Toward Trustworthy LLM Router Ecosystems: Incentive-Compatible Cryptographic Mitigations

Anonymous Authors[1]

Abstract

LLM API routers make access to frontier models convenient, but they also create a hidden trust problem: the router that forwards a request can observe prompts, credentials, and tool calls and can rewrite outputs before they reach the client. We argue that this is a structural security and accountability problem that detection alone cannot solve and that workable defenses must align with the incentives of users, routers, and providers. We therefore assemble a progressively deployable trust-reduction stack from existing mechanisms, with each phase designed to provide an immediate adoption benefit. We then implement a `LiteLLM`-compatible proof of concept and show that it preserves LLM router utility with acceptable overhead while enhancing user-side privacy.

1. Introduction

Large language models (LLMs) are increasingly used as infrastructure for research assistants, coding agents, enterprise automation, and public services. As these systems move from answering questions to taking actions—calling tools, accessing data, and executing workflows—their trustworthiness depends not only on model quality, but also on the systems that carry requests and enforce access control around them.

In many deployments, users do not connect directly to the model provider. Instead, requests pass through one or more LLM API routers that provide load balancing, model aggregation, failover, logging, and cost control. These services are operationally useful, but they also create a powerful and often hidden point of control: a router can read prompts and credentials in plaintext, observe tool schemas, and modify outputs before they reach the client.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Recent empirical studies show that these risks already occur in practice. A large-scale audit of commodity routers found that one in three paid routers were already passively draining researcher-owned credentials; one leaked upstream key reached 440 downstream sessions across 398 projects, exposing 13 GB of visible traffic, and one router silently drained cryptocurrency from a researcher's wallet (Liu et al., 2026). In March 2026, a supply-chain attack against `LiteLLM` (LiteLLM Team, 2026) showed how quickly router-layer compromise can spread across the ecosystem. A parallel audit of 17 shadow APIs found that nearly half served models that did not match the advertised identity (Zhang et al., 2026).

These incidents point to a structural problem in how LLM systems are deployed. When requests pass through hidden router chains, the user's trust boundary silently expands beyond the first authenticated hop (Liu et al., 2026; Wang et al., 2025). As a result, ordinary client-side safeguards are limited: they may catch some obviously dangerous tool calls, but they cannot reveal silent secret theft (Abdelnabi et al., 2023), and they can be bypassed by routers that behave normally during testing and attack only later (§2). Detection therefore remains useful, but it is not sufficient on its own for the attack classes we study.

The challenge is not purely technical. Users want confidentiality and auditability; routers may treat their upstream partners as sensitive business information; providers want to reduce credential abuse without accepting liability for downstream tool execution. Because these incentives diverge, many technically appealing proposals fail to deploy. A practical solution must therefore give each party a reason to adopt it while still improving security for everyone else. Moreover, any viable path must deliver Day-1 value to at least one stakeholder, coexist with unupgraded parties, and remain reversible.

For example, reducing what routers can observe and modify requires encrypting sensitive data while keeping encrypted requests compatible with the current ecosystem. The required building blocks already exist—including format-preserving tokenization and attenuable capabilities—but what is missing is a *deployment story* acceptable to users, routers, and providers.

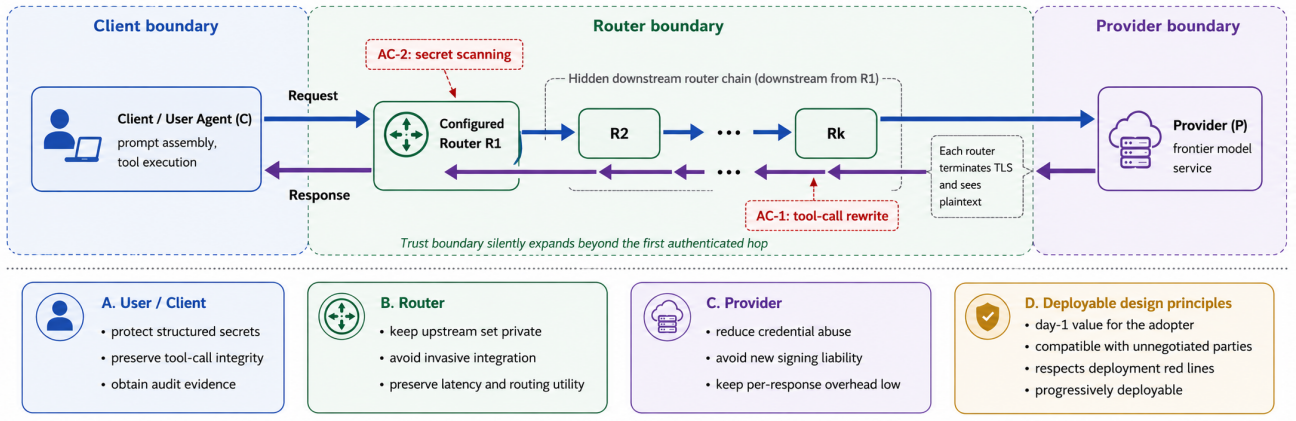

Figure 1. Overview of our deployable architecture for safer LLM routing across users, routers, and providers. The figure highlights how progressively adopted protections can reduce router visibility and constrain misuse while preserving compatibility with existing deployments.

