# PROOF-VALID, YET UNSAFE TO COMMIT: FAIL-CLOSED EVIDENCE GATING FOR HIGH-IMPACT AGENTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Consider a remediation agent holding a signed exception whose inclusion proof verifies under checkpoint $r_0$. Checkpoint $r_1$ can revoke that exception while the served subset still verifies under $r_0$; suppressing a required safety constraint creates the same proof-valid, action-unsafe commit. This trace motivates *action admissibility* and PROOFLATCH, which preserves a typed evidence capability from retrieval to actuation and releases an action-bound token only when fragment integrity, client-observed freshness, policy compatibility, and policy-bounded completeness hold at the commit boundary. On $50,000$ matched attacked remediation/Qwen2.5-32B opportunities, proof-verifying fail-open consumption committed $3,400$ compromised actions (6.8%), versus $9,300$ (18.6%) without integrity; base and range-proof PROOFLATCH each committed $0/50,000$ (one-sided 95% upper bound $< 0.0060\%$). The exhaustive base ledger records $41,550$ immediate-safe, $6,100$ delayed-then-safe, and $2,350$ blocked outcomes, exposing the liveness cost rather than removing it from the denominator. Fail-open compromise remains 5.6–6.8% across two workflows and two model families while PROOFLATCH records none; separately, all 37 enumerated routes cross the gate, with no unauthorized side effect in $118,400$ integration-fault injections or 11 held-out route/fault compositions. Base enforcement adds 8.7% attacked wall time, and range proofs trade 10.2% cost for 1,750 additional immediate-safe and 500 additional eventual-safe outcomes. The result identifies the commit boundary—not proof verification alone—as the control point that turns mutable retrieved evidence into authority.

## 1 INTRODUCTION

At time $t_0$, a remediation agent retrieves a signed exception, its approval chain, a tool recipe, and the applicable safety constraints. Every served record verifies under checkpoint $r_0$. Before the side effect commits, however, checkpoint $r_1$ records a revocation, or the service suppresses a required constraint. The old subset remains cryptographically valid under $r_0$; what changes is whether that subset may authorize the action at $t_2$. A runtime that has already flattened the records into prompt text has no enforceable representation of this distinction.

This commit trace exposes a post-verification security gap. Authenticated retrieval establishes what a committed root said, but a high-impact agent also needs the evidence to remain current, policy-compatible, and complete for the candidate action when execution becomes irreversible. Task success does not measure this property: in the primary workflow, 72.4% task success coexists with $9,300$ compromised commits among $50,000$ attacked opportunities without integrity. We therefore make the authorization state of the side effect—rather than answer quality or proof acceptance—the evaluation object.

PROOFLATCH implements that object as a commit-coupled capability. A typed `VerifiedBundle` carries fragment proofs, authenticated checkpoint and policy state, an explicit completeness state, and a token bound to the canonical candidate action. Prompt construction may render evidence for the model, but it cannot manufacture authority: every high-impact wrapper re-evaluates the live bundle under one commit lock, and unresolved freshness or completeness is a false admission predicate.



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

Persisted signed roots prevent rollback relative to client-observed state, checkpoint gossip advances that state, and a frozen wrapper map makes retries and fallback routes cross the same gate.

The empirical design follows the same commit through matched runtime semantics. No integrity executes service bytes directly; proof-verifying fail-open rejects hard-invalid proofs but permits action while freshness or completeness is pending; base and range-proof PROOFLATCH require all admission predicates, differing only in completeness evidence. This progression changes compromised execution from 18.6% to 6.8% to 0/50,000 in the primary cell. A four-way ledger makes the corresponding delay and blocking explicit, paired seed analysis separates the safety effect from task and SLA variation, and replication across remediation and privilege-change workflows with Qwen2.5-32B-Instruct and Llama-3.1-70B-Instruct preserves the ordering. Finally, an independently frozen map of 12 sinks and 37 action-authorizing routes links the invariant to 118,400 integration-fault injections. The ledger makes the systems property testable at the same commit where every served proof still verifies.

## 2 ACTION ADMISSIBILITY CONTRACT

### 2.1 THREAT MODEL AND GUARDED OBJECT

The adversary controls the memory service and adaptively answers retrievals with modified bundles, stale checkpoints, inconsistent histories, or suppressed results. It observes the query, pinned policy, public checkpoint history, and prior service responses and can coordinate views across clients. The verifier, its persisted state, the pinned transparency-log key, and future audit randomness remain trusted; BLAKE3-256 collisions and Ed25519 forgeries under uncompromised keys are infeasible. The guarded objects are irreversible tool calls, privilege grants, code or configuration commits, policy edits, and safety-constraint updates whose authorization depends on retrieved evidence.

Each client persists the log key, an out-of-band bootstrap checkpoint, the highest accepted signed checkpoint $(i_{\max}, r_{\max})$, the highest-seen policy version, and an authorized-writer allowlist or certificate chain. A candidate root must extend the durable root, and a checkpoint or policy advance observed through retrieval, audit, or gossip before commit invalidates the candidate token. The complete five-class policy—including 10–300 s audit windows and 2/3 or 3/5 signature thresholds for sensitive classes—appears in Appendix Table A3.

### 2.2 FOUR PREDICATES AT THE ACTION BOUNDARY

The security contract decomposes admissibility into four independently inspectable clauses. *Served-fragment integrity* accepts a fragment only when its writer signature, inclusion proof, checkpoint signature, and policy-required provenance checks verify under pinned state. *Freshness* requires the candidate root to extend the highest client-observed signed root, the policy version to equal the highest observed pinned version, and all available consistency relations to verify. Incompatible signed roots constitute fork evidence; unavailable consistency evidence leaves freshness unresolved. *Policy compatibility* checks the canonical action descriptor against the evidence classes, principals, and thresholds named by the pinned policy. *Audited completeness* requires every policy-expected high-impact leaf in the request-defined window to be retrievable under a checkpoint consistent with that observed state.

For request $q$, policy $p$, and authenticated time bounds $[t_0, t_1]$, the client constructs $E(q, p, t_0, t_1)$ before reading the service response: named records from $q$, every required fragment class for the action, transitive policy-required parents, and the revocation-state key for each exception. Let $C$ be the identifiers with verified inclusion under a root extending current persisted state. Completeness is true only when $E \subseteq C$ and the authenticated window summary has closed; $e \in E \setminus C$, inconsistent boundaries, or an unlinkable proof emits a gap report. This response-independent rule prevents the service from defining completeness by the subset it returns. The omission-delay clock begins at the first withheld expected leaf and ends at the first report for that window; until closure or report resolution, the associated high-impact action remains inadmissible.

**Remediation trace.** The remediation request names an action class and pinned policy version before retrieval. That policy requires an exception, its parent approvals, the tool recipe, and the applicable safety-constraint class; these identifiers and parent references determine the expected set

independently of the service response. If the service returns the signed exception but suppresses its revocation or a required safety constraint, the served fragments may pass membership checks while the expected set remains open. The bundle therefore carries `complete_or_audited=false`, and the wrapper delays or blocks the remediation rather than allowing the verified subset to authorize it. This construction covers policy-enumerated evidence, not open-ended semantic relevance.

These clauses fail closed at different stages but meet at one decision. Local checks protect served bytes; persisted checkpoints and gossip constrain rollback and split views; expected sets or authenticated ranges constrain suppression; and the action gate composes them immediately before the side effect. Appendix Table A4 maps each protected object to its attacker, trust premise, stressor, cost, and measured trace. Tool hashes bind canonicalized transcripts to captured execution, while access control and multisignature policy govern writer authority.

