic-evidence laundering: class confusion and bound-context confusion. In class confusion, external evidence is presented or consumed as native checked evidence. In bound-context confusion, an artifact is reused as evidence for a context that contradicts its own byte/hash bindings. We treat full native-shaped fabrication separately as an authenticity and issuer-binding problem (§2.4). The danger is not simply malformed JSON or a bad hash. The danger is a category error. A user may see a structured artifact and infer that it proves policy correctness, semantic equivalence, target-code generation, issuer authenticity, or independent verification, even when the artifact was never designed to prove any of those properties.

We present Vouch, a public artifact discipline for Lispex's checked CSK Profile. Vouch's central design is structural trust isolation. Native checked-subset evidence and external evidence are not two labels inside one receipt. They are different artifact classes. A native differential receipt has tag `csk.differential-receipt/v0` and contains source, input, canonical, graph, reference, Meaning Environment, comparison, diagnostics, and boundary sections. An external Bridge report has tag `vouch.bridge-report/v0` and contains a different required shape: profile, engine, subject, checks, linked artifacts, summary, boundary, and diagnostics. These artifacts are accepted by different verifier entrypoints. Changing a label or tag does not convert one into the other.

Vouch is deliberately narrow. It does not prove that a rule is fair, lawful, or correct. It does not prove semantic equivalence. It does not provide receipt authenticity, generation honesty, issuer binding, timestamping, or non-repudiation. These properties are outside the claim and belong to other layers, such as signatures, transparency logs, or supply-chain attestations. Vouch instead records what was evaluated under which public boundary, checks that artifact boundary offline, and reports rule-change effects over a committed input corpus.

This paper makes three contributions.

Structural trust isolation for semantic evidence. We identify evidence laundering as a security failure mode and introduce an artifact-class boundary that separates native checked-subset evidence from external Bridge evidence. Native and external artifacts are different structural types with different required fields, verifier entrypoints, and boundary declarations.

Reproducible decision evidence without overclaiming proof. We instantiate the boundary model in Lispex's checked CSK Profile. Native decision receipts bind source, pinned input, intermediate artifacts, transcripts, diagnostics, and lowered-subset transcript

agreement. Offline verify and rule-change replay provide artifact consistency and corpus-complete change disclosure, while explicitly excluding semantic equivalence, independent witnessing, policy correctness, and authenticity.

Evaluation of expressiveness, laundering resistance, and boundary honesty. We evaluate a welfare-shaped decision workload requiring zero profile extensions and detecting six decision flips across twelve replayed inputs. We also evaluate adversarial artifact-class fixtures and bound-context fixtures and observe promoted-to-native = 0. Finally, we show that receipt authenticity and generation honesty remain explicit non-goals enforced through artifact boundaries and public-claim checks.

The rest of the paper is organized as follows. §2 introduces the checked profile and evidence artifacts and states the threat model. §3 motivates two audit stories and two trust boundaries and situates laundering within a known security failure lineage. §4 defines the artifact taxonomy. §5 presents the design. §6 explains offline verify and replay. §7 describes implementation. §8 evaluates the system. §9 compares related work. §10 discusses broader impact. §11 states limitations. §12 concludes.

2 Background and Goals

2.1 Lispex and the CSK Profile

Lispex is a deterministic, macro-free, R7RS-shaped language. Vouch does not claim the full Lispex language. It operates on the CSK Profile, a checked source subset designed for deterministic decision artifacts. The profile's stated purpose is narrow: can a deterministic rule, evaluated over a pinned input datum, leave a portable artifact that can be re-checked without trusting a private service?

The checked profile includes core decision-programming forms such as quote, if, lowered cond, short-circuit and/or, begin, define, fixed-arity lambda and closures, parallel let, and top-level recursive define. It excludes features such as mutation, macros, continuations, unconstrained I/O, floats, and the full numeric tower unless a later profile explicitly admits them. Out-of-profile source constructs must produce profile escape diagnostics or lowering faults; they must not be silently approximated.

2.2 Pinned host input

A native decision artifact binds exactly one host input datum. The datum is immutable during evaluation and is bound to the distinguished profile name `input`. Source programs may reference `input`, but may not bind, define, or shadow it. The input is not injected into source, Canonical Core, or Meaning Graph bytes; instead, it is bound through the host API in the reference interpreter and Meaning Environment. Its canonical bytes are hashed under `csk/profile-input-hash/v0`.

This design makes a decision gallery different from a closed-form corpus. A closed-form corpus can show that programs produce fixed outputs. A decision gallery binds rules to concrete input records and can therefore support replay over a committed input corpus.

2.3 Native evidence pipeline

The native evidence pipeline connects several public contracts, each of which follows the same attests/excludes discipline: it names a narrow property and explicitly disclaims stronger ones. Canonical Core defines deterministic normalized Core bytes and a domain-separated core hash; it is a syntactic-normalization contract, not a semantic proof. Meaning Graph defines public graph shape and Meaning Law validity rules for the checked subset; Core-to-Meaning-Graph lowering defines a deterministic one-way transform from the admitted Core subset into Meaning Graph JSON bytes. None of these define execution semantics, semantic equivalence, independent witnessing, or target-code generation. Meaning Environment evaluates canonical Meaning Graph bytes as an internal second path, validating Meaning Law and emitting a transcript; its report is not a receipt and does not prove source-language semantic equivalence. Differential Receipt records lowered-subset transcript agreement between the reference interpreter and the Meaning Environment; it is not a semantic equivalence proof and not an independent witness. Section 4 summarizes the artifact boundaries we rely on.

2.4 Threat Model: Semantic-Evidence Laundering

The attacker may author arbitrary artifact bytes, relabel artifacts, graft fields across artifact classes, and present a genuine artifact together with a contradictory source, target, linked artifact, or expected context. We assume the attacker cannot find SHA-256 collisions; issuer identity, key compromise, and generation honesty are outside the model and belong to the authenticity layer that Vouch explicitly excludes. Within this model, we use semantic-evidence laundering for two in-scope confusion failures.

First, **class confusion**: an external Bridge report is presented or consumed as native checked-subset evidence. This is the attack addressed by artifact-class separation: native receipts and Bridge reports have different tags, required fields, boundaries, and verifier entrypoints.

Second, **bound-context confusion**: an artifact is reused as evidence for a context that contradicts the artifact's own byte/hash bindings. For Bridge reports, the bound context includes source bytes, target bytes, linked artifacts, declared route, profile, capability identifiers, and the consumer's expected context when one is supplied.

We do not use laundering to include full native-shaped **fabrication**. If an attacker recomputes all byte bindings and authors a fresh self-consistent artifact for a new context, the attack has crossed from confusion into fabrication. Vouch does not claim to solve fabrication. It keeps receipt authenticity, generation honesty, issuer binding, timestamping, and non-repudiation outside the claim, while providing deterministic bytes and domain-separated hashes that a separate issuer-binding layer can sign or log.