This paper studies how to make LLM routing safer, more auditable, and more reliable in real deployments. Under these constraints, we make three contributions:

- **[Stakeholder analysis]** We identify the distinct interests of users, routers, and providers, and organize them into a stakeholder taxonomy that explains why the problem persists and what any deployable solution must satisfy (§3).
- **[Design proposal]** We adapt and compose existing cryptographic mechanisms from the literature into a progressively deployable routing architecture for this setting, even though those mechanisms were not originally designed for it (§4, §5, Appendix F).
- **[Experimental evaluation]** We implement a `LiteLLM`-integrated prototype and show that it improves privacy by reducing credential leakage while preserving low overhead on mock and live APIs (§6).

We do not propose new cryptographic constructions; the contribution is to assemble existing mechanisms into a deployable architecture and validate the client-side layer empirically.[1] Appendices G–E provide the specification, proof sketchs, and an extended mechanisms.

2. Background and Motivation

The router chain. A single agent request traverses three classes of entities. A *client* C (the user's agent runtime, e.g., `Claude Code`, `Cursor`, `Codex CLI`, or an enterprise agent framework) assembles a prompt, system context, and tool schema, sends them to its configured endpoint, and later executes returned tool calls. Some agent runtimes also expose an auto-approval setting (often called `YOLO` mode) in which returned tool calls run without per-action confirmation. A *router chain* $(R_1, \ldots, R_k)$ of one or more HTTP-layer proxies stands between the client and the *provider* P (the final model service). Each R_i terminates the inbound TLS session, reads the plaintext JSON body, and re-originates a fresh TLS session to R_{i+1}; the client's WebPKI validation extends only to R_1, so the downstream chain is invisible by construction. This placement gives any router the ability to inspect prompts, structured secrets, and returned tool-call JSON and, if malicious, rewrite responses while remaining protocol-compliant to adjacent hops. Published measurements report chain depths from one to five in the wild (Liu et al., 2026).

Trust bottleneck in the routers. The single most important fact about this ecosystem is negative: *no cryptographic integrity or confidentiality is enforced between the client and the upstream model provider*. Mainstream frameworks do not sign their responses; routers do not attest their upstream set; providers do not issue channel-bound short-lived capabilities. Authentication consists of long-lived bearer API keys that any intermediate router can read, log, replay, or resell. The router layer is not written per deployment either: a small number of open-source templates (`LiteLLM` (BerriAI, 2026), `sub2api` (Wei-Shaw, 2026), `new-api` (QuantumNous, 2026), `one-api` (songquanpeng, 2026)) are forked and re-deployed tens of millions of times, so any compromise at the template level (as in the `LiteLLM` March 2026 incident (LiteLLM Team, 2026)) propagates instantly, and changes at that layer can have ecosystem-scale consequences. Recent empirical work shows that the problem is already live. Liu et al. (2026) formalize and measure two attack families at ecosystem scale.

[1] "Incentive-compatible" means *individually rational under stated conditions* (positive expected payoff to the adopting party regardless of whether other parties have adopted), not the formal mechanism-design sense (dominant strategy or Bayesian Nash equilibrium); see Appendix D.

AC-1 rewrites tool-call responses while preserving a valid schema; the variants used later are AC-1.a (package-install typosquatting) and AC-1.b (conditional delivery). *AC-2* passively scans request and response bodies for structured secrets and exfiltrates them out of band without changing the client-visible payload. Zhang et al. (2026) audit 17 shadow APIs and show nearly half fail model-identity fingerprinting; and a broader systematization (Wang et al., 2025) finds that 50.3% of LLM-related vulnerabilities live in the application layer, which is exactly where routers live.

Detection alone cannot close the loop. Detection alone is *insufficient* for a specific but consequential subset of threats. Monitoring, logging, and behavioral screening remain valuable components of defense in depth, but they cannot close the loop from within client-observable traffic alone against adaptive and passive router attacks. In particular, AC-1.b is pivotal because it defeats testing-based defenses: a malicious router can behave benignly while being probed and deliver a malicious payload only when a hidden predicate fires, while AC-2 passively exfiltrates structured secrets without modifying the client-visible payload. More generally, the client sees only the final message that emerges from the router chain, not a verifiable record of what the provider originally sent. If an intermediate router rewrites a tool-call payload but keeps it syntactically valid, the client has no independent reference point against which to compare the result. Closing the loop therefore requires changing the structure of the channel rather than merely refining the detector: the client needs some way to limit what the router can observe or alter, or to verify what the provider actually sent.

3. Stakeholder Interests

We focus on users, routers, and providers because they are the only parties whose refusal can directly block deployment: users must trust the confidentiality and compliance story, routers must accept the business and topology disclosures implied by the mechanism, and providers must accept the credential and liability model it creates. Other stakeholders matter, but in this paper their requirements are mediated through one of these three actors; that is why prior router-transparency and provider-signing proposals stalled even when the cryptography itself was not the blocker.

Users (U). Users matter because they absorb the immediate security, execution-safety, and compliance consequences of whatever the routing path does to their prompts, credentials, and tool calls. Individual developers fear local credential leakage and rewritten tool calls executed under `YOLO` mode (Liu et al., 2026). Enterprise and vertical builders (SOC 2, GDPR, HIPAA) need audit evidence for every data flow; even a passive plaintext router scan over PHI-adjacent data may constitute an unauthorized disclosure under HIPAA §164.502 regardless of whether data leaves the chain (U.S. Department of Health and Human Services, 2013). The March 2026 incident (LiteLLM Team, 2026) illustrates the compliance gap when downstream subprocessors are unattested.

Routers (R). Routers matter because they sit on the request path and can therefore enforce, dilute, or bypass any mechanism we propose. Commercial aggregators depend on keeping their upstream provider graph private, since revealing it exposes the negotiation table and weakens their bargaining position; this is the principal reason "router transparency" proposals have failed to deploy. Enterprise internal gateways and compliance routers often want stronger cryptographic evidence because they answer to internal audit and policy teams, whereas gray-market resellers are not voluntarily bindable and will route around mechanisms that weaken arbitrage.

