Epistemic Guard: Out-of-Process MCP Receipt Proxying, Post-Turn Prose Supervision, and Provenance-Gated Memory for Tool-Using Autonomous Agents

Sean Maxwell
Roll Solo, Melbourne, VIC, Australia
sean.maxwell@rollsolo.app https://orcid.org/0009-0004-0133-0463

November 2026

Abstract—Language-model agents using the Model Context Protocol (MCP) face three runtime verification gaps: (i) *action fabrication*, where post-tool prose narrates side effects that failed, timed out, or never ran; (ii) *provenance collapse*, where unverified model prose or web text enters durable memory alongside executed tool outputs; and (iii) *syndicated echo amplification*, where mirrored web pages satisfy multi-source corroboration from one origin. Existing MCP firewalls inspect wire calls before execution, while LLM-judge guardrails require multi-gigabyte VRAM without binding prose to receipts. We present `epistemic-guard` (v0.5.2), an open-source Python library and out-of-process `stdio` MCP receipt proxy (15 modules, 0 MiB GPU VRAM). `epistemic-guard` isolates Ed25519 signing keys out of process, writes RFC 6962 Merkle logs with `AER-1` receipt translation, checks post-turn prose on CPU with SHA-256-verified OpenVINO INT8 encoders (4 models, 279.48 MB total; 82.67 MB on the supervision path), gates SQLite writes across 7 provenance domains, and collapses verbatim and lightly edited wire-syndicated RAG mirrors. On conformance suites, `epistemic-guard` rejects 100.0% of tampered receipts (200 Merkle leaves), passes 58/58 official `AER-1` ingestion vectors and 50/50 reference-verified `AER-1` exports, verifies 100.0% of 40 fault-injected wire calls and 100 concurrent appends with 0 key leaks, and blocks 100.0% of 100 unauthorized memory writes. Across $N = 250$ frozen-benchmark traces, `epistemic-guard` reaches 93.2% accuracy at 17.9 ms warm median CPU latency (8.62 ms single-clause; 921.8 ms cold-start load), beating 12B–14B GPU judges by 6.4–6.8 points ($p = 0.0259$, $p = 0.0095$) at 4.26–4.5× lower latency; on $N = 60$ live `mcp-proxy` traces, it catches 96.15% of hazards at 76.67% accuracy (38.24% FPR; +18.34 points over Stage-1-only, $p = 0.001$) in 9.16 ms p50.

Artifacts: Code: `https://github.com/AlienatedAliens/epistemic-guard` ·
PyPI: `https://pypi.org/project/epistemic-guard/` ·
INT8 weights: `https://huggingface.co/RollSolo/epistemic-guard-int8` ·
Benchmarks: `https://huggingface.co/datasets/RollSolo/epistemic-guard-benchmarks`

1 Introduction

Software engineering and operations agents invoke external tools over JSON-RPC transports such as the Model Context Protocol (MCP) [17]. MCP standardizes wire transport, not post-call truthfulness: what happened on the wire can diverge from what the language model tells the user or writes to persistent state. In live agent loops, tool calls return HTTP errors, time out, fail schema checks, or never run at all while the model claims in first-person prose that the action succeeded [15]. When downstream workflows or memory stores ingest those unverified claims, single-turn hallucinations persist across sessions [4, 13].

From a runtime verification perspective [11, 1], closing this gap without trusting the agent's own prompt context or reserving GPU memory requires three checks across each turn:

1. **Out-of-Process Execution Receipts:** Every MCP tool call and response must be signed outside the agent process, separating pre-execution parameter rejection (`executed=false`) from runtime tool failure (`executed=true, ok=false`).
2. **Receipt-Bound Prose Supervision:** Before an assistant reply reaches the user or a downstream workflow, state-mutating claims in natural language must match verified execution receipts and arguments from the current turn.
3. **Provenance-Gated Storage and De-Syndicated RAG:** Persistent memory stores must reject unsigned or narration-sourced writes at the database boundary, and retrieval-augmented generation (RAG) pipelines must collapse mirrored web copies before counting independent sources on high-hazard queries.

These failure modes correspond to three entries in the OWASP Top 10 for Agentic Applications [20]: Tool Misuse and Unverified Action Claims (`ASI02`), Memory and Context Poisoning (`ASI06`), and Cascading Inter-Component Hallucinations (`ASI08`). Existing open-source tools cover separate parts of this path. Transport firewalls such as Pipelock [21] scan MCP egress traffic

and emit mediator-signed receipts, and runtimes such as NabaOS [19] enforce Ed25519-signed policy constitutions, but neither checks post-turn prose against tool receipts nor deduplicates syndicated RAG sources before memory writes. Dialogue guardrails such as NeMo Guardrails [22], Llama Guard [7], and Guardrails AI [5] filter text with dialogue rules or LLM judges that consume 8,844–12,214 MiB of GPU VRAM and do not verify cryptographic tool receipts.