(The native input datum is separately bound under `csk/profile-input-hash/v0`; input-corpus context reuse is not covered by this paper's bound-context fixtures and is left to future work.)

2.5 Goals and non-goals

Vouch's goal is to make semantic-evaluation evidence reproducible, portable, and boundary-aware. It supports offline artifact checking and rule-change replay. It does not prove that a policy is correct, legal, fair, or morally justified. It does not prove receipt authenticity

or generation honesty. It does not provide independent semantic verification. It records what was evaluated and which public boundaries were checked.

3 Motivation: Two Audit Stories, Two Trust Boundaries

Modern rule systems often need to answer two different audit questions. The first is internal and reproducible: given this rule, this input, and this checked execution boundary, what decision was produced, and what would change if the rule changed? The second is external and adversarial: if another engine emits evidence that resembles Vouch evidence, how do we prevent that external assertion from being mistaken for native checked-subset evidence? Vouch is designed around the claim that these are different audit stories and therefore require different trust boundaries.

3.1 A known failure class in a new artifact domain

This failure mode is not unique to Vouch artifacts. Security systems have repeatedly failed when the structure that was verified differs from the structure later consumed. JWT algorithm-confusion attacks and the later Best Current Practices for JWTs [13] show how mutable labels inside a shared token envelope can lead to confusion between token kinds; the mitigation includes explicit typing and different validation rules for different JWT kinds. XML Signature Wrapping attacks in SAML [15] similarly exploit gaps between signed XML elements and the elements consumed by application logic. Vouch targets the same failure class in a different artifact domain: semantic-evaluation evidence. Its response is to lift anti-confusion discipline from labels to structure: domain-separated hashes, artifact-class separation, and verifier-entrypoint separation.

The pressure on this boundary grows with AI-assisted development and agentic build pipelines. Such systems increasingly emit machine-readable claims — tests passed, conversion succeeded, policy preserved, semantically equivalent — that downstream tools consume automatically. Vouch treats such claims as external evidence unless they arrive as native checked-subset artifacts: the more claims are produced and consumed without a human in the loop, the more important it becomes to keep native checked evidence structurally separate from external assertion.

3.2 Native decision receipts

The native story begins with a deterministic rule in the CSK Profile. A host supplies one immutable input datum, the evaluator runs the rule over it, and the system emits a portable decision artifact binding rule source, input, engine version, transcript, diagnostics, and checked-profile boundary; verify re-checks it offline, and replay re-runs a committed decision corpus across rule changes. As in §2.2, the input is bound through the host API and recorded by its canonical hash rather than injected into source or graph bytes, which is what makes this a decision gallery rather than a closed-form corpus.

The native receipt therefore supports a useful but bounded audit workflow: what was evaluated, over which canonical input, under which checked profile boundary, with which transcript, and — via replay over a fixed corpus — which committed decisions change under a rule revision. This is not a claim that the rule is fair, lawful, optimal, or semantically equivalent to another implementation. The Profile boundary explicitly excludes policy correctness, whole-language coverage, semantic equivalence between implementations, regulatory or audit fitness, and overbroad independence claims.

3.3 External evidence and evidence laundering

The second story is different and adversarial. An external engine may emit a Vouch-compatible evidence shape that is not a native differential receipt; the security problem is that such an external assertion could be presented, relabeled, or structurally confused as native checked-subset evidence. A label-based design would be fragile: a mutable provenance field could be stripped or rewritten. Vouch instead uses artifact-class separation. Native evidence is `csk.differential-receipt/v0`; external Bridge evidence is `vouch.bridge-report/v0`. They have different tags, required fields, verifier entrypoints, and public boundaries, so a native verifier rejects a Bridge report not because a signature failed but because the artifact class does not match: native fields are missing and Bridge-only fields are unknown.

This separation gives Vouch a precise, narrower-but-stronger audit value. Native receipts and replay provide reproducibility and corpus-complete rule-change disclosure; Bridge reports carry external evidence without allowing promotion into native evidence. Vouch does not assert that every artifact is authentic, that every rule is correct, or that an external transformation claim is true. It asserts that the artifact's trust boundary is explicit, machine-checkable, and fail-closed: native evidence stays native, external evidence stays external, and claims outside the boundary remain outside it.

4 Artifact Taxonomy

Vouch's trust boundary is defined by artifact class. This section defines the three artifact classes used in the paper: native differential receipts, external Bridge reports, and Meaning Environment reports. The distinction is essential for the security argument in §8: artifact-class spoofing fails because native and external artifacts are different structural types, not because they carry different labels inside a shared envelope.

4.1 Native differential receipts

A native differential receipt has tag `csk.differential-receipt/v0`. It records one source input, one optional CSK Profile host input datum, the reference interpreter transcript, the Meaning Graph lowering when available, the Meaning Environment transcript when available, and the byte-level comparison between the two transcript payloads. The contract explicitly says that this artifact is not a semantic equivalence proof and not an independent witness.

A native differential receipt has the following required top-level fields: `differential_receipt`, `engine`, `source`, `input`, `canonical`, `graph`, `reference`, `meaning_env`, `comparison`, `diagnostics`, `boundary`.

The comparison object records status, reason, substrate, and first divergence. Its substrate is `shared-rust-reference`: the reference path and Meaning Environment path share the Rust reader, normalizer, Core AST, value representation, numeric tower, writers, hasher, and process. They do not share evaluation engines, but agreement

can still be produced by a shared component bug. Thus the receipt attests lowered-subset transcript agreement inside the shared Rust reference substrate only. Its boundary attests source bytes, profile input hash binding, Canonical Core bytes, Meaning Graph hash binding, the two transcript byte streams, and their agreement; it excludes semantic equivalence, independent witnessing, substrate independence, error agreement, input provenance, full CSKernel coverage, target-code generation, and private implementation detail. The receipt boundary also lists `topaz-reporting` among its excludes; per the contracts, the artifact does not attest Topaz reporting, Topaz independence, or external witnessing.

4.2 External Bridge reports

A Bridge report has tag `vouch.bridge-report/v0`. Per the public Bridge contract, it has the following required top-level fields: `bridge_report`, `profile`, `engine`, `subject`, `checks`, `linked_artifacts`, `summary`, `boundary`, `diagnostics`.

Unlike a native differential receipt, a Bridge report is not a transcript-agreement receipt. It does not carry the native receipt's source, input, canonical, graph, reference, meaning_env, or comparison structure. The Bridge contract fixes these nine top-level fields in order and admits no other top-level field, and it routes the report through the `lispex verify-bridge` entrypoint rather than the native `lispex verify` entrypoint [17, §§7, 3, 12].

The security role of the Bridge report is boundary preservation. It can establish that an external evidence report has the expected structure, linked-artifact references, summary, diagnostics, and boundary declarations, but it cannot establish that the external engine honestly produced the evidence, that a transformation occurred, that the result is semantically equivalent to a source, or that the evidence is a native receipt.