Providers (P). Providers matter because they control model access and credential issuance, so a mechanism that raises fraud risk, operational burden, or legal exposure can be refused at the API boundary. Providers want to reduce credential resale and model-identity fraud, but they emphatically do *not* want new legal exposure: cryptographically signing execution-relevant tool-call payloads would accept liability for actions like `rm -rf /` or `curl | bash` on an end-user machine, and provider ToS explicitly refuses that linkage. They do tolerate mechanisms that harden the credential layer using IETF RFCs already in production (OAuth 2.0 token exchange (Jones et al., 2020), DPoP (Fett et al., 2023)).

Interest matrix and four red lines. We surface these dimensions in the main text because they are where abstract trust-reduction proposals meet practical deployment constraints: **if a mechanism threatens router secrecy, expands provider liability, demands invasive framework changes, or adds noticeable response cost, it is unlikely to deploy regardless of its cryptographic appeal.** Additional dimensions and supporting evidence appear in Appendix D; here we focus on the dimensions most relevant to most common scenarios.

Across stakeholders, four **red lines** recur: forcing routers to reveal their upstream set, requiring providers to sign execution-relevant content, demanding invasive changes to agent frameworks, or adding more than 5% per-response cost. These constraints explain why technically attractive proposals, including per-response ZK-SNARK attestation, FHE provider inference, and all-to-all MPC, do not clear the adoption bar under current incentives, and why Section 4 asks which mechanisms actually *survive* these constraints

rather than enumerating everything cryptographically possible.

4. The Design Space of Trust-Reduction Mechanisms

Given the stakeholder red lines from Section 3, the question is not which trust-reduction mechanism looks strongest in isolation, but which mechanisms still reduce trust without crossing a deployment-blocking red line. Prior proposals drew from a narrow menu ("sign it," "log it," "put it in an enclave") and failed precisely because they ignored one or more of those adoption constraints. *Trusted execution environments (TEEs)* are an example: they are theoretically the strongest defense, but they fail D7 (no major LLM provider operates hosted inference inside a TEE), D8 (SGX enclave transitions add hundreds of milliseconds per call, well above the 5% P95 budget), and require provider hardware changes outside client or router control. Appendix E surveys roughly 40 mechanism families we eliminated; here we keep only the families that survive Table 6's red lines and therefore remain plausible under the incentive model established in the previous section.

Surviving mechanism families. We keep families that map cleanly to the two core router attacks from Section 2, AC-2 secret scanning and AC-1 tool-call rewriting, or that provide the provenance and accountability substrate needed to deploy those defenses. The result is a short list with a clear division of labor. At the client edge, *format-preserving tokenization* (FF1, NIST SP 800-38G (Dworkin, 2016)) is the cleanest deployable response to AC-2: it hides structured secrets from passive router scanning while preserving syntax, so downstream validators continue to operate. *Attenuable capability tokens* (macaroons (Birgisson et al., 2014), OAuth DPoP (Fett et al., 2023)) complement that defense by reducing the durable value of any credential a router does see or steal, replacing long-lived bearer secrets with short-lived, channel-bound authority. For router provenance, *anonymous source attestation* via ring signatures (Rivest et al., 2001) lets a router prove membership in an approved set without revealing which router handled the request, thereby preserving upstream-topology privacy; the construction can later upgrade to group signatures with an opening authority (Boneh et al., 2004) as accountability infrastructure matures. *Authorized PSI* (Freedman et al., 2004; Kolesnikov et al., 2016) complements that attestation story by proving that the hidden upstream set does not intersect a provider revocation list without publishing either set. For tool-call integrity, *policy-constrained transform signatures* (sanitizable (Ateniese et al., 2005), designated-verifier (Jakobsson et al., 1996)) are the cleanest bounded-rewrite response to AC-1: they let providers authorize constrained rewrites while letting clients reject tool-call changes outside that envelope, without creating public, execution-relevant signatures that would trigger new legal exposure. Finally, *transparency logs* (Laurie et al., 2013; Newman et al., 2022) are chosen not as a direct per-request defense, but as the ecosystem substrate that turns local proofs into auditable evidence for incident response and compliance. Table 1 then scores these candidate families against the stakeholder red lines and separates the core survivors from mechanisms that remain, at most, advanced options.

The deployable frontier. Taken together, the surviving families cluster into a small number of bundles, each aligned with a different stakeholder and a different security gain. FPE + request-scoped macaroon vault forms a *client-side* bundle lifting D1 (structured-secret confidentiality); OAuth DPoP forms a *provider-side* bundle lifting D10 (credential-abuse reduction); PSI with upstream-set commitment forms a *router-side* bundle lifting D4/D5 (auditable evidence and upstream-topology privacy); chameleon + DVS forms a *controlled-transform* bundle lifting D2/D3 without touching D9; transparency logs lift D4/D11 (auditable and regulatory evidence) at ecosystem scale. These bundles address complementary rows of Table 6, suggesting a *layered* rather than monolithic answer to the router trust gap and motivating the stack construction in the next section.