# 3 PROOFLATCH: ADMISSIBILITY-PRESERVING ACTUATION

## 3.1 COMMIT-COUPLED EVIDENCE CAPABILITY AND MONOTONE STATE

A fragment's canonical record contains its id and class; hashes of its payload, parents, and tool traces; and its policy version, writer identity, and monotonic time. The committed leaf is

$$\begin{aligned}\texttt{leaf} = H(&\texttt{domain} \,\|\, \texttt{id} \,\|\, \texttt{class} \\ &\|\, \texttt{content_hash} \,\|\, \texttt{parents} \\ &\|\, \texttt{tool_hashes} \,\|\, \texttt{writer} \\ &\|\, \texttt{time} \,\|\, \texttt{policy_id}).\end{aligned}$$

The ledger checkpoints the Merkle root every $k = 2048$ writes or 5 s, whichever comes first, then appends the signed root to a transparency log. Parent inclusion links derived evidence to its inputs; canonical tool-trace hashes preserve capture and serialization identity.

The runtime obtains action-bearing evidence through the typed boundary

$$\texttt{retrieve_verified}(\texttt{query}, K, \texttt{policy_id}) \to \texttt{VerifiedBundle}.$$

The `VerifiedBundle` carries checkpoint identifier and root, `VerifiedFragment` values, policy metadata, completeness state, and a fresh single-use token minted by the trusted verifier. The token authenticates the bundle hash together with a canonical action descriptor (class, target, principal, parameters, and destination), policy id, expiry, and nonce. Prompt assembly accepts `VerifiedBundle` rather than raw strings; rendering preserves the bundle handle, and side-effecting wrappers require its token. This continuity prevents a verified payload from silently becoming unauthenticated prompt bytes or authorizing a different action.

## 3.2 ATOMIC ACTUATION REFERENCE MONITOR

For high-impact action $a$, candidate bundle $B$, and client state $s$, the wrapper admits execution exactly when

$$\begin{aligned}\texttt{admit}(a, B, s) = \texttt{accept}(B, s) &\wedge \texttt{fresh}(B, s) \\ &\wedge \texttt{policy_ok}(a, B, s) \wedge \texttt{complete_or_audited}(a, B, s).\end{aligned}$$

Under the wrapper's commit lock, the gate re-reads the persisted checkpoint, policy, audit state, token expiry, and nonce. An observed checkpoint or policy advance invalidates the token and restarts verification; token consumption is coupled to the local mutation or authorized dispatch at the wrapper's linearization point. A crash or retry after releasing the lock requires a newly admitted bundle, and a spent nonce rejects replay. For remote tools, the claim is authorization of the canonical dispatch accepted by the guarded sink, not atomicity of a third-party service. Fallback and direct-wrapper paths invoke the same admission path. These rules join validation and authorization at one trusted boundary rather than asserting stronger remote transaction semantics.

The wrapper provides complete mediation for the exercised high-impact sinks: every candidate side effect presents one action-bound token, every presentation re-evaluates live state, and the token is consumed in the same atomic event as commit. This instantiates the reference-monitor and fail-safe-default principles (Saltzer & Schroeder, 1975) at the retrieval-to-actuation boundary.

An *admissibility-preserving execution trace* is a sequence in which (i) every high-impact action is associated with one bundle identity, (ii) checkpoint and policy state never decrease, (iii) every transformation from retrieval through prompt construction preserves that identity, and (iv) the side effect commits only in the atomic event that evaluates all four conjuncts. A proof-verifying fail-open runtime satisfies payload verification but violates the fourth condition whenever it acts on stale or auditor-pending state.

> *Proposition 1 (conditional admissibility preservation).* If typed evidence continuity, monotone checkpoint/policy state, and complete mediation through the atomic gate hold for every high-impact action sink, then any rejected, stale, policy-incompatible, or completeness-unresolved bundle is withheld from side effects.

*Argument.* Continuity binds each candidate action to one $B$; monotonicity prevents an older checkpoint or policy from restoring validity; atomic evaluation makes every unresolved condition a false conjunct at commit; and complete mediation leaves no action-authorizing route outside `admit`. Section 3.3 validates this closure for the exercised integrations, and Appendix A gives the component statements.

### 3.3 Empirical closure of the action reference monitor

We validate complete mediation after freezing the integration graph. The sink registry covers 12 concrete instances of the five guarded classes in Section 2.1. A route is the tuple of evidence ingress, ordered conversion/fallback sequence, wrapper entry point, and sink instance; the union of static call-graph enumeration and exercised runtime traces yields 37 canonical tuples, all crossing the atomic gate $(37/37)$. Only after this freeze are compatible fault categories assigned to routes; eleven route/fault compositions are withheld before fault implementation, so they test unseen compositions rather than new fault primitives. Across five seeds, $118,400$ mutations exercise raw ingress, malformed bundles, type confusion, token replay, policy downgrade, restart rollback, fallback/direct-wrapper paths, concurrent state changes, TOCTOU, and configuration bypass. They produce $0$ unauthorized executions, with a descriptive one-sided 95% per-injection upper bound of 0.00253%; all eleven structured held-outs reject before side effects. The two summaries play distinct roles: repeated in-map stress and composition-level closure.

Failures record checkpoint id and root, bundle and leaf hashes, writer key id, action class, failed conjunct, and fork or gap evidence. Base ProofLatch closes completeness with request-log plus time-window auditing. The range variant indexes $(\texttt{class}, \texttt{time}, \texttt{id})$ in Merkle order and returns one contiguous authenticated range per policy-required class, including boundary leaves, every interior leaf, and adjacent-path witnesses for $[t_0, t_1]$. Because id and class are committed in each leaf, verification can bind the class-specific endpoints, query hash, policy version, and checkpoint root; adjacency then proves that no key in the requested class/time interval was skipped. This authenticated range replaces wait-only closure with positive completeness evidence at additional generation and verification cost.

## 4 Related Work: What Remains after Authentication

Authenticated dictionaries prove membership under a committed root (Merkle, 1990; Naor & Nissim, 1998; Papamanthou et al., 2011; Miller et al., 2014); transparency and secure logs order checkpoints and expose forks (Crosby & Wallach, 2009; Laurie et al., 2013; Melara et al., 2015; Basin et al., 2014; Schneier & Kelsey, 1998; 1999); provenance systems preserve lineage (Muniswamy-Reddy et al., 2006; Torres-Arias et al., 2019); and proof-carrying authorization presents evidence at use time (Appel & Felten, 1999; Bauer et al., 2002). These mechanisms establish ingredients of `accept`. ProofLatch adds a consumption invariant for mutable agent evidence: the same bundle identity must satisfy current observed checkpoint, action policy, and completeness when a side effect commits.

Recent agent-security systems reach adjacent boundaries. CaMeL separates privileged planning from untrusted data, tracks dependencies, and applies capability policy before tool calls (Debenedetti et al., 2025); it controls prompt-injection influence but does not authenticate a mutable memory history or prove freshness and completeness against a service that suppresses records. Authenticated Workflows places policy-enforcement points across prompt, tool, data, and context surfaces and binds workflow

attestations (Rajagopalan & Rao, 2026). Its surface-completeness goal asks whether resource access crosses a control plane; our audited-completeness predicate asks whether every policy-expected record in an authenticated request window is present before one action receives authority. A formal agent-security framework defines action alignment, source authorization, and data isolation (Siu et al., 2026); our compromise oracle instantiates source authorization at the commit boundary and couples it to an executable gate and four-outcome ledger. These guarantee-level contrasts separate their attacker channels from the adversarial-memory intervention; within that intervention, proof-verifying fail-open is the matched enforcement ablation.