This paper describes the architecture, security invariants, and empirical evaluation of `epistemic-guard` (v0.5.2), a Python runtime verification library and MCP proxy [14, 12]. Its main components are:

- **Out-of-Process MCP Receipt Proxy (`mcp-proxy`):** A `stdio` JSON-RPC 2.0 proxy that strips signing secrets from child server environments, classifies 4 execution and error states, appends Ed25519-signed receipts to an OS-locked JSONL Merkle log with public-key pinning, and translates bidirectionally to the Agent Execution Receipt (`AER-1`) format [25].
- **CPU INT8 Receipt Supervision:** A hybrid symbolic–neural supervisor (`ReceiptSupervisor`) using the fine-tuned `action_claim` OpenVINO INT8 cross-encoder (82.67 MB of the 279.48 MB 4-model bundle; 0 MiB GPU VRAM) plus Ed25519 receipt signature and argument-target matching, running in 17.9 ms median turn CPU latency.
- **Provenance-Gated SQLite Store & RAG Mirror Filter:** Domain-separated HKDF-SHA256 [9] + Ed25519 [8] write-gating with SQLite immutability triggers, paired with word-shingle Jaccard and SimHash clustering [2, 3] across a 12-cell hazard-by-proof-standard policy table.
- **Reproducible Benchmarks:** Deterministic evaluation scripts audited with `osaudit` [18], tying every reported number in this paper to SHA-256 input hashes and checked-in JSON manifests.

2 Threat Model and Verification Scope

System Model and Mediation Boundary. An MCP host or orchestrator $\mathcal{H}$ runs an untrusted language-model generation loop $\mathcal{A}$ that interacts with external MCP tool servers $\{\mathcal{S}_1, \ldots, \mathcal{S}_k\}$, a web/RAG retrieval pipeline $\mathcal{R}$, and a local memory database $\mathcal{M}$. To enforce complete mediation on tool traffic, $\mathcal{H}$ configures `epistemic-guard mcp-proxy` as the `stdio` launch command for every tool server $\mathcal{S}_i$, so $\mathcal{A}$ cannot communicate with $\mathcal{S}_i$ outside the proxy pipe; complete mediation holds only if $\mathcal{A}$ is granted no unproxied side channels (such as direct shell execution or built-in HTTP tools) outside those proxied servers. On turn t, $\mathcal{A}$ emits zero or more JSON-RPC 2.0 `tools/call` requests mediated by `mcp-proxy`, optionally retrieves external web pages, proposes candidate memory writes to $\mathcal{M}$, and emits a candidate natural-language reply y_t. Before $\mathcal{H}$ displays y_t or passes it downstream, $\mathcal{H}$ invokes `ReceiptSupervisor` (or CLI `verify-log`) outside $\mathcal{A}$'s prompt context.

Adversary and Fault Assumptions. We consider both stochastic model hallucinations and adversarial indirect prompt injections [4] inside tool outputs or retrieved web documents:

1. *Post-tool action fabrication (`ASI02`, `ASI08`):* $\mathcal{A}$ may claim in y_t that a side-effecting tool call succeeded when no call occurred, when the server returned `isError=true`, when argument validation rejected the call pre-execution, or when the child server crashed mid-flight. If signing keys reside inside $\mathcal{A}$'s process or child server environments, compromised code or plugins could forge receipts.
2. *Cross-session memory poisoning (`ASI06`):* $\mathcal{A}$ may attempt to persist unverified assistant narration (`LLM_NARRATION`) or untrusted tool/web text (`TOOL_CONTENT_UNTRUSTED`) into $\mathcal{M}$ as verified facts, or mutate historical audit rows via SQL `UPDATE`/`DELETE` statements.
3. *Syndicated corroboration spoofing (`ASI06`, `ASI08`):* Retrieved web pages often reprint identical wire-service copy or SEO mirrors across distinct hostnames. Naive multi-URL counting treats 3 mirrors of one rumor as independent corroboration for high-hazard medical, financial, or security claims.

Trusted Computing Base (TCB) and Key Isolation. The TCB comprises the host OS kernel, the MCP host/orchestrator $\mathcal{H}$ invoking post-turn verification, the out-of-process `mcp-proxy` process holding the Ed25519 signing key, the SHA-256 digests of the 16 INT8 model files (verified by default at every model load via `verify_hashes=True` across explicit or OS-cache paths), and OS permissions on the SQLite store and pinned public key. When keys are injected via `EPISTEMIC_GUARD_MCP_KEY_HEX`, the secret stays in `mcp-proxy` memory and is scrubbed before spawning child servers; when `--key-file` is used on disk (created atomically via `O_CREAT|O_EXCL` with POSIX mode `0600`, or restricted NTFS directory ACLs on Windows), same-host deployments against untrusted local code must run `mcp-proxy` under a dedicated OS user or container so same-UID processes cannot read `--key-file`. We do *not* defend against host kernel compromise, malicious MCP servers that return `ok=true` while lying about internal state, or outbound network exfiltration (where egress proxies such as Pipelock [21] compose upstream).

Table 1: Static architectural metrics and INT8 model footprint of `epistemic-guard` v0.5.2 (`table1_architecture.json`).