5. A Layered, Progressively Deployable Stack

Building on the deployable frontier in §4, we now assemble the surviving bundles into an explicit stack and deployment ladder (Table 2). The key design property is incremental deployability: each phase gives its initiating stakeholder a Day-1 benefit while remaining compatible with unupgraded parties. Phases 0 and 2 are fully unilateral, whereas Phases 1 and 3 require a minimum two-party bootstrap. Here, "unilateral" means the *initiating* party can begin deployment without simultaneous action from others; even so, Phases 1 and 2 still carry organizational prerequisites that remain within a single stakeholder's control envelope. The implemented contribution of this paper is Phase 0. Phases 1–4 therefore remain a **research agenda**—architectural protocol contributions grounded in stakeholder analysis, per-party rationality analysis, and mixed-deployment compatibility, with overhead claims validated by purpose-built benchmarks (§B.5)—whose novelty lies in the stakeholder-compatible composition and phase ordering. This ordering matters because gray-market resellers will not adopt the later, coordination-heavy layers, but Phase 0 still protects the client because it requires no router cooperation.

Phase 0 — client-only value protection. Deployed by the client with no cooperation from router or provider, using FPE (or structure-preserving tokenization) + a local macaroon vault. On each request, the middleware detects

Table 1. Mechanism scoring against stakeholder red lines. Surviving rows remain in scope because they either mitigate AC-2 secret scanning, mitigate AC-1 tool-call rewriting, or provide the provenance and accountability substrate needed to deploy those defenses. The screening dimensions are structured-secret confidentiality (D1), upstream-topology privacy (D5), zero-code integration (D7), latency overhead $\leq 5\%$ of baseline (D8), no new legal liability from signing execution-relevant content (D9), and credential-abuse reduction (D10). A mechanism with any $--$ in a red-line column is excluded from the core candidate set and treated, at most, as an advanced option. $++$ strong positive, $+$ positive, $\cdot$ neutral, $-$ small negative, $--$ red-line violation. Mechanisms in **bold** survive all red lines. "Locus" denotes the stakeholder or institutional layer where the mechanism most naturally sits (C=client, P=provider, R=router, TA=transform authority, E=ecosystem).

Mechanism	D1	D5	D7	D8	D9	D10	Locus
Full-response standard signatures	$+$	$\cdot$	$-$	$-$	$--$	$\cdot$	–
ZK-SNARK (per-response)	$\cdot$	$\cdot$	$--$	$--$	$\cdot$	$\cdot$	–
FHE / MPC at router layer	$+$	$\cdot$	$--$	$--$	$\cdot$	$\cdot$	–
FPE / tokenization	$++$	$\cdot$	$+$	$\cdot$	$\cdot$	$\cdot$	C
Macaroons / OAuth DPoP	$\cdot$	$\cdot$	$++$	$+$	$\cdot$	$++$	C/P
Ring signatures	$\cdot$	$++$	$\cdot$	$+$	$++$	$-$	R
Authorized PSI	$\cdot$	$+$	$+$	$\cdot$	$\cdot$	$+$	R
Chameleon / sanitizable	$\cdot$	$+$	$-$	$\cdot$	$\cdot$	$\cdot$	P+TA
Designated verifier sig.	$\cdot$	$+$	$-$	$\cdot$	$++$	$\cdot$	P
Transparency logs	$\cdot$	$\cdot$	$+$	$\cdot$	$\cdot$	$+$	E

high-value structured fields and replaces each with a token derived by FPE under a local key, with a fresh request identifier folded into the per-request tweak and vault record. The token is recorded in a local append-only log keyed to a macaroon capability whose caveats include that request identifier, an expiry window, and the allowed type labels. On the return path, the middleware detokenizes only against that live local request context; once the request completes or expires, the capability is revoked and the token map is discarded, so captured tokens do not remain durable bearer artifacts. A malicious router sees only the tokenized form; AC-2 regex and LLM-based scanners extract ciphertext rather than plaintext. Phase 0 protects secret *values* (D1) and yields a local audit trail (D4/D11 partial); it does *not* improve tool-call integrity or require any provider change. Vault key lifecycle and sidecar deployment guidance appear in Appendix H.

Phase 1 — router-initiated attestation. *Phase 1 is bilateral at minimum*: a ring of size one is a standard signature with no anonymity or membership attestation; a minimum viable ring requires at least two members. A pair of mutually contracting routers can begin ring-signing without simultaneous action from clients or providers (who ignore the optional header), but at least two parties must cooperate at bootstrap. In early deployment, the router uses a ring signature (Rivest et al., 2001) over a recognized router key set: no trusted setup, D5 fully preserved. As accountability infrastructure matures, the ring upgrades to a group signature (Boneh et al., 2004) with selective de-anonymization on complaint. Each response also attaches a per-session PSI proof (Freedman et al., 2004; Kolesnikov et al., 2016) that $U_R \cap V_{\text{revoked}} = \emptyset$ with the upstream set committed but not published (D5). *Estimated overhead*: ring-verify (16-member RSA-2048 ring) adds ≈ 0.8 ms per response at native-C speed; the attestation header adds ≈ 600 B wire overhead—both within the D8 budget. Phase 1 governance (bilateral anchor bootstrap, trajectory to consortium-backed group signatures) is detailed in Appendix F.

Phase 2 — provider-issued short-lived capabilities. Deployed by the provider using OAuth 2.0 Token Exchange (Jones et al., 2020) and DPoP (Fett et al., 2023)—no new cryptographic standards, only IETF RFCs already in production at Microsoft, Google, and Meta. Leaked credentials no longer yield durable abuse capacity; a working reference implementation (337 LOC, `phase2_dpop.py`) over EC P-256 measures 0.062 ms proof-generation overhead and a $108{,}745\times$ reduction in abuse window vs. long-lived bearer tokens. Routers that do not understand capabilities fall back to the existing bearer-token path (D7). Phase 0 + Phase 2 is the practical minimum viable deployment, covering the highest-weight positives for users and providers with no new burden for either side.