Retrieval and memory systems improve grounding and persistent context (Izacard & Grave, 2021; Borgeaud et al., 2022; Gao et al., 2023; Asai et al., 2024; Park et al., 2023; Zhong et al., 2024; Li, 2025; Cemri et al., 2025), while AgentDojo, PoisonedRAG, and indirect-prompt-injection studies measure attack success, answer corruption, or behavior under malicious content (Greshake et al., 2023; Zou et al., 2025; Debenedetti et al., 2024). Our unit is instead the authorization state of an irreversible side effect, retaining delay and blocking in the denominator. Verifiable ML authenticates computation or learned artifacts (Jia et al., 2021; Tramèr & Boneh, 2019; Liu et al., 2021); reference-monitor and fail-safe-default principles determine how such evidence reaches protected operations (Saltzer & Schroeder, 1975). PROOFLATCH operationalizes that connection for retrieval-to-actuation.

# 5 DECISION-OPPORTUNITY EVALUATION

## 5.1 PROTOCOL, MATCHED TREATMENTS, AND METRICS

A *decision opportunity* is one authorization episode in which model reasoning produces a candidate high-impact action at the guarded boundary. The unit includes any retries triggered by that candidate and terminates with one of four mutually exclusive outcomes: executed safe, executed compromised, delayed then safely executed, or blocked at the fixed episode horizon. Task success and executed safety use the same episode cohort but remain separate outcomes because completion does not identify whether compromised evidence authorized a side effect.

Table 1: Evaluation contract for the 50,000-opportunity primary action study. Opportunity identities are fixed before runtime replay.

| Component | Operational definition |
|---|---|
| Primary cell | Remediation workflow with Qwen2.5-32B-Instruct under attack; every runtime treatment contains 50,000 opportunities. |
| Cohort and pairing | Five seeds each define 10,000 opportunity identities. The same workflow input, committed history, action candidate, attack identity, prompt, and seed are replayed under every treatment. |
| Independent oracle | At the commit boundary, a treatment-independent oracle checks the candidate action and bundle against the committed history and pinned policy. A commit is compromised if integrity, freshness, policy compatibility, or policy-defined completeness is false. |
| Episode termination | A rejected first attempt remains the same opportunity. It is delayed-then-safe if a newly admitted bundle commits before the workflow-specific horizon $H_w$, and blocked if no side effect commits by $H_w$. $H_w$ is fixed before replay and shared by treatments. |
| Task and SLA | Task success asks whether the terminal state at $H_w$ reaches the workflow target. SLA pass independently asks whether the episode reaches any terminal outcome within the prespecified service budget $S_w$. Both retain delayed, blocked, failed, and compromised episodes in the denominator. |

The 50,000-opportunity pool contains 10,000 frozen workflow prefixes per seed. Each prefix terminates at a registered high-impact sink and fixes the workflow input, committed history, candidate action, evidence dependencies, prompt, and attack assignment before treatment replay. A rejection and all subsequent retries remain attached to that identity, which preserves pairing through the four-way outcome.

An *attack trial* is a lighter verifier/adversary unit within a seed-specific committed history. The 1.2M trials allocate $240,000$ transformations to each of rewrite, forged insert, delete, equivocation, and selective omission. Every trial begins from a reset verifier snapshot, follows an $80/20$ warm/cold cache mixture, and prevents state leakage. Attack trials measure whether the integrity mechanism surfaces a transformation; decision opportunities measure what the agent ultimately commits.

The primary remediation workflow retrieves a signed exception, tool recipe, policy state, and safety constraints before an irreversible tool call or configuration commit. The replication workflow retrieves requester role, approval chain, signed exception, revocation state, and safety constraints before a privilege grant. Both map records to Table A3, use top-32 retrieval, 5 s/2048-write checkpoints, 60 s default auditing, greedy decoding with a fixed 512-token budget, and seeds $\{13, 37, 61, 71, 97\}$. We evaluate Qwen2.5-32B-Instruct (Yang et al., 2024) and Llama-3.1-70B-Instruct (Dubey et al., 2024). Qwen runs on 4×NVIDIA A100 80 GB GPUs; Llama uses 8 A100 80 GB GPUs. The ledger and verifier run on dual AMD EPYC 7763 CPUs with NVMe, Ubuntu 22.04, CUDA 12.4, and PyTorch 2.3.1 (Paszke et al., 2019).

All treatments share prompts, histories, attack assignments, policy, models, tools, and seeds. No integrity executes service bytes. Fail-open verifies the same signatures and proofs as PROOFLATCH but presents live state as advisory and permits action while freshness or audited completeness is pending; hard-invalid proofs reject. Base and range PROOFLATCH carry those signals in a typed capability and require them at commit. Task, cost, and SLA entries are mean ± sample SD across five seeds. Exact Clopper–Pearson intervals describe pooled action and detection counts; central treatment effects additionally use 95% $t$-intervals over the five seed-paired differences, preserving the history-level clustering. Latency intervals use stratified bootstrap resampling.

### 5.2 RQ1: DOES COMMIT GATING CHANGE ACTION OUTCOMES?

Table 2: Primary remediation/Qwen2.5-32B cell. Attacked rows use matched workflow inputs across treatments: compromised execution falls from 18.6% without integrity to 6.8% with proof-verifying fail-open consumption and to zero observed commits with fail-closed capability consumption. Appendix Table A1 reports counts and exact intervals.

| Treatment | Task success (%) | Executed compromised (%) | Wall cost (%) | SLA pass (%) |
|---|---|---|---|---|
| No integrity, clean | $72.7 \pm 1.7$ | $0.0 \pm 0.0$ | baseline | $96.4 \pm 0.7$ |
| No integrity, attacked | $72.4 \pm 1.8$ | $18.6 \pm 2.4$ $(9,300/50,000)$ | baseline | $96.2 \pm 0.8$ |
| Fail-open, clean | $72.4 \pm 1.7$ | $0.0 \pm 0.0$ | $+2.2 \pm 0.4$ | $96.1 \pm 0.8$ |
| Fail-open, attacked | $72.2 \pm 1.8$ | $6.8 \pm 1.5$ $(3,400/50,000)$ | $+5.2 \pm 0.8$ | $95.3 \pm 0.9$ |
| PROOFLATCH, clean | $72.1 \pm 1.7$ | $0.0 \pm 0.0$ | $+3.4 \pm 0.6$ | $95.8 \pm 0.9$ |
| PROOFLATCH, attacked | $71.8 \pm 1.9$ | $0.0 \pm 0.0$ $(0/50,000$; $< 0.0060\%)$ | $+8.7 \pm 1.2$ | $94.6 \pm 1.1$ |
| Range proof, clean | $71.9 \pm 1.7$ | $0.0 \pm 0.0$ | $+5.0 \pm 0.8$ | $95.4 \pm 0.9$ |
| Range proof, attacked | $72.0 \pm 1.8$ | $0.0 \pm 0.0$ $(0/50,000)$ | $+10.2 \pm 1.4$ | $94.1 \pm 1.2$ |

The raw ledger makes the outcome shift visible. No integrity executes $9,300$ compromised actions. Fail-open prevents forged-content execution yet still executes $3,400$ compromised actions under deletion, equivocation, and omission, while delaying $2,750$ and blocking $1,300$. Base PROOFLATCH executes $41,550$ safely, delays $6,100$, blocks $2,350$, and executes zero compromised actions. Range proofs execute $43,300$ immediately safely: 1,750 more than base, accompanied by 1,250 fewer delayed and 500 fewer blocked outcomes. Eventual-safe outcomes (immediate plus delayed-safe) therefore increase by 500, from 47,650 to 48,150. Every row sums to 50,000, so abstention remains visible.