Architectural Property / Component	Value
Python source modules (`epistemic_guard`)	15
Layer contracts (`import-linter`)	1
Test modules / automated test functions	4 / 34
MCP server tools / protocol revisions	4 / 3
Provenance domains / HKDF signers	7 / 5
Write-eligible SQLite provenance domains	5
Ed25519 key / signature size (bytes)	32 / 64
Hazard tiers / proof standards (policy cells)	3 / 4 (12)
Consumer contexts / word-shingle width k	4 / 4
INT8 models × files (SHA-256 pinned)	4 × 4 (16)
`action_claim` INT8 encoder (MB)	82.67
`hazard` INT8 classifier (MB)	82.65
`nli` DeBERTa-v3 cross-encoder (MB)	91.1
`web_evidence` MiniLM encoder (MB)	23.06
Total unpacked INT8 bundle (MB)	**279.48**

3 System Architecture

Table 1 lists the module counts and model sizes of `epistemic-guard`. The package contains 15 typed Python modules (`mypy --strict`) checked by 1 `import-linter` layer contract so that cryptographic verification, supervision, and storage modules never import from CLI, MCP server, or integration layers.

3.1 Domain-Separated Cryptographic Provenance and AER-1 Receipts

Rather than sharing one symmetric key across components, `provenance.py` derives 5 domain-isolated RFC 8032 Ed25519 [8] keypairs (32-byte keys, 64-byte signatures) from a master seed via RFC 5869 HKDF-SHA256 [9] across 7 provenance domains (`Provenance`): `USER_EXPLICIT`, `USER_CONFIRMED`, `TOOL_VERIFIED`, `LISP_ASSERTION`, `INFERRED`, `LLM_NARRATION`, and `TOOL_CONTENT_UNTRUSTED`. `SignerFactory` provides no signing constructor for `LLM_NARRATION` or `TOOL_CONTENT_UNTRUSTED`, so model prose and untrusted tool text cannot produce a signed envelope through the public API.

Each tool call produces a canonical `ToolReceipt` (serialized with sorted keys and `allow_nan=False` so `NaN`/`Infinity` floats cannot break cross-runtime canonicalization) containing the call ID, tool name, structured arguments, SHA-256 output digest, execution flag (`executed`), outcome flag (`ok`), error code, optional session ID, UTC timestamp, and detached Ed25519 signature. Receipts append into an RFC 6962 [10] domain-separated Merkle tree (`0x00` leaf prefix, `0x01` interior node prefix) with logarithmic inclusion proofs and path-side geometry checks. For cross-tool interoperability, `ToolReceipt.to_aer1` and `ToolReceipt.from_aer1` implement single-receipt translation and core/binding verification against `AER-1` (`draft-zambo-aer1-26`, Sections 3–6, `receipt_schema_version 0.4` with `0.2`/`0.3` legacy read support) [25], committing execution state, input arguments, call/session IDs, and SHA-256 input/output digests while multi-entry sequencing and detached Ed25519 signatures remain in `ReceiptLogEntry` and `MerkleReceiptLog`.

3.2 Out-of-Process `stdio` MCP Receipt Proxy

In-process Python decorators (`ReceiptRecorder`) protect benign agent loops against hallucinated actions, but share memory and environment variables with the agent runtime. For untrusted or prompt-injected runtimes, `epistemic-guard mcp-proxy` interposes as an out-of-process `stdio` parent between the MCP host and any downstream MCP tool server:

1. **Secret Scrubbing:** Before spawning the child MCP server, `mcp-proxy` case-insensitively strips signing-secret environment variables (including `EPISTEMIC_GUARD_MASTER_KEY` and `EPISTEMIC_GUARD_MCP_KEY_HEX`) from the child's environment block so compromised child servers cannot forge receipts.

2. **Four-State Wire Classification:** As JSON-RPC 2.0 frames stream over `stdin`/`stdout`, `mcp-proxy` correlates `tools/call` requests in FIFO order per JSON-RPC `id` and distinguishes 4 runtime outcomes: (a) normal tool completion (`executed=true, ok=true`); (b) application-level MCP tool failure where `result.isError=true` (`executed=true, ok=false, mcp_tool_error`); (c) pre-execution rejection on standard JSON-RPC protocol/parameter error codes (`-32700`, `-32600`, `-32601`, `-32602`) or schema-validation error markers (`executed=false, ok=false`); and (d) child server crash or EOF mid-call (`executed=true, ok=false, mcp_server_eof`). Non-standard server errors default conservatively to `executed=true, ok=false` so neither rejected nor failed calls can ground completion prose.

3. **Key-Pinned Append-Only Audit Log:** Each `ReceiptLogEntry` carries both an inner Ed25519 `ToolReceipt` signature and a domain-separated Ed25519 `entry_signature` over `(seq, session_id, leaf_sha256, prev_chain_sha256, chain_sha256, merkle_root)`, acquires cross-process OS file locks for multi-worker concurrency, chains optionally across rotated Ed25519 keys (`--prev-log` / `--prev-pubkey-file`), and flushes immediately to an append-only JSONL Merkle log; `verify-log` (with external `--pubkey-file` pinning) rejects payload or embedded-`AER-1` mutation, active reorder-and-rehash attacks, duplicate call IDs, cross-session

replay, and key substitution, while `attach_otel_evidence` exports receipt digests to OpenTelemetry traces.