Phase 3 — controlled transform and session assurance. Deployed by a provider *together with* a transform authority; the only bilateral layer in the stack. Components are sanitizable signatures (Ateniese et al., 2005) with a third-party trapdoor authority plus designated-verifier signatures (Jakobsson et al., 1996) bound to the client session, keeping D9 intact. A transform authority issues scoped rewrite tokens for specific schema adaptations; any modification not covered by a valid token invalidates the signature. The natural coordination vehicle is a compliance mandate: a regulated vertical can make Phase 3 a procurement requirement, collapsing bilateral coordination to a single contracting event. Architecture details and the trapdoor custody argument appear in Appendix F.

Table 2. Progressive deployment ladder. † Phase 1 requires a minimum of two ring members; the minimum viable deployment is a bilateral anchor between two mutually contracting parties. ‡ Phase 2 requires provider-side auth-flow engineering (OAuth 2.0 DPoP) but no simultaneous client or router code changes.

Ph.	Initiator	Day-1 benefit to	Unilateral?
0	Client	U-ind., U-ent./vert.	yes
1	Router pair	R-compl., U-ent.	**no**† (bilateral min.)
2	Provider	P, U-ent./vert.	yes‡
3	Prov. + TA	U-ind., R-aggr., P	**no**
4	Consortium	Ecosystem audit	no

Phase 4 — ecosystem-level transparency. Append-only transparency logs (Certificate Transparency (Laurie et al., 2013), Sigstore (Newman et al., 2022)) turn per-router, per-session cryptographic facts produced at Phases 0–3 into a publicly auditable substrate. Phase 4 requires collective action and sits last; Sigstore's rollout across `PyPI` and `npm` suggests the coordination problem is solvable once preceding layers produce facts worth logging.

Deployment ladder and mixed deployments. Table 2 summarizes who deploys what when; mixed-deployment compatibility under our wire-format assumptions is shown in Table 9 (Appendix F).

Returning to the motivating lab setting. In the motivating research-lab setting, the phases matter differently. At Phase 0, the middleware tokenizes credentials in outgoing prompts, so any passive scanner recovers ciphertext rather than AWS keys and project identifiers. At Phase 2, a leaked bearer credential no longer yields durable abuse capacity because it is short-lived and channel-bound, curtailing the 440-session blast radius documented in Liu et al. (2026). At Phases 1 and 3, the institution can demand router attestation and constrain legal rewrites as procurement conditions. Taken together, Phases 0 and 2 provide unilateral Day-1 value, whereas Phases 1 and 3 require a bilateral bootstrap but remain incrementally adoptable without waiting for ecosystem-wide deployment. All layers compose. Section 6 evaluates the implemented Phase 0 layer and benchmarks the cryptographic costs that anchor the remaining phases.

6. Implementation and Evaluation

We instantiate the Phase 0 layer of the stack from §5 as a `LiteLLM` middleware and evaluate only the resulting security, utility, and latency outcomes here; implementation details, workloads, attack definitions, and extended benchmarks are moved to Appendix B.

We report four metrics. **Secret Exposure Rate (SER)** is the fraction of ground-truth structured secrets recoverable by a malicious router's scanner (lower is better). **Tool-call Integrity Preservation (TIP)** is the fraction of tool-call arguments matching provider intent byte-for-byte (higher is better). **Latency Overhead (LO)** is the percent increase in end-to-end request latency at P50/P95 vs. naked `LiteLLM`. **Downstream Utility (DU)** is normalized LLM-judge response quality when prompts carry tokenized rather than plaintext secrets. Appendix B gives the exact protocol, workload construction, judge-calibration procedure, and confidence-interval methodology.

Table 3 summarizes the headline results. Against AC-2, both Phase 0 modes drive SER to 0.00 ± 0.00 on fully matched secret families, while the more policy-relevant format-variant row remains 0.02 ± 0.01, which captures the residual false-negative boundary under evolving credential formats. Regex redaction reduces exposure only partially and materially degrades utility.

Table 3. Phase 0 headline results vs. baselines. B1: naked `LiteLLM`; B2: regex redaction; B3: high-risk policy gate (Liu et al., 2026). SER $\in [0, 1]$ (lower = better); LO = P95 latency overhead; DU = LLM-judge quality (1.00 = baseline). All values explicit.

Metric	B1	B2	B3	P0-A	P0-B
SER (AC-2, all secrets)	1.00	0.12 ± 0.04	1.00	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$
SER (AC-2, format-variant)	1.00	0.63 ± 0.07	1.00	$\mathbf{0.02 \pm 0.01}$	$\mathbf{0.02 \pm 0.01}$
TIP (AC-1 generic)	0.00	n/a	1.00 (HR only)	0.00	0.00
TIP (AC-1.b conditional)	0.00	n/a	0.00	0.00	0.00
LO P95	0%	<1%	2.0%	$\mathbf{2.9\% \pm 0.5\%}$	$\mathbf{4.8\% \pm 0.6\%}$
DU (mock judge)	1.00	0.64 ± 0.08	≈0.99	$\mathbf{0.93 \pm 0.03}$	$\mathbf{0.91 \pm 0.04}$
DU (live API, n=300)	0.936	–	–	$\mathbf{0.947}$ [$\mathbf{0.937}$, $\mathbf{0.957}$]	$\mathbf{0.929}$ [$\mathbf{0.917}$, $\mathbf{0.941}$]

Against AC-1 and AC-1.b, Phase 0 provides no benefit by design: TIP matches the unprotected baseline because tool-call integrity is deferred to Phase 3 (§5). The relevant Phase 0 result is therefore that tokenization introduces no additional integrity regression while staying within the deployment budget on latency.