A Bridge report is not only an adversarial rejection target: the contract includes a positive conversion-evidence fixture (a TypeScript source, a Python target, an external engine identity, a declared route, capability identifiers, gate results, and linked proof-artifact hashes), checked by running `lispex verify-bridge` with source and target file arguments and the Bridge report path. The success result is a `vouch.bridge-verify-report/v0`, still external evidence: the v0 boundary lists target-code-correctness, semantic-equivalence, generation-honesty, and issuer-binding among its excludes [17, §§14, 9]. This is why the paper uses no mutable provenance-label field: there is no shared envelope carrying such a label; the distinction is structural between `csk.differential-receipt/v0` and `vouch.bridge-report/v0`.

4.3 Meaning Environment reports

A Meaning Environment report has tag `csk.meaning-env-report/v0`. It is the report emitted by the bounded evaluator for canonical Meaning Graph bytes. Its role is to make a Meaning Graph executable without routing back through the Lispex source interpreter. It is used as an internal second path for comparing lowered-subset transcripts, but it is not itself a receipt and does not prove source-language semantic equivalence.

Its top-level fields are `meaning_env_report`, `engine`, `graph`, optional `input`, `status`, `law_errors`, `trace`, `transcript`, `result`, `fault`, `steps`, and `boundary`. Its public comparison surface is the transcript. The boundary attests Meaning Law validation, bounded deterministic evaluation, input binding when supplied, transcript bytes, and graph hash binding; it excludes semantic equivalence, differential receipt status, independent witnessing, source lowering, and target-code generation. The contract states directly: this is not a receipt; it binds to graph bytes but does not attest source bytes or prove interpreter agreement.

Artifact-class summary. Native differential receipts are lowered-subset transcript-agreement artifacts; Bridge reports are structural boundaries for external evidence; Meaning Environment reports are bounded graph-evaluation reports and not receipts.

4.4 Taxonomy as a security boundary

The taxonomy is not bookkeeping. It is the security boundary.

If an attacker submits a Bridge report to the native verifier, changing only the top-level tag cannot make it native. The artifact still has Bridge-only fields such as `bridge_report`, `profile`, `subject`, `checks`, `linked_artifacts`, and `summary`, while it lacks native fields such as `source`, `input`, `canonical`, `graph`, `reference`, `meaning_env`, and `comparison`. This mismatch is what §8 evaluates: the native verifier rejects such artifacts with tag mismatch, unknown top-level fields, missing native fields, engine mismatch, and boundary mismatch diagnostics, and no external claim is promoted to native evidence.

The taxonomy also clarifies what Vouch does not claim. A fully native-shaped forged artifact is outside the authenticity claim. The verifier may check internal consistency, but without a separate issuer-binding, signature, timestamping, or transparency-log layer, Vouch does not prove honest generation. This is not a bypass of the taxonomy; it is a documented non-goal encoded in artifact boundaries and public-claim checks.

4.5 Relationship to the rest of the system

The three artifact classes sit on top of the lower CSK contracts (Canonical Core, Meaning Graph, and lowering), each of which excludes runtime values, stdout, execution semantics, semantic equivalence, independent witnessing, and target-code generation, as summarized above.

Thus, Vouch's evidence pipeline is deliberately layered. Lower artifacts define bytes and structure. The Meaning Environment evaluates graph artifacts. Native differential receipts compare transcript surfaces. Bridge reports keep external evidence structurally separate. No layer is allowed to silently inherit a stronger claim from another layer.

5 Design: Structural Trust Isolation

5.1 One anti-confusion principle at four levels

Vouch applies one anti-confusion principle at four levels.

First, hashes are separated by domain strings. The same bytes cannot silently change meaning by being consumed under a different hash domain.

Second, artifacts are separated by class. A native differential receipt and a Bridge report are different structural types with different required fields and different boundary blocks.

Third, verifiers are separated by entrypoint. A Bridge verifier result is not a native transcript-agreement result, and a native receipt verifier does not consume Bridge reports as native evidence.

Fourth, context checks are relative to declared intent. An expected-context manifest is not a trusted authority; it is the consumer's declaration of the context they intend to verify. Vouch checks whether the artifact's bindings match that declaration, not whether the declaration is correct. This keeps context confirmation deterministic while keeping context authenticity outside the claim.

In short: domain separation for hashes, class separation for artifacts, entrypoint separation for verifiers, and intent binding for context. A source hash cannot become a target hash by reusing bytes under another domain, a Bridge report cannot become a native receipt by changing a label, a Bridge result cannot be consumed as native agreement, and an artifact cannot be evidence for a context it does not bind. This is consistent with the existing hash-preimage practice, `<domain>\0<payload>`.

5.2 No silent claim inheritance

Vouch's negative design principle is that no layer silently inherits a stronger claim. Each contract names a narrow attestation and explicit exclusions: lower layers define bytes and structure, the Meaning Environment evaluates graph artifacts, receipts compare transcript surfaces, and Bridge reports carry external evidence without promoting it into native evidence.

5.3 Checked profile, graph, and environment

The native workflow starts in the CSK Profile and its pinned-input model (§2.1–2.2). One immutable host datum is bound as `input`, never injected into source, Canonical Core, or graph bytes, and recorded by `csk/profile-input-hash/v0`; this is what makes replay over a committed input corpus meaningful. The source pipeline lowers the checked subset into a structural Meaning Graph and evaluates it in the Meaning Environment. The native receipt then compares reference and Meaning Environment transcripts: matching bytes yield `agree`, differing bytes yield `disagree`, and prerequisite failures yield `not-comparable`.

5.4 Same-origin second path, not independent witnessing

The Meaning Environment is an internal second path, not an independent implementation witness. The differential receipt contract discloses that the reference path and Meaning Environment path share the Rust reader, normalizer, Core AST, value representation, numeric tower, canonical datum writer, graph JSON writer, graph hasher, and process. They do not share evaluation engines, but a shared component bug can still produce agreement.

Therefore, Vouch's native agreement claim is intentionally narrow: lowered-subset transcript agreement inside a disclosed shared substrate. It is not semantic equivalence, substrate independence, independent witnessing, target-code generation, or proof of policy correctness. The receipt boundary explicitly excludes those stronger claims. This is the paper's core honesty discipline. The system is designed to make useful artifacts, but also to prevent those artifacts from being read as stronger proofs.

5.5 Bridge reports and external isolation

External evidence is isolated by artifact class: native receipts and Bridge reports carry different tags and different required fields. The contract declares the two classes disjoint — distinct tags, distinct verifier entrypoints — and states that changing a tag or adding native-looking fields does not promote external evidence into native transcript agreement [17, §§12, 3]. This is the security boundary, not a cosmetic one: a Bridge report submitted to the native verifier lacks required native fields and carries Bridge-only fields, and vice versa. The Bridge boundary is also limited — it does not prove the external engine's transformation truth, generation honesty, issuer binding, timestamping, non-repudiation, semantic equivalence, or target-code generation — and those must be supplied, if needed, by a separate signing, timestamping, or transparency layer.