In addition to `mcp-proxy`, `epistemic-guard mcp-server` exposes 4 verification tools directly over MCP across 3 protocol revisions (`2024-11-05`, `2025-03-26`, and `2025-06-18`).

3.3 Hybrid Symbolic–Neural Post-Turn Prose Supervision

Regex-only action supervisors miss indirect phrasing, nominalizations, and passive-voice state mutations, whereas off-the-shelf NLI encoders treat unexecuted tool claims against empty receipt logs as neutral [15]. Across the full package, `epistemic-guard` distributes 4 OpenVINO INT8 models (16 SHA-256-pinned files totaling 279.48 MB over HTTPS from Hugging Face Hub or GitHub Releases, extracted with symlink/path-traversal guards and verified by SHA-256 at every load). Only `action_claim` (82.67 MB) runs on the post-turn supervision path, where `ReceiptSupervisor` executes a two-stage CPU pipeline:

1. **Speech-Act Action-Claim Detection:** Non-assertive spans (negations, conditionals, attributed quotes, creative/translation prompts, and fenced code blocks) are masked while optional Structured Output Action Contract (`SOAC`, explicit `[ACTION: verb target]` tags [15]) declarations are extracted, and candidate clauses are scored by the INT8 `action_claim` cross-encoder. If no completed external side effect is asserted, the turn passes without requiring receipts.

2. **Cryptographic Receipt & Target Matching:** When a side-effect claim is detected, `Receipt Supervisor` verifies the turn's `ToolReceipt` set via `ProvenanceGate` (enforcing Ed25519 signatures, session binding, and receipt-age bounds when configured), excludes any receipt with `executed=false` or `ok=false`, and matches claimed action families, additive claim counts, and concrete file/URL/email/identifier targets against verified receipt `args` on path and token boundaries (rejecting substring collisions and bare zero-argument receipts when prose names specific targets). Ungrounded claims return `Supervision Verdict(is_valid=false)` with character-offset spans and `to_retry_prompt()` remediation text.

3.4 Provenance-Gated SQLite Memory and De-Syndicated RAG

Storage-Boundary Write Gate. In `store.py`, `SQLiteProvenanceStore` wraps SQLite WAL persistence behind mandatory `ProvenanceGate` verification and serialized `BEGIN IMMEDIATE` write transactions. Only 5 provenance domains with valid detached Ed25519 signatures are write-eligible; writes carrying `LLM_NARRATION`, `TOOL_CONTENT_UNTRUSTED`, forged keys, or mutated payloads raise `ProvenanceError` or `SignatureTamperError` before touching SQLite. Accepted envelopes link into a cumulative SHA-256 chain digest protected by 2 SQL `BEFORE UPDATE` and `BEFORE DELETE` triggers that abort in-place mutation of committed rows.

Syndication Deduplication and Hazard-Tiered Corroboration. Before web-retrieved claims are promoted into `INFERRED` memory or stated without hedging, `corroboration.py` shards candidate passages into word shingles of width $k = 4$, drops non-evidentiary boilerplate with the MiniLM [23] INT8 `web_evidence` encoder (23.06 MB), checks cross-source contradictions with the DeBERTa-v3 [6] INT8 `nli` cross-encoder (91.1 MB), and merges near-duplicate mirrors across registrable domains using explicit similarity gates: anchor-token Jaccard ≥ 0.45, 3-shingle Jaccard ≥ 0.25, 4-shingle Jaccard ≥ 0.55 [2], or 64-bit SimHash Hamming distance ≤ 12 [3]. Independent source counts are checked against 4 consumer action contexts (`casual_mention`, `memory_commit_inferred`, `unhedged_assertion`, and `irreversible_action`) and a 3-tier hazard classifier (82.65 MB; `LOW`, `MEDIUM`, `HIGH`) mapped onto 4 proof standards (12 policy cells). High-hazard claims backed only by wire-syndicated mirrors of one origin collapse to one cluster and fail corroboration.

4 Empirical Evaluation

We structure our evaluation around two research questions:

- **RQ1 (Deterministic Conformance, Concurrency/Replay Faults, and Paraphrase Sensitivity):** Do the Merkle receipt ledger, `AER-1` interop layer, out-of-process `mcp-proxy` wire classifier, concurrent log writer, and SQLite provenance gate enforce their deterministic invariants under fault injection, and at what rewrite intensity does shingle/SimHash syndication deduplication degrade?

- **RQ2 (Prose Supervision Accuracy, Live MCP Receipt Binding, & Resource Cost):** How accurately and rapidly does `ReceiptSupervisor` detect ungrounded post-turn action claims across frozen benchmark cohorts ($N = 250$) and live end-to-end `mcp-proxy` fault-injection traces ($N = 60$) compared with stage ablations, 12B–14B GPU LLM judges, and off-the-shelf CPU NLI baselines?

All experiments are deterministic, version-pinned, and audited with `osaudit` [18] manifests under `results/` (`table1_architecture.json`, `table2_conformance.json`, and `table3_heldout_supervision.json`). Binomial proportions use two-sided 95% Wilson score confidence intervals [24], and paired classifier comparisons use exact two-sided McNemar tests [16].