DU is preserved in both modes. In the live API evaluation, LIVE-PLAIN scores 0.936, LIVE-P0A scores 0.947 [0.937, 0.957], and LIVE-P0B scores 0.929 [0.917, 0.941]; both tokenized conditions satisfy the pre-specified $\Delta = 0.05$ non-inferiority margin. This keeps the main empirical claim narrow: Phase 0 is already deployable as an AC-2 mitigation, with sub-5% P95 latency overhead and no practically meaningful loss in downstream task quality. Extended adversarial-evasion results, sidecar and streaming measurements, detector coverage, and the phase-by-phase microbenchmarks for Phases 1–4 appear in Appendix B.

7. Related Work

Agent safety and tool use. Work on LLM agent safety targets risks *within* the agent's context window: prompt injection via third-party content (Abdelnabi et al., 2023), unsafe reasoning-acting loops (Yao et al., 2023), and tool-emulation sandboxes. These works assume honest delivery

of content and ask whether the model is fooled by what it receives. We ask a prior question: whether the content reaches the model unmodified at all. Under the framing of Abdelnabi et al. (2023), a multi-hop router is a channel where "trusted content" enters the trust boundary silently; our Phase 0 structurally prevents secrets from appearing in that channel as plaintext, independently of the model's behavior.

Prompt-level privacy and PII protection. GPTWall (Li et al., 2024) and Pr$\epsilon\epsilon$mpt (Chowdhury et al., 2025) protect prompt privacy on a *direct* client-provider channel with no intermediate hops. Because they target a direct client-provider channel, they do not need the request-scoped local vault bookkeeping that a hidden router chain introduces. Our Phase 0 instead pairs tokenization with macaroon caveats (Birgisson et al., 2014) and request-scoped detokenization, so plaintext is restored only inside the client's live local workflow rather than exposed on the router path. Neither GPTWall nor Pr$\epsilon\epsilon$mpt addresses multi-hop router chains, router attestation, or credential short-liveness.

Middleware security and trusted intermediaries. Decades of systems-security work on application-layer middleware grappled with structurally similar intermediary-trust problems. CORBA and SOAP-era message-security specifications (OASIS, 2006) addressed integrity and confidentiality through intermediary chains, but they required centralized policy enforcement points that the LLM router ecosystem—with its fragmented, business-sensitive topology (D5)—cannot support. gRPC interceptor chains and service-mesh security in Envoy and Istio (Istio Authors, 2024) handle per-hop mutual-TLS re-origination, but they assume a single organizational trust domain (the service mesh) rather than cross-organizational chains with adversarially heterogeneous operators. The key lesson from this three-decade literature is that intermediary trust is an *institutional* problem before it is a cryptographic one: the CA/Browser Forum—not PKIX algebra—is what made WebPKI deployable at scale. The same lesson motivates our phase ladder: Phase 0 provides cryptographic protection without institutional coordination, while Phases 1–4 progressively add the institutional infrastructure (governance bodies, compliance mandates, transparency logs) that makes the later layers deployable.

Software supply-chain integrity and PKI governance. Build-pipeline integrity tools (IN-TOTO (Torres-Arias et al., 2019), TUF (Samuel et al., 2010), SLSA (OpenSSF SLSA Working Group, 2023)) address static artifact provenance: the threat model is a compromised build step, verification is asynchronous, and there is no session-level attestation model. LLM routing, by contrast, is a live, stateful, per-request service; our Phase 4 layer extends Certificate Transparency (Laurie et al., 2013) to per-router, per-session facts rather than static certificate-issuance events. The CT deployment record—nearly a decade of CA/Browser Forum coordination before SCT inclusion became mandatory (Ryan, 2014)—is therefore a useful analogy for the Phase 1 governance trajectory we describe.

LLM supply-chain measurement. Liu et al. (2026) measure payload injection and secret exfiltration at ecosystem scale, but the mitigations they discuss remain detection-based even though their own AC-1.b analysis shows how easily such defenses can be evaded. We take those measurements as motivation and argue that structural prevention is the better frame. Relative to that line of work, our contribution is not a new detector but a deployment ladder that (i) is grounded in a cross-stakeholder red-line analysis, (ii) composes individually mature cryptographic mechanisms against those red lines, and (iii) orders phases so each can be initiated by a single stakeholder with a Day-1 benefit independent of whether other parties have upgraded.

8. Discussion and Conclusion

We have shown that the LLM router trust gap is not an engineering oversight closable by a better policy gate, but a structural consequence of three stakeholders whose incentive constraints have made every strong-coordination proposal deploy-infeasible. Our deployable stack composes already-mature mechanisms against a red-line analysis; each layer is unilaterally adoptable by one party and backward-compatible with unupgraded partners. Interpreted through the lens of Cooperative AI, the routing layer is a coordination problem under asymmetric information: the stack produces a path toward constrained cooperation where each party participates for its own individually rational reasons, with the aggregate effect of raising the trustworthiness floor.