5.6 Summary

The design can be summarized as follows. Lower artifacts define bytes and structure. Meaning Environment evaluates graph artifacts. Native differential receipts compare transcript surfaces. Bridge reports keep external evidence structurally separate. No layer silently inherits a stronger claim from another layer.

This design principle is what §8 evaluates: realistic decision rules run under the minimal checked profile; Bridge-shaped artifacts do not become native receipts; and authenticity remains outside the claim rather than being silently implied.

6 Offline Verify and Rule-Change Replay

6.1 Verification goal

Offline verification in Vouch is artifact checking, not independent semantic verification. The verifier checks whether committed artifacts satisfy the public contracts — schema shape, hash domains, canonical bytes, boundary blocks, known checks, expected report structure, and claim boundaries. It does not re-run source as a semantic authority, prove policy correctness or receipt authenticity, bind an issuer, timestamp artifacts, or establish non-repudiation. This follows from the native receipt boundary, which attests the byte streams and their agreement while excluding semantic equivalence, independent witnessing, substrate independence, input provenance, and target-code generation.

6.2 Verify report contract

The verify command emits `csk.verify-report/v0` with exit codes 0 (pass), 1 (fail), 2 (usage). The verifier is a same-origin JavaScript artifact checker: it does not execute, link, or inspect the Rust implementation, only checks artifacts against public contracts and expected shapes. It is written against the public contracts only — a same-origin public artifact checker, not a third-party verifier or an independent witness.

6.3 Canonical byte acceptance (Bridge)

For Bridge artifacts, v1.3.11 adds read-side canonical byte acceptance. Vouch treats artifact bytes, not abstract JSON values, as the evidence object. The Bridge verifier may parse JSON only to recover a candidate value. Before it consumes any semantic field, it re-serializes that value with the `csk.artifact-json/v0`

writer and byte-compares the result with the submitted bytes. If the bytes differ, verification fails before semantic interpretation with `non-canonical-artifact-json`. When an external expected report hash is provided, the hash is checked over the submitted bytes, not over a repaired or normalized serialization.

Bridge validation is ordered as canonical bytes first, recursive closed-world schema second, and semantic boundary checks third. No field from a non-canonical artifact is trusted.

In this implementation, duplicate-key cases fail as `non-canonical-artifact-json` before semantic interpretation: a duplicate-key document whose parsed value re-serializes to different bytes fails the canonical gate. The verifier does not emit a separate duplicate-key diagnostic because the current reader is `JSON.parse`-based; implementations with token-level readers may report duplicate-key directly.

This canonical acceptance rule is Bridge-specific. Native receipt read-side canonical enforcement is not yet unified; see §11.

6.4 Expected context

For Bridge reports, context verification is optional but explicit, and it uses two distinct consumer-side inputs. Byte bindings are checked by supplying the actual files: the `--source`, `--target`, and `--linked` options re-hash the supplied bytes and compare them against the report's declared hashes [17, §§3, 7]. Subject-context intent is checked by an expected-context manifest, `vouch.bridge-context-manifest/v0`, which declares only the profile, subject kind, case id, route id, checked profile, and capability ids the consumer intends to inspect [17, §8]. The manifest is not an authority about what the correct context should be; it is the consumer's stated intent. Vouch checks whether the report matches that intent. A mismatch is reported with a field-specific, closed diagnostic; a fully recomputed artifact for the new context is treated as a fresh artifact, whose authenticity remains outside the claim.

6.5 Replay report contract

Replay emits `csk.replay-report/v0`. Exit codes are pinned as: 0 unchanged, 1 changed, 2 usage.

Replay compares a committed corpus under a baseline rule and a changed rule. The fixed variables are the input corpus, checked profile, artifact schema, and replay report structure. The intended variable is the rule revision. Time, randomness, ambient environment, and unconstrained I/O are outside the checked profile, so replay results are deterministic under the committed artifacts.

The roadmap frames replay as the corpus regression workflow built on receipts: rule-change replay comes before engine-upgrade replay because rule changes are the first realistic single-user pain.

6.6 What replay proves

Replay proves neither policy correctness nor independent semantic truth. Given a fixed input corpus, baseline rule, changed rule, fixed checked profile, and deterministic artifact contracts, it reports which cases changed at the byte layer, which projected decisions changed when projection is available, which cases failed or became not-comparable, and which boundary each result belongs to. This gives auditors corpus-complete change disclosure — "which previously committed decisions would change under this new rule?" — answered deterministically. It does not answer whether the new rule is fair, legal, or better.

6.7 Invalid input as a first-class decision

The welfare replay evaluation includes malformed application-level records that are still valid Lispex profile data. Those cases are not pipeline-level input failures. The pipeline executes normally: input is bound, transcripts are produced, fault_class is null, and replay status is agree. The rule itself returns `(decision invalid-input ...)`.

The paper uses this distinction: application-level malformed values or policy tables that were still valid Lispex profile data were classified as `(decision invalid-input ...)`; profile-level parse or input-domain failures remain input-error receipt events.

This distinction matters because fail-closed behavior is not merely an exception path. It can also be a first-class business decision inside the deterministic rule logic.

6.8 Verify/replay boundary summary

Verify checks artifact schema, hash domains, linked artifacts, boundary blocks, expected report structure, and claim boundaries; it does not check source truth, generation honesty, receipt authenticity, issuer binding, timestamping, non-repudiation, or external independent verification. Replay checks deterministic behavior changes over a fixed committed corpus and changed rule; it does not check fairness, legality, policy correctness, semantic equivalence, or external transformation truth.

The security value is not independence. The value is reproducibility, fail-closed artifact checking, and corpus-complete change disclosure.

7 Implementation

7.1 Artifact JSON, hashes, and release bytes

Vouch artifacts use per-class deterministic JSON bytes: stable field order, UTF-8, LF newlines, path-neutral JSON, no host metadata in hashable reports, and no self-hashes. Bridge read-side canonical enforcement exposed Rust/JS writer-format divergence; Bridge acceptance is pinned to the JS writer, while native artifacts remain produced by the Rust writer. Cross-writer unification is deferred rather than overclaimed.

All public hashes are domain-separated: `<domain>\0<payload>` with SHA-256 and lowercase hex. Bridge context checks use `vouch/external-source-hash/v0`, `vouch/external-target-hash/v0`, and `vouch/linked-artifact-hash/v0`. Reports never embed their own hash; expected hashes live in release or review manifests. Layer A masks volatile commit fields for repository goldens, while Layer B uses exact unmasked bytes from a clean public or anonymized release bundle.

7.2 Native pipeline implementation

The native pipeline runs source and optional profile input through reader/normalizer, Canonical Core, CSK Profile lowering, Meaning Graph JSON, reference and Meaning Environment transcripts, comparison status, and finally a differential receipt. The reference

interpreter remains the full-language authority; the Meaning Environment is the bounded evaluator for the lowered CSK Profile subset. Agreement is byte-level lowered-subset transcript agreement, not semantic equivalence.