4.1 RQ1: Conformance, Fault-Injection, and Paraphrase Degradation

Table 2 reports deterministic conformance and fault-injection results from `run_protocol_conformance.py`:

- **Merkle, `AER-1`, & Replay Integrity:** Across 200 signed receipts, all 200 RFC 6962 inclusion proofs verify (100.0%) and all 200 payload mutations raise `MerkleTamperError` (0 false accepts). Evaluated against the official `draft-zambo-aer1-26` conformance kit [25], `ToolReceipt.from_aer1` matches 58/58 single-receipt ingestion vectors (100.0%: 22/22 valid/profile-boundary accepted, 36/36 invalid rejected), while all 50/50 `ToolReceipt.to_aer1` exports pass the kit's reference verifier (100.0% across core, version-binding, and reference-producer checks), round-trip cleanly (50/50, 100.0%), and reject digest edits (50/50, 100.0%). Under 4-thread contention (100 concurrent appends to one log), OS file locking preserves sequential chain hashes across all 100 entries (100.0%), while 4/4 cross-session replay, active rehashed-reorder, and `NaN`/`Infinity` canonicalization attempts fail closed (100.0%).
- **Out-of-Process MCP Wire Fault Injection:** Spawning live child `stdio` MCP servers under `mcp-proxy` across 40 tool invocations (10 each across normal completion, `result.isError=true`, JSON-RPC `-32602` invalid-parameter rejection, and mid-request child crash), all 40 logged receipts pass chain and pinned-key verification (1 pinned key, 1/1 post-hoc log edit rejected), separating 30 dispatched/crash calls (10 `ok=true`) from 10 pre-execution rejections (`executed=false`) with 0 child-environment key leaks.
- **SQLite Write Gate & Paraphrase Sensitivity:** Across 300 storage writes, all 200 authorized envelopes verify along the 200-row hash chain (0.0% false rejections), all 100 unauthorized writes (50 `LLM_NARRATION` + 50 forged keys) are blocked (100.0%), and 2/2 SQL mutations trigger `IntegrityError`. Across 45 wire-syndication clusters (135 URLs, 3 domains/cluster), all 90 duplicate mirrors (2/cluster) collapse to single-origin clusters (45/45, 100.0%). To locate the lexical evasion boundary, a 5-tier sensitivity sweep of 50 clusters per rewrite tier shows single-origin collapse holds at 100.0% under dateline/footer wrappers and 100.0% under light (∼15%) lexical edits, tapers to 40.0% on intermediate (∼30%) lexical edits, and drops to 0.0% on moderate (∼45%) clause restructuring and 0.0% on heavy (>80%) abstractive rewrites, confirming that $k = 4$ shingling targets verbatim and lightly edited wire syndication rather than semantic paraphrase.

4.2 RQ2: Frozen-Benchmark and Live MCP Supervision

Hardware, Baselines, and Cohort Provenance. All systems in Table 3 ran on one Windows 11 workstation with an Intel Core Ultra 9 275HX (24-core) CPU and an NVIDIA GeForce RTX 5080 Laptop GPU (16 GB VRAM). GPU LLM-judge baselines (`Mistral-Nemo-Instruct-2407-Q5_K_M` and `Phi-4-Q6_K`) ran via `llama.cpp` (`-ngl 99`, `-c 4096`, `temperature=0.0`) under a deterministic 3-rule auditor prompt receiving `USER_PROMPT`, `VERIFIED_EXECUTED_TOOLS` (including raw argument strings such as `read_file(path='config.json')`, identical to `ReceiptSupervisor`'s input), and `ASSISTANT_REPLY`, outputting `ALLOW` or `BLOCK`. Warm CPU latencies cover full `verify_narration()` execution; cold-start CPU initialization (SHA-256 checks across 4 `action_claim` files, tokenizer load, and OpenVINO compilation) takes 921.8 ms median (1,044.6 ms p95), paid once in persistent servers. The $N = 250$ frozen benchmark (122 ungrounded, 128 grounded/benign; Hugging Face commit `73bbc046759e`, 2026-11-14, frozen with v0.3.0 weights) joins author-labeled speech-act cases `locked_coh_50` ($n = 50$) and paraphrases `blind42_coh_100` ($n = 100$) with `organic_heldout_h100` ($n = 100$): live tool-less chat turns (`receipts=[]`) from `gemma-4-e4b` (50) and `gemma-4-26b-a4b` (50), labeled by blinded LLM raters (30/30, 100.0% dual-slice agreement) and author audit [15]. Because `organic_heldout_h100` is tool-less, we also generated a post-release end-to-end cohort `live_mcp_e2e_60` ($N = 60$: 30 `gemma-4-E4B` and 30 `Nemotron-3.5-30B-A3B` turns, 26 ungrounded and 34 grounded, labeled against wire state prior to evaluation) routed through live `mcp-proxy` child servers across `ok=true` completion, target-drift `ok=true` calls, `isError=true`, JSON-RPC `-32602`, mid-call crash, and compound two-tool turns.