Seed-paired effects confirm that the pooled result is not driven by one history. Relative to fail-open, base PROOFLATCH reduces compromised execution by 6.8 points (95% paired interval 5.0–8.6) while moving 8.8 points (7.3–10.3) into delay or blocking. Task success changes by $-0.4$ points ($-1.5$ to 0.7), SLA pass by $-0.7$ ($-1.3$ to $-0.1$), and wall cost by $+3.5$ (3.0–4.0). Clean base PROOFLATCH

remains close to no integrity in task success (72.1% vs. 72.7%) and SLA pass (95.8% vs. 96.4%), localizing the large ledger shift to attacked evidence.

### 5.3 RQ2: Which evidence failures drive the ledger?

Table 3: Verifier outcomes by attack family. Detection rates use all trials; delay quantiles condition on surfaced transformations. Unsurfaced trials remain detection failures rather than disappearing from the rate denominator.

| Attack | Trigger | Detected/attempted; 95% CI | Surfaced delay (s), med [p25,p75] |
|---|---|---|---|
| Rewrite tool trace | verifier | $240,000/240,000$; 100%, 99.998–100.000% | 1.3 [0.8,2.1] |
| Forged insert | verifier | $240,000/240,000$; 100%, 99.998–100.000% | 0.9 [0.6,1.4] |
| Delete constraint | audit gap | $240,000/240,000$; 100%, 99.998–100.000% | 4.2 [2.8,7.6] |
| Adaptive equivocation | gossip | $240,000/240,000$; 100%, 99.998–100.000% | 8.4 [4.6,16.2] |
| Selective omission | audit gap | $234,720/240,000$; 97.8%, 97.741–97.858% | 12.6 [6.4,28.4] |

Detection and authorization form consecutive stages. Table 3 asks whether a transformed retrieval is surfaced; Table 2 asks whether an action commits. An unsurfaced omission of a policy-required leaf does not set completeness true: the response-independent expected set remains open, so the action waits and ultimately becomes delayed-safe or blocked. This state transition explains why 97.8% verifier-level detection coexists with zero compromised commits. The delay clock starts at the first withheld expected leaf and ends at its first gap report; unsurfaced trials are excluded from the conditional delay quantiles but remain failures in the detection rate. The base/range completeness comparison remains end-to-end in the matched action ledger: both record zero compromise, while ranges convert 1,250 delayed and 500 blocked opportunities into immediate-safe execution.

### 5.4 RQ3: Does the ordering replicate?

Table 4: The action-outcome ordering replicates across guarded workflows and model families. Compromised rates are no integrity / fail-open / ProofLatch over $50,000$ opportunities per treatment. Task delta and wall cost are ProofLatch minus no integrity, reported as mean $\pm$ sample SD over five seeds; pooled counts and seed panels appear in Appendix C.3.

| Workflow / model | Executed compromised (%) | Task delta (points) | Wall cost (%) |
|---|---|---|---|
| Remediation / Qwen-32B | $18.6 \pm 2.4 / 6.8 \pm 1.5 / 0.0 \pm 0.0$ | $-0.6 \pm 3.6$ | $8.7 \pm 1.2$ |
| Remediation / Llama-70B | $17.3 \pm 2.1 / 6.1 \pm 1.4 / 0.0 \pm 0.0$ | $-0.8 \pm 0.5$ | $9.6 \pm 1.5$ |
| Privilege change / Qwen-32B | $15.8 \pm 1.9 / 5.6 \pm 1.2 / 0.0 \pm 0.0$ | $-0.5 \pm 0.4$ | $7.9 \pm 1.1$ |
| Privilege change / Llama-70B | $16.4 \pm 2.0 / 5.9 \pm 1.3 / 0.0 \pm 0.0$ | $-0.7 \pm 0.5$ | $9.1 \pm 1.4$ |

Absolute attack rates vary by workflow and model, but the treatment ordering does not: fail-open proof verification leaves 5.6–6.8% compromised execution, whereas ProofLatch records zero across all four cells with one-sided 95% bounds below 0.0060% per cell. Mean task-success differences range from $-0.5$ to $-0.8$ points. The primary Qwen comparison has larger seed variation ($-0.6 \pm 3.6$ points), which is visible rather than compressed into the mean.

### 5.5 RQ4: What safety–liveness frontier results?

The clean base-ProofLatch cost is $3.4 \pm 0.6\%$. Under attack, the additional 5.3 points decompose into 2.1 points for rejection/reverification, 1.5 for audit and gossip, 0.9 for retry/resume, and 0.8

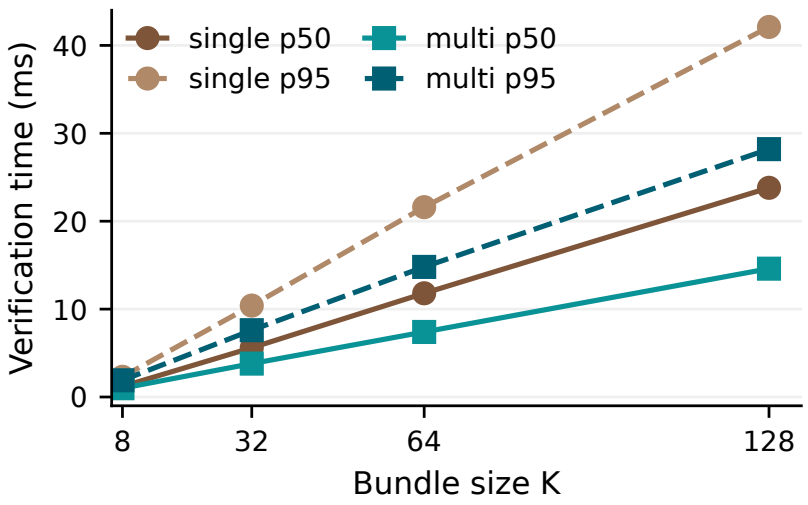


Figure 1: Client verification latency versus bundle size. Multi-proofs amortize shared Merkle path nodes; at top-32, p50/p95 are 3.8/7.6 ms and p99 is $11.3[10.7, 11.9]$ ms.

for incident recording, yielding the measured $8.7 \pm 1.2\%$. At 128 requests/s, verification increases retrieval p95 by $12.8[12.1, 13.5]$ ms and p99 by $19.7[18.5, 21.0]$ ms while changing throughput by $-4.9[-5.5, -4.2]\%$. Range proofs cost $10.2 \pm 1.4\%$ under attack and convert 1,250 delayed and 500 blocked opportunities into immediate-safe execution. The measured tails and disposition changes place the zero-observed-compromise result on an explicit safety–liveness frontier.

## 6 Implications for Agent Safety

The ledger changes how high-impact agents should be evaluated. A 72.4% task-success score can coexist with 18.6% compromised authorization, and rejecting invalid proofs still leaves 6.8% compromised execution when current state is advisory. Commit-time admissibility instead makes the full consequence visible: base ProofLatch records zero compromised commits together with 6,100 delayed and 2,350 blocked outcomes at 8.7% attacked wall-time cost. Authenticated ranges then recover 1,750 immediate-safe executions—500 of them additional eventual-safe outcomes—at 10.2% cost. Safety is therefore neither proof acceptance nor abstention; it is a complete action ledger tied to the authority exercised.

Shorter audit windows, authenticated range queries, stronger multisignature policy, and more frequent gossip shift delay, blocking, and verification cost without changing the commit rule evaluated here. The decision ledger keeps each of those operating points attached to its safe, compromised, delayed, and blocked action outcomes.