7.3 Verify, replay, and Bridge implementation

The npm verifier path provides the portable public checking surface. It emits verify and replay reports and checks schema, hashes, boundaries, known checks, and expected report structure. It is same-origin and public-contract based; it does not execute or link the Rust evaluator.

The Bridge implementation adds a separate verifier entrypoint, `lispex verify-bridge`, governed by a public Bridge contract [17]. The Bridge verifier checks `vouch.bridge-report/v0` shape and boundary declarations. Consistent with the contract's stated scope, it does not run the external engine, treat Bridge reports as native receipts, judge target-code correctness, or prove whole-language semantic equivalence [17, §§1, 9].

This implementation division mirrors the paper's taxonomy: native differential receipts, Bridge reports, and Meaning Environment reports are different artifact classes with different entrypoints.

7.4 Detection infrastructure

The implementation includes detection fixtures and mutation drills. These are not performance tests; they test whether artifact-boundary claims fail closed.

The mutation and adversarial fixture infrastructure covers artifact byte mutation, hash mutation, field insertion/deletion/reordering, optional-null misuse, path-neutrality violations, unknown trust-field smuggling, golden hand editing, claims-registry boundary weakening, Layer A/B masking mistakes, Bridge-to-native artifact-class spoofing, and authenticity overclaim attempts. The v1.3.11 slice adds canonical-acceptance tests (compact JSON, CRLF JSON, duplicate-key collapse), recursive closed-world tests for nested unknown fields, and bound-context mismatch tests over source, target, linked-artifact, and expected-context bindings.

The key adversarial metric is not the number of rejections. It is whether any external claim is promoted to native evidence. In the implemented spoofing evaluation, promoted-to-native = 0.

7.5 Release and artifact evaluation support

The implementation also includes release-audit and reviewer-support infrastructure — deterministic report hashes, release manifests, public-claim checks, a cold-reviewer drill protocol — so that an evaluator can download the bundle, run verify/replay, and compare report hashes without a private service. This demonstrates reproducibility and boundary checking; it does not add receipt authenticity, third-party independence, or semantic proof.

8 Evaluation

We evaluate Vouch against three questions, each corresponding to one part of the paper's claim.

RQ1: Expressiveness under a minimal checked profile. Can the checked CSK Profile express a realistic decision workload without widening the profile?

RQ2: Laundering resistance. Can external or misused evidence be promoted into, or reused as, native checked-subset evidence? We split this into RQ2a (class-confusion resistance) and RQ2b (bound-context-confusion resistance).

RQ3: Boundary honesty. When Vouch cannot establish authenticity or generation honesty, does the system make that non-goal explicit rather than silently overclaiming it?

These questions intentionally avoid claims outside the system boundary. We do not evaluate policy correctness, legal validity, semantic equivalence to an external engine, or third-party independent verification. The Profile contract likewise limits decision artifacts to source/input/engine/profile/transcript/diagnostic boundaries and excludes policy correctness, semantic equivalence, regulatory fitness, and overbroad independence claims.

8.1 RQ1: Welfare-shaped rule replay

Our first evaluation asks whether the checked profile is expressive enough for realistic decision logic without adding language features for the benchmark. We implemented a welfare-shaped rule set with marginal bands, dependent adjustments, income tests, period-specific parameters, and multi-stage table lookup — not a legally correct benefits program, but a realistic structured decision workload. It required zero CSK Profile extensions, using the existing profile and host-input model. The representative rule file implements three marginal bands and three dependent categories in 103 lines of Lispex source.

We then ran rule-change replay on a fixed corpus of 12 inputs, varying only the rule. Replay detected 6 decision flips out of 12 cases. Application-level malformed values that were still valid Lispex profile data were classified as `(decision invalid-input ...)` — the pipeline executed normally (input bound, transcripts produced, fault_class null, replay status agree, the rule itself returning the invalid-input decision) — whereas profile-level parse or input-domain failures remain input-error receipt events.

This supports two claims: the minimal checked profile suffices for this class of realistic workload, and rule-change replay provides corpus-complete change disclosure. The result does not say the new rule is better, fairer, or legally correct; it says exactly which committed decisions changed under a deterministic rule revision.

Beyond the welfare workload, a transcription-fidelity ledger maps five rules from JSON Logic, Cedar, and OpenFisca Core into the checked profile under declared input/output maps. Across 676 mapped cases, the Lispex transcription agreed with the declared oracle output in all 676, including 544 cases checked against an executable OpenFisca Core oracle; 160 of the 676 are invalid-input decisions, counted separately. Agreement here is fidelity to a declared mapping, not semantic equivalence or policy correctness. A profile-growth review over this ledger recommended zero L2 or L3 admissions, leaving CSK Profile v0 unchanged.

8.2 RQ2: Laundering resistance

RQ2 has two parts because Vouch defends against two mechanisms. Class separation is structural: tags, required fields, boundary blocks, and verifier entrypoints. Context binding is byte-level and cryptographic: domain-separated hashes bind source, target, and linked

artifacts, and an optional expected-context manifest states the consumer's intended context.

8.2.1 RQ2a: Class-confusion resistance. The fixture suite is not the main security argument. The argument is the artifact-class disjointness invariant: native and Bridge artifacts require different tags, different top-level structures, and different verifier entrypoints, and both verifiers reject unknown top-level fields.

We confirm the implemented schema disjointness with bidirectional cross-submission and adversarial regression fixtures. The repository also records a fixture-level dual-accept invariant: zero artifacts are accepted by both verifiers across a seven-artifact cross-submission set. Generated disjointness testing covers 32 deterministic native/Bridge candidates with `dual_accept` = 0, and an artifact-grammar gate (26 cases across 27 entrypoint checks; 5 positive accepts, 22 rejects) and a mutation gate (15 active mutations, zero wrong failure classes) reject malformed, hybrid, and boundary-weakening variants. Randomized property-based testing remains a hardening step.

As a positive control, valid native receipts in the committed corpus pass the native verifier as native evidence, while the positive Bridge conversion fixture (§4.2) passes only as external evidence, recorded `accepted-as-external`.

Bridge → native. A valid Bridge report submitted to the native verifier exits with code 1 and produces 24 diagnostics, including unknown Bridge-only top-level fields, tag-mismatch, missing native fields, engine mismatch, and boundary mismatch.

Native → Bridge. The reverse direction has two fail-closed paths after v1.3.11. A native receipt whose submitted bytes are not byte-identical to the Bridge checker's canonical JS-writer serialization fails before semantic interpretation as `non-canonical-artifact-json`. In the measured refund-window case, rejection stops at this canonical gate. A roundtrip-identical native receipt reaches the class-field layer and is rejected with 34 diagnostics, including unknown native-only fields, missing Bridge fields, tag mismatch, and profile, subject, summary, and boundary mismatch diagnostics (the full 34-diagnostic listing is included in the artifact bundle).

Stability of A.1–A.12. The A.1–A.12 failure classes remain unchanged from the pre-hardening run because those fixtures are canonical-by-construction: they pass the new canonical gate and reach the same field-level checks. Formatting-layer attacks are covered by separate hardening tests, including compact JSON, CRLF JSON, and duplicate-key collapse, all of which fail as `non-canonical-artifact-json`.