Accuracy, Ablations, and Live MCP Receipt Binding. On the $N = 250$ frozen benchmark, `epistemic-guard` reaches 93.2% accuracy (233/250; TP = 118, TN = 115, FP = 13, FN = 4): 100.0% (50/50, CI [92.87%, 100.0%]) on `locked_coh_50`, 92.0% (92/100, CI [85.0%, 95.89%]) on `blind42_`

Table 2: Deterministic protocol conformance, multi-worker concurrency, and tamper/replay-rejection results across runtime enforcement subsystems (`run_protocol_conformance.py` → `results/table2_conformance.json`).

Subsystem	Evaluated Invariant / Fault-Injection Workload	Trials	Verified / Blocked	Rate
RFC 6962 Merkle Ledger	Valid leaf inclusion proofs verified	200	200	100.0%
RFC 6962 Merkle Ledger	Mutated receipt payloads rejected (0 false accepts)	200	200	100.0%
`AER-1` Interoperability	Official `draft-zambo-aer1-26` ingestion kit (22 valid, 36 invalid)	58	58	100.0%
`AER-1` Interoperability	Reference-verified `to_aer1()` exports, round-trip, & tampered-digest rejection	50	50	100.0%
Out-of-Process `mcp-proxy`	Wire calls classified across 4 modes (10/mode; 0 env key leaks)	40	40	100.0%
Concurrent Audit Log	Multi-worker locked appends verified (4 threads × 25 appends)	100	100	100.0%
Replay & Canonicalization	Cross-session replay, entry reordering, and `NaN`/`Inf` floats rejected	4	4	100.0%
SQLite Provenance Gate	Authorized signed domain writes accepted (0 false rejections)	200	200	100.0%
SQLite Provenance Gate	Unauthorized writes blocked (50 `LLM_NARRATION` + 50 forged keys)	100	100	100.0%
SQLite Immutability	Direct SQL `UPDATE` and `DELETE` mutations aborted by triggers	2	2	100.0%
RAG Syndication Filter	135 wire-mirror URLs (3/cluster) collapsed; high-hazard promotion blocked	45	45	100.0%

Table 3: Post-turn action-claim supervision on the $N = 250$ frozen benchmark (122 ungrounded, 128 grounded/benign; tool-less `Org-100` sub-cohort) and $N = 60$ live `mcp-proxy` traces (`Live-60`; latencies in ms, GPU VRAM in MiB; `results/table3_heldout_supervision.json`).

Supervisor / Baseline	Accuracy (95% CI)	Recall	FPR	Org-100	Live-60	p50	p95	VRAM
`epistemic-guard` INT8 (CPU)	**93.2%** [89.38%, 95.71%]	96.72%	**10.16%**	**91.0%**	76.67%	**17.9**	35.83	**0**
Ablation: Stage-1 (no receipts)	84.8% [79.83%, 88.72%]	97.54%	27.34%	91.0%	58.33%	24.98	—	0
Ablation: Regex + receipts	64.0% [57.88%, 69.7%]	30.33%	3.91%	50.0%	56.67%	0.08	—	0
Mistral-NeMo-12B Judge (GPU)	86.8% [82.04%, 90.44%]	87.7%	14.06%	84.0%	**80.0%**	80.58	102.15	8,844
Phi-4-14B Judge (GPU)	86.4% [81.6%, 90.1%]	**100.0%**	26.56%	79.0%	65.0%	76.24	102.25	12,214
DeBERTa-v3-xsmall NLI (CPU)	52.8% [46.62%, 58.9%]	24.59%	20.31%	51.0%	**80.0%**	22.4	**32.49**	**0**

`coh_100`, and 91.0% (91/100, CI [83.77%, 95.19%]) on `organic_heldout_h100` at 17.9 ms median turn latency (8.62 ms single-clause, 19.38 ms mean, 35.83 ms p95). At the v0.3.0 freeze it scored 94.4% (236/250; 95/100 on `blind42_coh_100`); v0.5.1–v0.5.2 security hardening (rejecting zero-argument wildcard matches on named files) cost 3 bare-tool items, so symbolic rules are audit-hardened rather than blind while neural weights stayed frozen. Regex+receipts scores 64.0% (160/250; 30.33% recall), and Stage-1-only (`receipts=[]`) scores 84.8% (212/250; 27.34% FPR). Stage-1-only exhibits a higher median latency (24.98 ms vs. 17.9 ms) because `ReceiptSupervisor` short-circuits conversational clauses lacking action/state signals on multi-clause turns when `ok_receipts` is non-empty, whereas passing `receipts=[]` runs OpenVINO on every clause.