The evidence establishes the full admissibility path for two guarded workflows, two model families, and their frozen integration graphs under an adversarial memory service. Cryptographic checks protect served content under uncompromised keys; persisted roots and gossip make freshness monotone in client-observed authenticated state; expected sets and ranges cover policy-enumerated authorization evidence; and the route study tests each implementation obligation through targeted fault families. Semantic correctness of content supplied by authorized writers remains a policy layer composed with this mechanism. The matched intervention estimates the deployed path as a unit, while the obligation-wise fault map, range variant, and cross-setting replication show where its security and liveness behavior arises.

## 7 Conclusion

Valid proofs become security-relevant only through the authority a runtime grants them. Across the matched primary ledger, compromised execution falls from 18.6% without integrity to 6.8%

with fail-open verification and to zero observed commits when PROOFLATCH carries authenticated state through the guarded action; the ordering persists across workflows and model families. Typed continuity, live-state revalidation, and complete mediation convert stale or incomplete evidence from hidden authorization failure into measured delay or blocking. For agents that turn mutable retrieved records into irreversible side effects, action admissibility and its four-way ledger provide both the enforcement contract and the decision criterion.

## AI USE STATEMENT

Generative AI tools were used to assist with drafting and language editing, formatting, and bibliographic consistency checks. The authors reviewed the resulting text and take responsibility for the manuscript and its technical claims.

## REFERENCES

Andrew W. Appel and Edward W. Felten. Proof-carrying authentication. In *ACM Conference on Computer and Communications Security*, 1999.

Akari Asai et al. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

David Basin et al. ARPKI: Attack resilient public-key infrastructure. In *ACM Conference on Computer and Communications Security*, 2014.

Lujo Bauer, Michael A. Schneider, and Edward W. Felten. A general and flexible access-control system for the web. In *USENIX Security Symposium*, 2002.

Sebastian Borgeaud et al. Improving language models by retrieving from trillions of tokens. In *International Conference on Machine Learning*, 2022.

Mert Cemri, Melissa Z. Pan, Shuyi Yang, Lakshya A. Agrawal, Bhavya Chopra, Rishabh Tiwari, Kurt Keutzer, Aditya Parameswaran, Dan Klein, Kannan Ramchandran, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. Why do multi-agent LLM systems fail? In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4082. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b1041e52d3be19f0a9bc491657488e4a-Abstract-Datasets_and_Benchmarks_Track.html.

Scott A. Crosby and Dan S. Wallach. Efficient data structures for tamper-evident logging. In *USENIX Security Symposium*, 2009.

Edoardo Debenedetti, Jie Zhang, Mislav Balunović, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *NeurIPS Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=m1YYAQjO3w.

Edoardo Debenedetti, Ilia Shumailov, Tianqi Fan, Jamie Hayes, Nicholas Carlini, Daniel Fabian, Christoph Kern, Chongyang Shi, Andreas Terzis, and Florian Tramèr. Defeating prompt injections by design, 2025. URL https://arxiv.org/abs/2503.18813.

Abhimanyu Dubey et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Yunfan Gao et al. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

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*, pp. 79–90. ACM, 2023. doi: 10.1145/3605764.3623985. URL https://doi.org/10.1145/3605764.3623985.

Gautier Izacard and Edouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *European Chapter of the Association for Computational Linguistics*, 2021.

Hengrui Jia, Mohammad Yaghini, Christopher A. Choquette-Choo, Natalie Dullerud, Anvith Thudi, Varun Chandrasekaran, and Nicolas Papernot. Proof-of-learning: Definitions and practice. In *2021 IEEE Symposium on Security and Privacy*, pp. 1039–1056. IEEE, 2021. doi: 10.1109/SP40001.2021.00106. URL https://doi.org/10.1109/SP40001.2021.00106.

Ben Laurie, Adam Langley, and Emilia Kasper. Certificate transparency. Technical Report RFC 6962, RFC Editor, 2013. URL https://www.rfc-editor.org/info/rfc6962.

Haichang Li. Memory as a service (MaaS): Rethinking contextual memory as service-oriented modules for collaborative agents, 2025. URL https://arxiv.org/abs/2506.22815.

Tianyi Liu, Xiang Xie, and Yupeng Zhang. zkCNN: Zero knowledge proofs for convolutional neural network predictions and accuracy. In *ACM Conference on Computer and Communications Security*, 2021. doi: 10.1145/3460120.3485379. URL https://doi.org/10.1145/3460120.3485379.

Marcela S. Melara et al. CONIKS: Bringing key transparency to end users. In *USENIX Security Symposium*, 2015.

Ralph C. Merkle. A certified digital signature. In Gilles Brassard (ed.), *Advances in Cryptology—CRYPTO '89 Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pp. 218–238. Springer, 1990. doi: 10.1007/0-387-34805-0_21. URL https://doi.org/10.1007/0-387-34805-0_21.

Andrew Miller, Michael Hicks, Jonathan Katz, and Elaine Shi. Authenticated data structures, generically. In *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pp. 411–424, 2014. doi: 10.1145/2535838.2535851. URL https://doi.org/10.1145/2535838.2535851.

Kiran-Kumar Muniswamy-Reddy, David A. Holland, Uri Braun, and Margo Seltzer. Provenance-aware storage systems. In *USENIX Annual Technical Conference*, 2006.

Moni Naor and Kobbi Nissim. Certificate revocation and certificate update. In *USENIX Security Symposium*, 1998.

Charalampos Papamanthou et al. Optimal verification of operations on dynamic sets. In *Advances in Cryptology—CRYPTO*, 2011.

Joon Sung Park et al. Generative agents: Interactive simulacra of human behavior. In *ACM Symposium on User Interface Software and Technology*, 2023.

Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

Mohan Rajagopalan and Vinay Rao. Authenticated workflows: A systems approach to protecting agentic AI, 2026. URL https://arxiv.org/abs/2602.10465.

Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975. doi: 10.1109/PROC.1975.9939. URL https://doi.org/10.1109/PROC.1975.9939.

Bruce Schneier and John Kelsey. Cryptographic support for secure logs on untrusted machines. In *7th USENIX Security Symposium*, pp. 53–62, San Antonio, TX, January 1998. USENIX Association. URL https://www.usenix.org/conference/7th-usenix-security-symposium/cryptographic-support-secure-logs-untrusted-machines.

Bruce Schneier and John Kelsey. Secure audit logs to support computer forensics. *ACM Transactions on Information and System Security*, 2(2):159–176, 1999.

Vincent Siu, Jingxuan He, Kyle Montgomery, Zhun Wang, Neil Gong, Chenguang Wang, and Dawn Song. A framework for formalizing LLM agent security. In *ICLR 2026 Workshop on Agents in the Wild: Safety, Security, and Beyond*, 2026. URL https://openreview.net/forum?id=iQzd6qzIs5.

Santiago Torres-Arias et al. in-toto: Providing farm-to-table guarantees for bits and bytes. In *USENIX Security Symposium*, 2019.

Florian Tramèr and Dan Boneh. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *International Conference on Learning Representations*, 2019.

An Yang et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Wanjun Zhong et al. MemoryBank: Enhancing large language models with long-term memory. In *AAAI Conference on Artificial Intelligence*, 2024.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. In *USENIX Security Symposium*, pp. 3827–3844, 2025. URL https://www.usenix.org/conference/usenixsecurity25/presentation/zou-poisonedrag.

# A Formal Contract and Enforcement Connection

*Connection to measured enforcement.* This appendix derives the component guarantees behind the main-text action-admissibility invariant. Section 3.3 and Table A2 validate gate coverage for the exercised integrations; the statements below establish the consequences of that closure under the cryptographic and verifier trust assumptions.