Duplicate-key shadowing. The result is not defeated by duplicate-key shadowing: `JSON.parse` can shadow a value for an existing top-level key but cannot remove top-level key names from the parsed root, so Bridge-only keys (`bridge_report`, `profile`, `subject`, `checks`, `linked_artifacts`, `summary`) remain visible and trigger unknown-field diagnostics. Nested shadowing is covered by the recursive closed-world validator for Bridge; native nested read-side ambiguity remains a §11 limitation.

Fixture group. A.1–A.12 are not only Bridge-to-native spoofing: they cover relabeling, grafting, boundary weakening, hash-domain substitution, trust-field smuggling, and one positive external-acceptance case (recorded `accepted-as-external`). Across the group, promoted-to-native = 0; per-fixture attacks, gates, and failure classes (e.g., A.4 → `unknown-top-level-field:bridge_report`; A.7 → `source-hash-domain`) are recorded, with `group_a_promoted_to_native` = 0, in the artifact bundle. No path lets external evidence become native evidence: the system structurally rejects artifact-class laundering, not merely malformed JSON.

8.2.2 RQ2b: Bound-context-confusion resistance. RQ2b tests whether a Bridge report can be reused against a context that contradicts its own bindings. We evaluate source, target, linked-artifact, and expected-context mismatch fixtures.

The source and target cases are self-contained. The checker rehashes the supplied source and target bytes under `vouch/external-source-hash/v0` and `vouch/external-target-hash/v0`; mismatches fail as source or target hash mismatches. Linked artifacts are checked similarly under `vouch/linked-artifact-hash/v0` [17, §4]. The contract's worked example combines all four consumer-side checks (`--source`, `--target`, `--linked`, `--expect-context`) on one report [17, §14].

For route/profile/capability context, the expected-context manifest is not a trusted authority. It is the consumer's declaration of intended context. Vouch checks whether the report's declared context matches that declaration. If it does not, the artifact is not evidence for the consumer's intended context. This confirms intent binding without claiming context authenticity. The bound-context fixtures cover wrong source bytes (`source-hash-mismatch`), wrong target bytes (`target-hash-mismatch`), wrong linked artifact bytes (`linked-artifact-hash-mismatch`), and six expected-context mismatches: `context-profile-mismatch`, `context-subject-kind-mismatch`, `context-case-id-mismatch`, `context-route-id-mismatch`, `context-checked-profile-mismatch`, and `context-capability-ids-mismatch`.

If an attacker recomputes every binding for a new context, Vouch no longer treats the attack as context confusion. The attacker has authored a fresh self-consistent artifact for that context. Vouch keeps authenticity of that artifact outside the claim and leaves issuer binding to a separate trust layer.

8.3 RQ3: Authenticity boundary honesty

The third evaluation is deliberately not a rejection test. A verifier that checks self-consistency cannot, by itself, prove that an artifact was honestly generated, issued by a particular party, or produced at a particular time; a fully forged native-shaped artifact can be internally self-consistent. If Vouch claimed to detect such a forgery, that would be an overclaim. We instead evaluate whether Vouch makes this non-goal explicit and machine-checkable.

A fully native-shaped, internally self-consistent artifact is not a class-confusion success — it is fabrication, which is more demanding than writing a native-looking JSON object: the attacker must satisfy native schema shape, hash-chain consistency, boundary arrays, engine and comparison fields, and transcript-agreement structure. Under the current native reader the consistency required is parsed-value-level; the residual byte-level exposure is a §11 limitation. If an artifact is merely relabeled, grafted, or reused against a contradictory context, Vouch rejects or exposes the mismatch; if all bindings are recomputed, the attack has become fabrication, which Vouch keeps outside the claim.

The system carries a uniform authenticity boundary across native, verify, and Bridge classes: their `boundary.excludes` include receipt-authenticity, generation-honesty, issuer-binding, timestamping, and non-repudiation, and public-claim checks reject promotional claims implying authenticity or non-repudiation across the public-copy surfaces. A fully native-shaped forged artifact is thus not a bypass: Vouch checks structure, hashes, transcript agreement, replay differences, and declared boundaries, and keeps authenticity outside the claim.

8.4 Mapping evaluations to claims

Each element of the main claim maps to a specific evaluation and to an explicit non-result. RQ1 shows the checked profile expresses realistic decision logic (0 profile extensions; 676/676 declared-map agreement across three external rule systems) and that replay gives corpus-complete change disclosure (6 flips over 12 inputs) — but not legal correctness or fairness. RQ2a shows external evidence is structurally isolated from native (disjointness invariant, bidirectional cross-submission, promoted-to-native = 0) — but not receipt authenticity or issuer identity. RQ2b shows a byte-bound artifact cannot be reused against a contradictory context (source/target/linked/expected-context mismatches fail closed) — but not context authenticity, and not a fully recomputed fresh artifact. RQ3 shows boundary exclusions are enforced rather than rhetorical (authenticity terms appear in `excludes` and are denied in public claims) — but not cryptographic signing or timestamping. Underlying all of these, the disclosed shared substrate means transcript agreement is honest but not semantic equivalence or third-party verification.

Together, the evaluations establish the paper's narrowed security claim: Vouch is not a proof system for policy correctness, external transformation truth, or receipt authenticity. It is a boundary enforcement system for checked-subset decision evidence and external Bridge reports. It makes native evidence reproducible, makes rule-change effects visible over a fixed corpus, structurally prevents external reports from being promoted into native receipts, exposes bound-context reuse when the consumer checks against supplied bytes or a declared context, and explicitly marks authenticity as a separate trust layer.

9 Related Work

Vouch sits at the intersection of three bodies of work: policy and rule systems, provenance and attestation systems, and differential testing. None is a direct replacement for Vouch. Policy systems decide rules but usually do not produce portable, service-independent decision receipts with structural isolation for external evidence. Provenance systems authenticate artifacts and issuers but do not evaluate or replay decision-rule semantics. Differential testing compares implementations, but does not by itself create a user-facing trust-boundary artifact. (The security-failure lineage for evidence laundering itself — JWT algorithm confusion, SAML XML Signature Wrapping — is discussed in §3.1.)

9.1 Policy and rule systems

Cedar and Cedar Analysis. Cedar is the strongest adjacent policy system. It is a purpose-built authorization policy language with a formal semantics, a Rust implementation, and a Lean model used to verify properties of the language and its tooling [2]. The more direct comparison is Cedar Analysis [1], which translates Cedar policies into SMT formulas and uses a Lean-implemented symbolic compiler with soundness and completeness guarantees. The toolkit can compare policy sets, identify equivalent or more/less permissive policies, detect whether a change grants new permissions, and reason about all possible access scenarios rather than sample test cases.