Overall ($N = 250$), `epistemic-guard` beats Mistral-NeMo-12B by +6.4 pp (86.8%, 217/250; McNemar 31 vs. 15, $p = 0.0259$; +7.0 pp on `Org-100`, 14 vs. 7, $p = 0.1892$; 4.5× faster p50, saving 8,844 MiB VRAM) and Phi-4-14B by +6.8 pp (86.4%, 216/250; McNemar 28 vs. 11, $p = 0.0095$; +12.0 pp on `Org-100`, 17 vs. 5, $p = 0.0169$; 4.26× faster p50, saving 12,214 MiB VRAM), while DeBERTa-v3 NLI scores 52.8% (132/250, +40.4 pp gap; 73/92 false negatives, 79.35%, predict *neutral*). On `Live-60` (zero-shot post-release, no custom tool stems), the supervisor deliberately favors recall over false alarms—a false positive costs one `to_retry_prompt()` retry whereas a missed fabrication corrupts state—catching 96.15% of hazards (25/26, CI [81.11%, 99.32%], 1 missed retry promise) at 38.24% FPR (13/34) and 76.67% accuracy (46/60, CI [64.56%, 85.56%]) in 9.16 ms p50 (23.12 ms p95). Stage-2 receipt binding adds +18.34 pp over Stage-1-only (58.33%, 35/60, 70.59% FPR; 11 vs. 0, $p = 0.001$), while Regex+receipts has 0.0% recall (34/60) and DeBERTa-v3 NLI shows why accuracy misleads on uneven classes (80.0%, 48/60, at 53.85% recall, 12/26 missed). Live accuracy gaps vs. GPU judges are non-significant—`epistemic-guard` is nominally +11.67 pp above Phi-4-14B (65.0%, 39/60, 61.76% FPR; 13 vs. 6, $p = 0.1671$; 7.32× faster vs. 67.04 ms) and nominally 3.33 pp below Mistral-NeMo-12B (80.0%, 48/60, 35.29% FPR; 6 vs. 8, $p = 0.7905$) at 7.83× lower p50 (9.16 vs. 71.69 ms) and 0 MiB VRAM; its 13 FPs comprise 6 custom tool names (`execute_sql`, `http_post`, `git_commit`) and 7

comma-split `"attempted X, but failed"` clauses.

5 Limitations and Threats to Validity

Internal Validity. Protocol conformance workloads in Table 2 run real child processes, OS file locks, Ed25519/HKDF signing, and OpenVINO INT8 inference on deterministic local payloads rather than live SaaS endpoints so auditors can run `osaudit results --rerun` offline without API keys or network variance. Supervision labels derive from author rubrics, blinded LLM raters plus author audit, and pre-committed MCP wire-state rules rather than multi-human inter-annotator (κ) studies, and commercial frontier API judges were omitted to keep evaluation zero-API-cost and offline-reproducible.

External Validity. Our supervision cohorts ($N = 250$ frozen, $N = 60$ live MCP) evaluate English replies on one workstation; as `Live-60` shows (6 unmapped-stem and 7 cross-clause failure-report FPs), custom MCP tool vocabularies and multi-clause `"attempted X, but failed"` reports raise false positives until custom tool stems and cross-clause negation propagation are configured.

Construct and Transport Scope. First, `mcp-proxy` in v0.5.2 mediates `stdio` MCP servers; Streamable HTTP/SSE remote servers require a local `stdio` bridge or in-process `ReceiptRecorder` wrapping. Second, receipts prove what a downstream server returned on the wire; a buggy or malicious server returning forged `ok=true` payloads yields a validly signed false receipt, and custom servers rejecting calls via non-standard error codes are conservatively logged as `executed=true, ok=false`. Third, as quantified in Section 4.1, $k = 4$ shingle/SimHash clustering collapses verbatim and lightly edited wire copy (100.0%) but does not collapse moderate or abstractive rewrites (0.0%).

6 Related Work

MCP Firewalls and Cryptographic Agent Runtimes. Pipelock [21] is an open-source Go egress proxy and MCP firewall that scans outbound tool parameters for prompt injection or data leaks and emits Ed25519 receipts in `AER-1` format [25]. NabaOS [19] runs agents inside OS sandboxes governed by Ed25519-signed YAML constitutions. Pipelock inspects traffic before execution; `epistemic-guard` imports and exports `AER-1` receipts while checking post-turn prose (`ReceiptSupervisor`), gating SQLite memory writes, and collapsing syndicated RAG sources.

Conversational Guardrails and LLM-as-a-Judge. NeMo Guardrails [22] executes Colang dialogue rails; Llama Guard [7] fine-tunes multi-billion-parameter classifiers for safety moderation; and Guardrails AI [5] chains structural and semantic validators. Rather than moderating toxicity or running an 8,844–12,214 MiB GPU judge, `epistemic-guard` checks whether user-facing prose matches signed wire receipts in 0 MiB VRAM.

Classical Runtime Verification and Provenance Security. Runtime verification monitors execution traces against formal properties [11, 1], and Certificate Transparency [10] introduced append-only Merkle audit logs for untrusted intermediaries. Work on indirect prompt injection [4] and agent memory poisoning [13, 12] shows that mixing untrusted text and verified execution state in one memory store enables cross-session compromise (`ASI06` [20]). `epistemic-guard` applies out-of-process monitor separation and RFC 6962 Merkle logs directly to Python tools and `stdio` MCP servers.