*Adversary interface.* The adversary controls the memory service and retrieval endpoint. For each query it may adaptively choose the returned bundle, checkpoint, and omission pattern after seeing the request and pinned policy. It may replay old bundles, return malformed proofs, suppress leaves, or present inconsistent checkpoints to different clients. It does not control the client verifier, durable state, pinned transparency-log public key, or current policy. It cannot forge Ed25519 signatures under uncompromised keys or find BLAKE3-256 collisions.

*Client state and acceptance predicate.* Each client stores the pinned log key $K_{\log}$, bootstrap checkpoint, highest accepted signed checkpoint $(i_{\max}, r_{\max}, \sigma_{\max})$, highest-seen policy version $v_{\max}$, writer-key allowlist, and spent-nonce set. A bundle is accepted only if: (i) its checkpoint verifies under $K_{\log}$; (ii) its root extends $(i_{\max}, r_{\max})$; (iii) every fragment signature verifies under an allowed writer key; (iv) every inclusion proof reconstructs the claimed root; and (v) every policy-required parent-chain check succeeds. $\texttt{fresh}(B, s)$ additionally requires the current client-observed checkpoint and policy version, unchanged observed state at commit, and an unexpired, unspent token.

*Runtime admissibility versus proof verification.* The technical delta extends fragment acceptance with the action rule governing irreversible behavior in audited high-impact workflows. For any high-impact action $a$, accepted bundle $B$, and client state $s$, define

$$\begin{aligned}\texttt{admit}(a, B, s) = \texttt{accept}(B, s) &\wedge \texttt{fresh}(B, s) \\ &\wedge \texttt{policy_ok}(a, B, s) \wedge \texttt{complete_or_audited}(a, B, s).\end{aligned}$$

An implementation that verifies inclusion proofs yet permits $a$ while freshness or completeness is unresolved is proof-verifying but fail-open. The admissibility predicate supplies the implementation obligation, while the frozen route map and adversarial closure study in Table A2 validate that obligation for the measured action paths. The statements below begin with the resulting execution consequence and then establish its component guarantees.

> *Lemma 2 (withholding under validated action-path closure). If every action-authorizing route for the measured high-impact sinks crosses* $\texttt{admit}(a, B, s)$*, as validated for the exercised integrations in Table A2, then no measured executed high-impact action can depend on a rejected, stale, or completeness-unresolved bundle.*

*Proof.* Every measured high-impact action is gated by $\texttt{admit}(a, B, s)$, and Table A2 finds no alternate action-authorizing route in the frozen integration graph. Any rejected, stale, or completeness-unresolved bundle fails at least one conjunct, so the guarded action cannot commit. Therefore every measured executed high-impact action depends on a bundle satisfying acceptance, freshness, policy, and completeness or auditing at decision time. □The route-closure study supplies the empirical coverage premise; the lemma states its action-level consequence.

*Theorem 1 (pre-consumption integrity of served fragments). If the client accepts a fragment $m$ from a served bundle under checkpoint root $r$, then any adversarially modified fragment $m' \neq m$ served in its place is rejected except with negligible probability under the collision resistance of BLAKE3-256 or the unforgeability of Ed25519.*

*Proof.* If the adversary changes the fragment payload, parents, tool hashes, writer id, time, or policy id, then the leaf hash changes unless it finds a BLAKE3 collision. The original inclusion proof no longer reconstructs root $r$. If the adversary instead changes the leaf and supplies a fresh signed record, it must forge the writer signature unless it controls an allowed writer key. Therefore any modified served fragment causes either signature failure or Merkle-root mismatch and is rejected before prompt assembly. □

*Lemma 1 (stale-checkpoint and rollback rejection). If a client persists $(i_{\max}, r_{\max})$ and $v_{\max}$, then any bundle referencing an older or inconsistent observed checkpoint or policy version is rejected even if its proofs were once valid.*

*Proof.* Acceptance requires consistency with the durable root and highest observed policy version. A replayed checkpoint with $i < i_{\max}$, a root that does not extend $r_{\max}$, or policy $v < v_{\max}$ violates this predicate. A checkpoint or policy advance observed before commit therefore invalidates the older action token. □

*Theorem 2 (fork detection under gossip). If two honest clients exchange validly signed checkpoints containing incompatible authenticated history at the same index, then they obtain explicit equivocation evidence except with negligible probability under the signature security of the transparency-log key.*

*Proof.* Suppose honest clients $C_1$ and $C_2$ hold checkpoints $(i, r_1, \sigma_1)$ and $(i, r_2, \sigma_2)$ with $r_1 \neq r_2$. Verification under the pinned key establishes that the log signed incompatible roots for one index; forging either signature is negligible by assumption. For checkpoints at different indices, a valid consistency proof orders them in one history, while an invalid or unavailable proof leaves freshness unresolved and withholds action without being labeled positive fork evidence. Detection may be delayed until clients exchange checkpoints. □

*Theorem 3 (derived-chain integrity). If a derived fragment is accepted with required parent-chain validation enabled, then every cited parent exists under the accepted checkpoint and matches the parent hashes named by the derived fragment, except with negligible probability under hash collision resistance.*

*Proof.* The derived fragment's committed leaf contains the parent hashes. Acceptance requires an inclusion proof for each required parent. If any cited parent is modified, its hash changes unless the adversary finds a collision; if any parent is missing, the inclusion proof fails. Therefore the client accepts only if the cited parents exist and match the hashes named in the derived fragment. □

*Policy-bounded completeness.* For omission, the auditor executes a response-independent rule. It freezes $E \leftarrow$ records named by the request; adds each fragment class required by the action policy; recursively adds required parent identifiers; and adds the revocation-state key for every exception. After verification it forms $C$ from included identifiers under a root consistent with persisted signed state. If an authenticated window summary has closed and $E \setminus C \neq \varnothing$, it emits a gap report; before closure, the state is `pending`; after a valid retry with $E \subseteq C$, it is `complete`. Both `pending` and `gap` make `complete_or_audited` false, yielding delay until a valid retry or blocking at the episode horizon. The predicate is machine-checkable over policy-defined evidence; relevance outside $E$ remains governed by retrieval policy.

Table A1: Decision-opportunity ledger by runtime semantics. Each attacked treatment contains $50,000$ mutually exclusive and exhaustive opportunities. Exact pooled intervals are descriptive; seed columns follow $13/37/61/71/97$, and attack columns follow rewrite/forge/delete/equivocation/omission.