This is stronger than Vouch in one dimension: Cedar Analysis can provide static, symbolic guarantees over the authorization policy space. Vouch does not attempt that. Its replay is not a proof over all possible inputs. Instead, Vouch targets a different audit question: given a committed corpus of concrete decision inputs and a changed rule, replay deterministically reports every behavioral change in that corpus, while the underlying artifacts remain portable and checkable offline. Cedar Analysis asks whether two authorization policy sets are equivalent or whether one is more permissive; Vouch asks what happened to these recorded decisions, whether the artifacts remain internally consistent, and whether external evidence was kept outside the native trust boundary. "Policy-change impact analysis" alone is no longer a differentiator; Vouch's position is narrower: deterministic replay over portable decision receipts plus structural separation between native and external evidence.

OPA/Rego and Styra DAS. Open Policy Agent and Rego are mature policy-as-code systems with strong deployment support [8]; OPA decision logs record policy query events for auditing, and Styra DAS adds commercial log-replay impact analysis that replays past decisions on modified policies or data [16]. The difference is the artifact boundary: Vouch's audit unit is a portable decision receipt/report corpus that is rechecked offline, not a management-plane decision-log service, and its Bridge reports are a separate artifact class from native receipts, so external evidence cannot be structurally relabeled into native evidence.

Rules-as-code and minimal formats. OpenFisca [9] and its family member PolicyEngine [12] are open-source engines for tax/benefit modeling and reform simulation — the closest work to our welfare-shaped evaluation, but aimed at population-level microsimulation rather than portable per-decision evidence. JSON Logic [4] encodes one decision as JSON with no loops or functions. Vouch does not compete with OpenFisca on microsimulation, and it chooses a richer point in the design space than JSON Logic: a checked Lispex profile with lexical scope, exact arithmetic, pinned input, receipts, replay, and explicit artifact boundaries. What none of these provide is byte-level offline evidence for individual evaluations with external-evidence isolation.

9.2 Provenance, attestation, and transparency systems

These systems are complementary to Vouch: they authenticate who produced an artifact and when, which is precisely the layer Vouch excludes. SLSA defines supply-chain levels and provenance attestations that a build platform produced given artifacts [10], and the in-toto Attestation Framework [3] specifies verifiable claims about how software was produced; both could authenticate who produced a Vouch artifact without evaluating a decision rule or preventing class relabeling. Transparency logs — Rekor [14] and Certificate

Transparency (RFC 9162) [5] — provide append-only publication with Merkle inclusion proofs; a deployment could publish Vouch bundles into such a log, but the log authenticates publication history, not the semantic content of a rule evaluation. W3C Verifiable Credentials (v2.0) [18] let a verifier trust that claims came from an issuer untampered — almost the inverse of a Bridge report, which instead keeps external claims from being mistaken for native evidence. An issuer-binding layer built on any of these could be added around Vouch artifacts if authenticity and issuer accountability are required.

9.3 Differential testing and translation validation

Vouch's native receipt compares two transcript surfaces: a reference path and a Meaning Environment path. This relates to differential testing, a long-standing technique for finding behavioral differences between implementations [6]. Cedar also uses differential testing between a Lean model and production implementation to help ensure that proven model properties apply to the production engine [2]. The purpose, however, is different. Cedar uses differential testing as an implementation-correctness technique; Vouch uses transcript agreement as a user-facing evidence artifact, recorded in a native differential receipt together with source, input, intermediate artifacts, diagnostics, and explicit boundaries. Vouch borrows the intuition that distinct execution paths can expose mismatches, but it does not claim independent witnessing: its reference path and Meaning Environment path are same-origin and share substrate, so the result is lowered-subset transcript agreement, not semantic equivalence.

Vouch's native comparison is also related to per-instance translation validation. Classic translation validation checks a particular translation instance rather than proving an entire compiler correct; Pnueli et al. [11] introduced this line of work, and Necula [7] demonstrated translation validation for optimizing compilers. Vouch is not a compiler validator, but the analogy clarifies the role of the Meaning Environment: for each lowered artifact, Vouch records whether the source-side reference transcript and lowered graph-side transcript agree. Unlike classic translation validation, the result is not only a compiler-internal proof obligation; it is exposed as a user-facing differential receipt with explicit boundaries.

9.4 Summary: the intersection Vouch occupies

Each closest system covers part of the space — Cedar/Cedar Analysis (authorization evaluation and symbolic analysis), OPA/Styra (deployment, decision logs, log replay), OpenFisca/PolicyEngine (tax-benefit modeling), JSON Logic (a one-decision format), SLSA/in-toto/Rekor/CT/VC (provenance, issuer binding, append-only evidence), and differential testing (comparing implementations). Vouch's niche is the intersection they leave open: deterministic decision-rule evidence rechecked offline without a private service, replayed over a committed corpus under rule changes, and structurally prevented from laundering external evidence into native evidence — narrower than symbolic verification, broader than a log entry, weaker than cryptographic authenticity, and more structured than an ordinary rule-engine replay. The contribution is a fail-closed artifact boundary for semantic-evaluation evidence, not a new policy language or a proof system.

10 Broader Impact: Procedural Transparency Is Not Justice

Vouch-style receipts improve procedural transparency: they show that a rule was evaluated over a particular canonical input under a particular checked boundary, and replay discloses how rule changes affect a committed corpus. The same mechanism can be misused as an accountability shield: a deployer might present a receipt as if it proved an automated decision fair, lawful, or morally justified. That is not what Vouch attests. A receipt records what was evaluated under which public boundary; it does not decide whether the rule should exist, whether inputs were truthful, whether a policy discriminates, or whether affected people had recourse.

This limitation is deliberate. Deployments should publish rule source or reviewable policy summaries, preserve appeal channels, disclose that receipts do not attest policy legitimacy, avoid using receipt existence as a defense against contestation, treat replay reports as prompts for review, and minimize or redact sensitive input data. Vouch provides procedural transparency, not justice.

11 Limitations

Vouch is deliberately narrow. The limitations below are part of its security claim, not afterthoughts.

Checked subset only. Vouch targets the CSK Profile, not full Lispex; the Rust reference interpreter remains the operational authority for the full language.

Same-origin shared substrate. The reference and Meaning Environment paths share reader, normalizer, Core AST, value representation, numeric tower, writers, hasher, and process, so agreement can be produced by a shared bug. The artifact attests lowered-subset transcript agreement inside a disclosed shared substrate, not independent witnessing.

No semantic equivalence proof. Transcript agreement observes rendered transcript bytes only; it does not observe object identity, sharing, mutation, continuation behavior, host I/O, or error agreement.

No policy correctness. A receipt records what was evaluated under which boundary; it does not prove a business rule is morally, legally, or commercially correct.

No receipt authenticity or generation honesty. Vouch checks internal consistency but does not prove that an artifact was honestly generated, issued by a particular party, timestamped, or non-repudiable. Those require a separate trust layer.