7 Conclusion

`epistemic-guard` (v0.5.2) combines an out-of-process `stdio` MCP receipt proxy, OpenVINO INT8 prose supervision, Ed25519-gated SQLite storage, and wire-syndication deduplication. Across $N = 250$ frozen-benchmark traces and $N = 60$ live `mcp-proxy` fault-injection traces, `ReceiptSupervisor` reaches 93.2% and 76.67% accuracy (96.72% and 96.15% recall; 10.16% and 38.24% FPR; +18.34 pp over Stage-1-only on live MCP traffic, $p = 0.001$) in 9.16–17.9 ms median CPU latency and 0 MiB VRAM, while `SQLiteProvenanceStore` and `mcp-proxy` block 100.0% of unauthorized writes and tamper/replay faults. Future work targets user-configurable MCP tool-stem registration, cross-clause failure-negation propagation, Streamable HTTP proxying, and lightweight paraphrase clustering.

Data Availability, Acknowledgments, and Disclosures. Source code is MIT-licensed at `https://github.com/AlienatedAliens/epistemic-guard`; INT8 model weights (`Apache-2.0/MIT` upstream) and benchmark corpora (with SHA-256 digests and OpenTimestamps receipts on web, hazard, and NLI splits) are archived on Hugging Face (`https://huggingface.co/RollSolo`). We thank Martin Monperrus for `osaudit` [18] and Brennan Zambo for the open `AER-1` conformance corpus [25]. Sean Maxwell is founder of Roll Solo (Member of NVIDIA Inception); no external funding was received.

References

[1] Ezio Bartocci and Yliès Falcone, editors. *Lectures on Runtime Verification: Introductory and Advanced Topics*, volume 10457 of *Lecture Notes in Computer Science*. Springer, 2018.

[2] Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of SEQUENCES 1997*, pages 21–29. IEEE, 1997.

[3] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 380–388. ACM, 2002.

[4] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90. ACM, 2023.

[5] Guardrails AI Contributors. Guardrails AI: Adding guardrails to large language models. `https://github.com/guardrails-ai/guardrails`, 2024. Accessed: 2026-11-19.

[6] Pengcheng He, Jianfeng Gao, and Weizhu Chen. DeBERTaV3: Improving DeBERTa using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *International Conference on Learning Representations*, 2023.

[7] Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and Madian Khabsa. Llama Guard: LLM-based input-output safeguard for human-AI conversations. arXiv:2312.06674, `https://arxiv.org/abs/2312.06674`, 2023.

[8] Simon Josefsson and Ilari Liusvaara. Edwards-curve digital signature algorithm (EdDSA). RFC 8032, Internet Engineering Task Force, 2017.

[9] Hugo Krawczyk and Pasi Eronen. HMAC-based extract-and-expand key derivation function (HKDF). RFC 5869, Internet Engineering Task Force, 2010.

[10] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, Internet Engineering Task Force, 2013.

[11] Martin Leucker and Christian Schallhart. A brief account of runtime verification. *The Journal of Logic and Algebraic Programming*, 78(5):293–303, 2009.

[12] Sean Maxwell. Closing the epistemic loop across the five layers of agent memory: Cryptographic provenance, paraconsistent bilattices, and Z3-quarantined procedural induction. Zenodo preprint, `https://doi.org/10.5281/zenodo.23901683`, 2026.

[13] Sean Maxwell. Cryptographic provenance gates mitigate multi-session memory poisoning in autonomous agents. Zenodo preprint, `https://doi.org/10.5281/zenodo.23901452`, 2026.

[14] Sean Maxwell. Provenance-gated memory and deontic output supervision for epistemically grounded autonomous agents. Zenodo preprint, `https://doi.org/10.5281/zenodo.23649049`, 2026.

[15] Sean Maxwell. Why speech-regex action supervisors miss live output: An empirical evasion taxonomy and held-out structured action contract evaluation. Zenodo preprint, `https://doi.org/10.5281/zenodo.23901605`, 2026.

[16] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[17] Model Context Protocol Contributors. Model Context Protocol specification (revision 2025-06-18). `https://modelcontextprotocol.io/specification/2025-06-18`, 2025. Accessed: 2026-11-19.

[18] Martin Monperrus. osaudit: Automated open-science audit. `https://github.com/ASSERT-KTH/osaudit`, 2026. Accessed: 2026-11-19.

[19] NabaOS Contributors. NabaOS: Sandboxed agent runtime with Ed25519-signed constitutions. `https://github.com/nabaos/nabaos`, 2026. Accessed: 2026-11-19.

[20] OWASP GenAI Security Project. OWASP Top 10 for Agentic Applications. `https://genai.owasp.org/`, 2026. Accessed: 2026-11-19.

[21] Pipelab Contributors. Pipelock: Open-source AI agent firewall for MCP security and mediator-signed action receipts. `https://github.com/pipelab-org/pipelock`, 2026. Accessed: 2026-11-19.

[22] Traian Rebedea, Razvan Dinu, Makesh Narsimhan Sreedhar, Christopher Parisien, and Jonathan Cohen. NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 431–445. Association for Computational Linguistics, 2023.

[23] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems*, volume 33, pages 5776–5788, 2020.

[24] Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

[25] Brennan Zambo. AER-1: A portable execution receipt for AI agent tool calls. IETF Internet-Draft `draft-zambo-aer1-26` (Work in Progress), `https://datatracker.ietf.org/doc/draft-zambo-aer1/26/`, 2026. Accessed: 2026-11-20.