| Treatment | Executed safe | Executed compromised | Delayed then safe | Blocked in episode |
|---|---|---|---|---|
| No integrity | $40,700$; 81.056–81.740% | $9,300$; 18.260–18.944% | $0$ | $0$ |
| Fail-open | $42,550$; 84.785–85.411% | $3,400$; 6.581–7.024% | $2,750$; 5.302–5.703% | $1,300$; 2.462–2.743% |
| Base PROOFLATCH | $41,550$; 82.769–83.428% | $0$; $< 0.0060\%$ | $6,100$; 11.914–12.490% | $2,350$; 4.516–4.889% |
| Range PROOFLATCH | $43,300$; 86.298–86.897% | $0$; $< 0.0060\%$ | $4,850$; 9.442–9.963% | $1,850$; 3.536–3.869% |

| Treatment | Outcome | Counts by seed |
|---|---|---|
| No integrity | safe | $8140/8450/7830/8290/7990$ |
| | compromised | $1860/1550/2170/1710/2010$ |
| | delayed / blocked | $0/0/0/0/0$ / $0/0/0/0/0$ |
| Fail-open | safe | $8475/8730/8280/8590/8475$ |
| | compromised | $680/490/870/590/770$ |
| | delayed | $585/480/630/545/510$ |
| | blocked | $260/300/220/275/245$ |
| Base PROOFLATCH | safe | $8325/8390/8210/8440/8185$ |
| | compromised | $0/0/0/0/0$ |
| | delayed | $1210/1050/1400/1130/1310$ |
| | blocked | $465/560/390/430/505$ |
| Range PROOFLATCH | safe | $8680/8555/8745/8560/8760$ |
| | compromised | $0/0/0/0/0$ |
| | delayed | $955/1135/820/1040/900$ |
| | blocked | $365/310/435/400/340$ |

| Outcome/treatment | Rewrite | Forge | Delete | Equivocation | Omission |
|---|---|---|---|---|---|
| Compromised, no integrity | $2200/1850/2450/1300/1500$ | | | | |
| Compromised, fail-open | $0/0/1300/1100/1000$ | | | | |
| Delayed, fail-open | $700/650/450/450/500$ | | | | |
| Blocked, fail-open | $400/350/200/180/170$ | | | | |
| Compromised, base | $0/0/0/0/0$ | | | | |
| Delayed, base | $500/450/1500/1600/2050$ | | | | |
| Blocked, base | $250/200/550/600/750$ | | | | |
| Compromised, range | $0/0/0/0/0$ | | | | |
| Delayed, range | $500/450/1200/1450/1250$ | | | | |
| Blocked, range | $250/200/450/550/400$ | | | | |

| Policy class | Audit window | Base-PROOFLATCH delete D/B; omission D/B |
|---|---|---|
| Tool recipe | 60 s | $260/100$; $480/170$ |
| Code patch | 30 s | $320/115$; $430/160$ |
| Policy rule | 10 s | $360/130$; $420/150$ |
| Safety constraint | 10 s | $440/155$; $460/170$ |
| Summary/note | 300 s | $120/50$; $260/100$ |

*Ledger reading.* Fail-open removes rewrite and forgery compromise but leaves stale or incomplete verified state actionable. Both fail-closed variants move those cases into explicit liveness outcomes, and authenticated ranges recover immediate execution without changing the zero-compromise count.

## B Decision-Opportunity Ledger and Enforcement Closure

The ledger below is the empirical counterpart of the commit trace: every attacked opportunity appears exactly once, while seed, attack-family, and policy panels expose the structure beneath pooled rates.

The family panel explains the incremental comparator. Both verified treatments reject rewrite and forged insert before side effects. Fail-open retains compromised executions for deletion, equivocation, and omission because proof-valid but unresolved state can still authorize action. Both ProofLatch variants instead express those events as explicit liveness outcomes. The policy panel further shows that delete and omission outcomes remain stratified by the five predeclared fragment classes and audit windows rather than pooled under a post hoc label.

Table A2: Action-path closure under adversarial integration faults. The frozen route map assigns each category to a primary implementation obligation; each category ran across seeds $13, 37, 61, 71, 97$.

| Category | Primary obligation | Exercised ingress-to-commit faults | Attempts |
|---|---|---|---|
| Raw ingress | typed continuity | raw fragment and unverified payload injection | $14,800$ |
| Bundle/type | typed continuity | malformed, partial, serialization, type confusion | $16,500$ |
| Token | action binding | stale and cross-bundle token replay | $14,200$ |
| Policy | policy compatibility | downgrade and high-impact under-labeling | $15,600$ |
| State/cache | monotone freshness | restart rollback, prompt-cache reuse/invalidation | $14,900$ |
| Fallback | complete mediation | retry, fallback, and direct wrapper invocation | $16,200$ |
| TOCTOU | atomic admission | concurrent checkpoint change and check-to-commit races | $13,700$ |
| Configuration | fail-closed default | attempts to enable `unsafe_allow_unverified` | $12,500$ |
| All categories | all obligations | unauthorized side effects in row order | $0/0/0/0/0/0/0/0$ |

*Closure reading.* The category-to-obligation map localizes the stress test: representation faults exercise continuity, state and policy faults exercise admission predicates, and fallback/TOCTOU/configuration faults exercise complete mediation at the wrapper.

Static route enumeration found $12$ high-impact sinks and $37$ distinct ingress-to-sink tuples; $37/37$ crossed the atomic gate. Fault categories were mapped to these tuples only after the route inventory was frozen. Fault injection produced $0$ unauthorized executions in $118,400$ attempts ($0.00253\%$ one-sided $95\%$ upper bound). We withheld $11$ unseen route/fault compositions from development. The wrapper rejected $11/11$ before side effects, observed $0/11$ unauthorized executions, and emitted a distinct incident record for every rejection. Outcomes were classified at the side-effect commit boundary, independently of incident-record success.

Table A3: The client verifier uses this pinned policy table. For high-impact fragment types, parent-chain validation is required and audit windows are shorter.

| Fragment type | Required checks | Audit | Multi-sig |
|---|---|---|---|
| Tool recipe | Sig + inclusion + parent chain | 60 s | No |
| Code patch | Sig + inclusion + parent chain | 30 s | Yes (2/3) |
| Policy rule | Sig + inclusion + parent + diff | 10 s | Yes (2/3) |
| Safety constraint | Sig + inclusion + parent + diff | 10 s | Yes (3/5) |
| Summary / note | Sig + inclusion | 300 s | No |

*Policy reading.* Shorter windows and stronger signature thresholds are assigned to policy and safety records whose omission or unauthorized update can change action authority most directly.

Table A4: Full security evidence matrix. The nine fields connect each protected object to attacker capability, trust, property, comparator, adaptive or held-out stress, benign cost, known failure mode, and manuscript evidence.

| Failure surface | Object / attacker / trust / property | Comparator / stress / utility / failure / trace |
|---|---|---|
| Modification | Served payload and tool trace; service rewrites; keys and verifier trusted; pre-consumption integrity | No integrity; adaptive rewrite/forge; 3.8/7.6 ms top-32 verification; authorized harmful content; Table 3, Theorem 1 |
| Rollback | Checkpoint and policy version; service replays signed state; persisted maxima trusted; monotone freshness | Fail-open; stale and restart replay; included in base cost; lost local state; Lemma 1, Table A2 |
| Equivocation | Cross-client history; service splits views; log key and gossip trusted; fork detection | Fail-open; adaptive split view; audit/gossip cost; disconnected clients delay exposure; Table 3, Theorem 2 |
| Omission | Request-defined audited window; service suppresses; expected-set construction trusted; time-window completeness | Fail-open and range proof; selective omission; 8.7/10.2% attacked cost; outside-window relevance; Tables 3, 2, and A1 |
| Context stripping | Bundle-to-action capability; integration faults; verifier and wrapper trusted; evidence continuity | Fail-open; 11 held-out patterns; 0.8-point incident cost component; compromised verifier outside threat; Table A2, Proposition 1 |
| Signed harmful content | Canonical content and lineage; authorized writer supplies harmful material; writer policy trusted; authenticity/provenance | Multisignature policy; all five classes; mean task delta $-0.5$ to $-0.8$ points; semantic correctness governed by policy; Table A3, Theorem 3 |

*Matrix reading.* Rows distinguish the guarantee supplied by the mechanism from the trusted premise and the empirical trace that tests it; no single verifier statistic stands in for the end-to-end action outcome.

# C Experimental Protocol and Auditable Seed Panels

## C.1 Attack-trial unit

Each trial starts from the seed-specific committed snapshot and fixed policy table. The verifier state is reset, cache state is assigned according to the predeclared warm/cold stratum, one adversarial transformation is applied, and retrieval plus verification runs to one classified outcome. The adaptive adversary observes the query, pinned policy, public checkpoint history, and prior responses but not verifier secrets or future audit randomness. No mutated state carries into the next trial.