Native read-side canonical enforcement is not unified. Bridge read-side canonical acceptance is implemented, but native read-side enforcement is not. The exposure is narrow: a native receipt's submitted bytes and parsed value may diverge under a permissive parser, so two consumers could read a duplicate-key native-shaped artifact differently. This does not undermine the class-confusion result — Bridge-only top-level key names cannot be hidden by duplicate-key shadowing — and in bundles with external expected hashes, doctored bytes fail the bundle hash. Unifying the native reader is future work.

Cross-writer formatting divergence. The byte discipline is frozen per class, not yet across writers: implementing Bridge canonical acceptance exposed formatting divergence between the Rust and JS writers. Bridge acceptance is pinned to the JS writer, native production to the Rust writer; a writer-contract unification slice should make cross-writer byte identity a single contract.

Expected-context manifest is not authority. The manifest is a consumer intent declaration, not a trusted authority. Vouch checks whether bindings match declared intent, not whether the intent is correct; context authenticity remains outside the claim.

No target-code-generation proof. Bridge reports may carry external evidence but do not prove target-code generation or semantic equivalence of a transformation.

Divergence corpus coverage. The public source corpus has no naturally occurring end-to-end divergence between the two transcripts. A seeded-divergence drill exercises the negative path deterministically: a transcript substitution yields comparison.status = disagree with a recorded first divergence, checked by the public verifier. Natural engine divergence remains unexercised.

Small evaluation. The welfare workload demonstrates realistic structure, not a comprehensive benchmark; the adversarial evaluation tests class laundering, bound-context reuse, and boundary honesty, not all deployment attacks. A full-paper extension is planned along two axes: corpus scaling with richer rule-complexity metrics, and randomized disjointness testing beyond the deterministic generated set.

No replacement for legal or human review. Receipts support audit but do not replace appeals, policy review, legal analysis, or accountability mechanisms.

12 Conclusion

Vouch addresses a narrow but important failure mode: external semantic evidence being mistaken for native checked evidence, or a genuine artifact being reused against a context it does not bind. It separates native differential receipts from external Bridge reports as different artifact classes — each with its own required fields, verifier entrypoint, and boundary — and binds each Bridge report's context through domain-separated source, target, and linked-artifact hashes. Native receipts support offline checking and rule-change replay over pinned inputs; Bridge reports carry external evidence without promoting it into native agreement; Meaning Environment reports provide a same-origin second path without claiming independent witnessing or semantic equivalence.

Class confusion is structurally rejected through artifact-class separation. Bound-context confusion is exposed when the consumer checks the artifact against supplied bytes or a declared context. Full fabrication is not prevented; it is reduced to an authenticity and issuer-binding problem kept outside the claim, with deterministic bytes and domain-separated hashes left for a separate trust layer. The result is not a proof system for correctness, authenticity, or justice — it is a reproducible, fail-closed artifact boundary for semantic-evaluation evidence that structurally rejects Bridge-to-native laundering and exposes bound-context reuse.

Artifact Availability

Artifacts are available as an anonymized artifact bundle carrying the release artifacts and their expected hashes; the bundle reproduces offline verification, replay, and hash comparison over committed artifacts without a private service. The bundle is hosted at https://anonymous.4open.science/r/vouchartifactscored26/.

References

[1] Cedar Policy Project. 2025. Cedar Analysis: SMT-based Policy Analysis for Cedar. Open-source toolkit, https://github.com/cedar-policy. Translates Cedar policies into SMT formulas via a Lean-implemented symbolic compiler with soundness and completeness guarantees.

[2] Joseph W. Cutler, Craig Disselkoen, Aaron Eline, Shaobo He, Kyle Headley, Michael Hicks, Kesha Hietala, Eleftherios Ioannidis, John Kastner, Anwar Mamat, Darin McAdams, Matt McCutchen, Neha Rungta, Emina Torlak, and Andrew M. Wells. 2024. Cedar: A New Language for Expressive, Fast, Safe, and Analyzable Authorization. *Proceedings of the ACM on Programming Languages* 8, OOPSLA1, Article 118 (2024), 28 pages. doi:10.1145/3649835

[3] in-toto Project. 2026. in-toto Attestation Framework. Online documentation, https://in-toto.io.

[4] JsonLogic. 2026. JSON Logic Documentation. Online documentation, https://jsonlogic.com.

[5] Ben Laurie, Eran Messeri, and Rob Stradling. 2021. *Certificate Transparency Version 2.0.* Request for Comments RFC 9162. Internet Engineering Task Force (IETF). doi:10.17487/RFC9162

[6] William M. McKeeman. 1998. Differential Testing for Software. *Digital Technical Journal* 10, 1 (1998), 100–107.

[7] George C. Necula. 2000. Translation Validation for an Optimizing Compiler. In *Proceedings of the ACM SIGPLAN 2000 Conference on Programming Language Design and Implementation (PLDI '00).* ACM, Vancouver, British Columbia, Canada. doi:10.1145/349299.349314

[8] Open Policy Agent Project. 2026. Open Policy Agent (OPA) Documentation: Decision Logs and Bundles. Online documentation, https://www.openpolicyagent.org/docs.

[9] OpenFisca. 2026. OpenFisca: A Rules-as-Code Engine for Tax and Benefit Systems. Online documentation, https://openfisca.org.

[10] OpenSSF. 2026. Supply-chain Levels for Software Artifacts (SLSA). Online documentation, https://slsa.dev.

[11] Amir Pnueli, Michael Siegel, and Eli Singerman. 1998. Translation Validation. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 1998) (Lecture Notes in Computer Science, Vol. 1384)*, Bernhard Steffen (Ed.). Springer, Berlin, Heidelberg, 151–166. doi:10.1007/BFb0054170

[12] PolicyEngine. 2026. PolicyEngine: OpenFisca-family Microsimulation. Online documentation, https://policyengine.org.

[13] Yaron Sheffer, Dick Hardt, and Michael B. Jones. 2020. *JSON Web Token Best Current Practices.* Request for Comments RFC 8725. Internet Engineering Task Force (IETF). doi:10.17487/RFC8725

[14] Sigstore. 2026. Rekor: Software Supply Chain Transparency Log (Sigstore). Online documentation, https://docs.sigstore.dev.

[15] Juraj Somorovsky, Andreas Mayer, Jörg Schwenk, Marco Kampmann, and Meiko Jensen. 2012. On Breaking SAML: Be Whoever You Want to Be. In *21st USENIX Security Symposium (USENIX Security 12).* USENIX Association, Bellevue, WA, 397–412.

[16] Styra, Inc. 2026. Styra DAS: Decision Log Replay and Impact Analysis. Product documentation, https://docs.styra.com.

[17] Vouch 2026. Lispex Vouch Bridge Contract (VOUCH-BRIDGE.md, v1.3.11). Public contract document, anonymized artifact bundle. Defines vouch.bridge-report/v0, vouch.bridge-context-manifest/v0, and the lispex verify-bridge checker. Included in the anonymized artifact bundle accompanying this submission.

[18] World Wide Web Consortium. 2025. Verifiable Credentials Data Model v2.0. W3C Recommendation, https://www.w3.org/TR/vc-data-model-2.0/.