Societal relevance. If a Phase 0-style deployment were adopted across the 400 free routers audited by Liu et al. (2026), then under a stylized assumption of one leaked upstream credential per router per month it could reduce exposure across roughly 176,000 downstream sessions monthly. We present that figure only as an order-of-magnitude illustration of potential risk reduction, not as a forecast of real-world impact. Potential beneficiaries include research labs running autonomous experiment pipelines, healthcare organizations whose prompts may embed patient-linked identifiers or clinical access tokens, and public-sector agencies operating LLM systems in compliance-bound workflows. For such institutions, the main value may be the ability to reduce passive credential scanning without requiring trust in the router operator; depending on deployment context, that property may also support internal data-flow controls and subprocessor-accountability requirements associated

with regimes such as HIPAA §164.502, GDPR Article 28, and SOC 2 CC6.1. The harms potentially mitigated include API-key and cloud-credential leakage, unauthorized access to sensitive records, financial theft such as cryptocurrency-wallet compromise, and some of the downstream incident-response and breach-notification burden that such leaks can impose.

Limitations. We do not claim a direct contribution to cooperative-agent algorithms or equilibrium theory; our contribution is *infrastructure for Cooperative AI*: a deployment-ready substrate that gives aligned agents the integrity guarantees their alignment training presupposes, and gives multi-stakeholder deployments the accountability evidence that institutional trust requires. Our evaluation is centered on Phase 0 and a controlled middleware prototype, so the later stack layers remain justified by mechanism-level analysis and microbenchmarks rather than end-to-end deployment evidence. The security gains are correspondingly narrow: Phase 0 materially reduces exposure of structured secrets to passive router scanning, but it does not solve natural-language leakage, metadata leakage, or conditional tool-call manipulation. Finally, our deployability argument is incentive-based rather than predictive; real-world adoption will still depend on vendor risk tolerance, integration priorities, and regulatory interpretation across domains.

References

Abdelnabi, S., Greshake, K., Mishra, S., Endres, C., Holz, T., and Fritz, M. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *AISec@CCS 2023*, pp. 79–90, 2023. doi: 10.1145/3605764.3623985. URL https://dblp.org/rec/conf/ccs/AbdelnabiGMEHF23.

Ateniese, G., Chou, D. H., de Medeiros, B., and Tsudik, G. Sanitizable signatures. In *ESORICS 2005*, pp. 159–177, 2005. doi: 10.1007/11555827_10. URL https://dblp.org/rec/conf/esorics/AtenieseCMT05.

BerriAI. Litellm. https://github.com/BerriAI/litellm, 2026. GitHub repository. Accessed 2026-05-08.

Birgisson, A., Politz, J. G., Erlingsson, Ú., Taly, A., Vrable, M., and Lentczner, M. Macaroons: Cookies with contextual caveats for decentralized authorization in the cloud. In *NDSS*, 2014.

Boneh, D., Boyen, X., and Shacham, H. Short group signatures. In *CRYPTO 2004*, pp. 41–55, 2004. URL https://dblp.org/rec/conf/crypto/BonehBS04.

Chowdhury, A. R., Glukhov, D., Anshumaan, D., Chalasani, P., Papernot, N., Jha, S., and Bellare, M. Pr$\epsilon\epsilon$mpt: Sanitizing sensitive prompts for LLMs. *CoRR*, abs/2504.05147, 2025. URL https://dblp.org/rec/journals/corr/abs-2504-05147.

Durak, F. B. and Vaudenay, S. Breaking the FF3 format-preserving encryption standard over small domains. In *CRYPTO*, pp. 679–707, 2017.

Dworkin, M. Recommendation for block cipher modes of operation: Methods for format-preserving encryption. Technical Report SP 800-38G, National Institute of Standards and Technology, 2016.

Fett, D., Campbell, B., Bradley, J., Lodderstedt, T., Jones, M., and Waite, D. OAuth 2.0 demonstrating proof of possession (DPoP). RFC 9449, IETF, 2023.

Freedman, M. J., Nissim, K., and Pinkas, B. Efficient private matching and set intersection. In Cachin, C. and Camenisch, J. (eds.), *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pp. 1–19. Springer, 2004. doi: 10.1007/978-3-540-24676-3_1. URL https://doi.org/10.1007/978-3-540-24676-3_1.

Istio Authors. Istio security: Architecture and concepts. Istio Documentation, 2024. URL https://istio.io/latest/docs/concepts/security/.

Jakobsson, M., Sako, K., and Impagliazzo, R. Designated verifier proofs and their applications. In *EUROCRYPT 1996*, pp. 143–154, 1996. URL https://dblp.org/rec/conf/eurocrypt/JakobssonSI96.

Jones, M., Nadalin, A., Campbell, B., Bradley, J., and Mortimore, C. OAuth 2.0 token exchange. RFC 8693, IETF, 2020.

Kolesnikov, V., Kumaresan, R., Rosulek, M., and Trieu, N. Efficient batched oblivious PRF with applications to private set intersection. In *CCS 2016*, pp. 818–829, 2016. doi: 10.1145/2976749.2978381. URL https://dblp.org/rec/conf/ccs/KolesnikovKRT16.

Krawczyk, H. and Rabin, T. Chameleon signatures. In *NDSS 2000*, 2000. URL https://dblp.org/rec/conf/ndss/KrawczykR00.

Laurie, B., Langley, A., and Kasper, E. Certificate transparency. RFC 6962, IETF, 2013.

Li, Q., Wen, J., and Jin, H. Governing open vocabulary data leaks using an edge LLM through programming by

example. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 8(4):179:1–179:31, 2024. URL https://dblp.org/rec/journals/imwut/LiWJ24.

LiteLLM Team. Security update: Suspected supply chain incident. https://docs.litellm.ai/blog/security-update-march-2026, 2026. Accessed 2026-04-11.

Liu, H., Shou, C., Wen, H., Chen, Y., Fang, R. J., and Feng, Y. Your agent is mine: Measuring malicious intermediary attacks on the LLM supply chain, 2026.