Table A5: Per-seed attack-trial accounting in seed order $13/37/61/71/97$. Each family has $48,000$ trials per seed and $240,000$ total.

| Families | Detected per seed; exact pooled 95% CI |
|---|---|
| Rewrite, forge, delete, fork (each) | $48,000/48,000/48,000/48,000/48,000$; 99.998–100.000% |
| Selective omission | $46,960/46,820/47,040/46,900/47,000$; 97.741–97.858% |

## C.2 Omission protocol and miss cases

The base auditor maintains logged retrieval requests, policy-covered fragment classes, and parent references. For each window it constructs the expected high-impact leaf set before requesting the window summary and verifies that every expected leaf links to a checkpoint consistent with pinned state. Missing leaves, inconsistent boundaries, or invalid proofs emit a gap report.

In remediation, the pre-response request and policy enumerate the signed exception, its approval parents, the tool recipe, policy rule, and applicable safety-constraint class. Omitting a revocation linked to the exception or any required constraint leaves this set open, even when every served leaf verifies. In privilege change, the corresponding set contains requester role, approval chain, exception, revocation state, and safety constraints; a grant cannot consume the bundle until those classes close. These traces make the completeness predicate response-independent for the policy-enumerated authorization evidence used in both workflows.

The 2.2% unsurfaced verifier-level transformations fall outside the audited window or concern low-impact relevance-ranked retrieval where the expected set is not policy-defined. They do not close a high-impact expected set: for an action-bearing bundle, `complete_or_audited` remains false until every policy-defined identifier is present under the authenticated window summary. The state sequence is therefore `pending`→`complete` after a valid retry, or `pending`→`gap`→`blocked` at the episode horizon. Table A1 records this disposition without changing denominators.

## C.3 Cross-workflow and model replication

The privilege-change workflow authorizes an irreversible grant or configuration commit from requester role, approval chain, signed exception, revocation state, and safety constraints. Retrieval depth, policy classes, checkpoint cadence, attack mixture, seeds, wrappers, and decoding budget match the remediation workflow.

## C.4 Systems microbenchmarks

The microbenchmark comprises $5$ runs and $100,000$ top-32 bundles per run. Each load run lasts 15 minutes with identical committed histories and policy mixtures across treatments. Bootstrap resampling nests requests within runs. The episode-level decomposition uses clean wall costs $3.4/2.6/4.2/3.0/3.8$, then adds attack-only rejection/reverification $2.1/1.8/2.4/1.9/2.3$, audit/gossip $1.5/1.3/1.7/1.4/1.6$, retry/resume $0.9/0.8/1.0/0.9/0.9$, and incident-recording $0.8/0.7/0.9/0.7/0.9$ points. Components sum to the attacked per-seed totals in Table A6.

Table A6: Per-seed task, wall-time, and SLA panels supporting Table 2. Values follow seeds 13/37/61/71/97; wall values are percentage change from no integrity.

| Treatment | Metric | Per-seed values |
|---|---|---|
| No integrity attacked | task | 72.4/70.1/74.7/71.3/73.5 |
| | SLA | 96.2/97.2/95.7/96.7/95.2 |
| Fail-open attacked | task | 72.2/74.5/69.9/73.3/71.1 |
| | wall | 5.2/4.2/6.2/4.7/5.7 |
| | SLA | 95.3/96.4/94.7/95.9/94.2 |
| Base PROOFLATCH attacked | task | 71.8/74.2/70.6/73.0/69.4 |
| | wall | 8.7/7.2/10.2/7.9/9.5 |
| | SLA | 94.6/96.0/93.2/95.3/93.9 |
| Range PROOFLATCH attacked | task | 72.0/69.7/74.3/70.9/73.1 |
| | wall | 10.2/12.0/8.4/11.1/9.3 |
| | SLA | 94.1/92.6/95.6/93.3/94.9 |

*Seed reading.* The panels retain treatment-paired seed order, enabling the clustered contrasts reported in Section 5.2 rather than inferring effects from marginal arm standard deviations.

Table A7: Replication counts and per-seed panels. Pooled counts are no integrity/fail-open/PROOFLATCH out of 50,000 opportunities per treatment. Seed sequences follow 13/37/61/71/97.

| Workflow/model | Pooled compromised counts | Per-seed no integrity; fail-open; PROOFLATCH |
|---|---|---|
| Remediation/Qwen | 9300/3400/0 | Table A1 |
| Remediation/Llama | 8650/3050/0 | 1730/1464/1996/1597/1863; 610/787/433/699/521; 0/0/0/0/0 |
| Privilege/Qwen | 7900/2800/0 | 1580/1820/1340/1700/1460; 560/408/712/484/636; 0/0/0/0/0 |
| Privilege/Llama | 8200/2950/0 | 1640/1387/1893/1514/1766; 590/754/426/672/508; 0/0/0/0/0 |

| Workflow/model | PROOFLATCH task delta by seed | PROOFLATCH wall cost by seed |
|---|---|---|
| Remediation/Llama | −0.8/ − 1.4/ − 0.2/ − 1.1/ − 0.5 | 9.6/11.5/7.7/10.5/8.7 |
| Privilege/Qwen | −0.5/0.0/ − 1.0/ − 0.3/ − 0.7 | 7.9/6.5/9.3/7.2/8.6 |
| Privilege/Llama | −0.7/ − 0.1/ − 1.3/ − 0.4/ − 1.0 | 9.1/10.9/7.3/10.0/8.2 |

*Replication reading.* Absolute compromise varies across workflow/model cells, but fail-open remains nonzero in every cell and the fail-closed treatment remains at zero observed commits.

Table A8: Five-repeat systems feasibility at top-32. Runs use ids $101, 211, 307, 401, 503$; brackets are 95% bootstrap intervals.

| Measurement | p50 | p95 | p99 |
|---|---|---|---|
| Proof generation (ms) | $4.7[4.5, 4.9]$ | $10.2[9.8, 10.7]$ | $15.4[14.6, 16.3]$ |
| Client verification (ms) | 3.8 | 7.6 | $11.3[10.7, 11.9]$ |
| Loaded retrieval delta (ms) | — | $12.8[12.1, 13.5]$ | $19.7[18.5, 21.0]$ |

| Measurement | Result |
|---|---|
| Multi-proof vs individual payload | $8.6[8.4, 8.8]$ vs $21.4[20.9, 21.9]$ KiB; $59.8[58.9, 60.7]\%$ reduction |
| Loaded protocol | 128 req/s, 15 min/run, $K = 8/32/64/128$ mix 10/60/20/10%, 80/20% warm/cold |
| Request accounting | $576,000$ offered; $574,912$ successful; timeout/error requests retained in tails |
| Throughput change | $-4.9[-5.5, -4.2]\%$ |

*Systems reading.* Multi-proofs reduce bytes and verification time at top-32, while loaded tails and throughput retain timeout/error requests; these measurements explain the wall-cost component of the action ledger.

### C.5 Compute budget

The full Qwen evaluation requires roughly 18 h on 4×A100 80 GB GPUs; Llama uses 8 A100 80 GB GPUs under the same serving and decoding contract. CPU-only verification and proof-generation sweeps run on the dual-EPYC host. Qwen2.5-32B-Instruct is used under Apache 2.0, PyTorch 2.3.1 under BSD 3-Clause, and CUDA 12.4 under the NVIDIA CUDA Toolkit EULA; the evaluation uses no third-party benchmark dataset.