Newman, Z., Meyers, J. S., and Torres-Arias, S. Sigstore: Software signing for everybody. In *CCS 2022*, pp. 2353–2367, 2022. doi: 10.1145/3548606.3560596. URL https://dblp.org/rec/conf/ccs/NewmanMT22.

OASIS. Web services security: SOAP message security 1.1 (WS-Security). Technical report, OASIS Standard, 2006. URL https://docs.oasis-open.org/wss/v1.1/.

OpenSSF SLSA Working Group. SLSA: Supply-chain levels for software artifacts, v1.0. OpenSSF Specification, 2023. URL https://slsa.dev/spec/v1.0.

QuantumNous. New api. https://github.com/QuantumNous/new-api, 2026. GitHub repository. Accessed 2026-05-08.

Rivest, R. L., Shamir, A., and Tauman, Y. How to leak a secret. In *ASIACRYPT 2001*, pp. 552–565, 2001. URL https://dblp.org/rec/conf/asiacrypt/RivestST01.

Ryan, M. D. Enhanced certificate transparency and end-to-end encrypted mail. In *Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2014.

Samuel, J., Mathewson, N., Cappos, J., and Dingledine, R. Survivable key compromise in software update systems. In *CCS 2010*, pp. 61–72, 2010. doi: 10.1145/1866307.1866315. URL https://dblp.org/rec/conf/ccs/SamuelMCD10.

songquanpeng. One api. https://github.com/songquanpeng/one-api, 2026. GitHub repository. Accessed 2026-05-08.

Torres-Arias, S., Afzali, H., Kuppusamy, T. K., Curtmola, R., and Cappos, J. in-toto: Providing farm-to-table guarantees for bits and bytes. In *USENIX Security Symposium 2019*, pp. 1393–1410, 2019. URL https://dblp.org/rec/conf/uss/Torres-AriasAKC19.

U.S. Department of Health and Human Services. Modifications to the HIPAA privacy, security, enforcement, and breach notification rules under the HITECH act and the GINA. 78 Fed. Reg. 5566 (Jan. 25, 2013); codified at 45 C.F.R. Parts 160 and 164, 2013. "Omnibus Rule"; §164.502 governs permissible uses and disclosures of protected health information.

Wang, S., Zhao, Y., Liu, Z., Zou, Q., and Wang, H. SoK: Understanding vulnerabilities in the large language model supply chain. *CoRR*, abs/2502.12497, 2025. URL https://dblp.org/rec/journals/corr/abs-2502-12497.

Wei-Shaw. Sub2api. https://github.com/Wei-Shaw/sub2api, 2026. GitHub repository. Accessed 2026-05-08.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. ReAct: Synergizing reasoning and acting in language models. In *ICLR 2023*, 2023. URL https://dblp.org/rec/conf/iclr/YaoZYDSN023.

Zhang, Y., Jiang, Y., Chen, Z., Backes, M., Shen, X., and Zhang, Y. Real money, fake models: Deceptive model claims in shadow APIs. *CoRR*, abs/2603.01919, 2026. URL https://dblp.org/rec/journals/corr/abs-2603-01919.

Zheng, L., Chiang, W., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-Judge with MT-Bench and chatbot arena. In *NeurIPS 2023*, 2023. URL https://dblp.org/rec/conf/nips/ZhengC00WZL0LXZ23.

Table 4. Main evaluation results. B1: naked `LiteLLM`; B2: client-side regex redaction; B3: high-risk policy gate (Liu et al., 2026); P0-A: opaque-token mode (**SER = 0.00 ± 0.00, LO = 1.8% ± 0.3% P50 / 2.9% ± 0.5% P95, DU = 0.93 ± 0.03**); P0-B: FF1 FPE mode (**SER = 0.00 ± 0.00, LO = 3.2% ± 0.4% P50 / 4.8% ± 0.6% P95, DU = 0.91 ± 0.04**). Headline values are repeated here so the table is self-contained independently of macro expansion. "n/a" means the defense is not designed to address the corresponding attack dimension. TIP rows for P0-A and P0-B equal the unprotected B1 baseline by design (tool-call integrity is a Phase 3 concern); B1 TIP = 0.00 because the evaluation places the malicious mock router in the active attack path, so all tool-call arguments are rewritten. SER and TIP $\in [0, 1]$; LO is percent overhead vs. B1; DU normalized to B1 = 1.00. 50 independent runs per cell, 95% bootstrap CI.

Metric	B1	B2	B3	P0-A	P0-B
SER, AC-2, all secrets	1.00	0.12 ± 0.04	1.00	0.00 ± 0.00	0.00 ± 0.00
SER, AC-2, format-variant secrets	1.00	0.63 ± 0.07	1.00	0.02 ± 0.01	0.00 ± 0.00
TIP, AC-1 (generic rewrite)	0.00	n/a	1.00 (HR only)	0.00	0.00
TIP, AC-1.b (conditional trigger)	0.00	n/a	0.00	0.00	0.00
LO, P50 latency overhead	0%	<1%	1.2%	1.8% ± 0.3%	3.2% ± 0.4%
LO, P95 latency overhead	0%	<1%	2.0%	2.9% ± 0.5%	4.8% ± 0.6%
DU, LLM-judge average	1.00	0.64 ± 0.08	≈0.99	0.93 ± 0.03	0.91 ± 0.04

A. Evaluation Results

B. Implementation Details and Extended Evaluation

This appendix collects the material moved out of §6 so that the main body reports only the empirical outcomes needed for the paper's central claim.

B.1. Prototype architecture

The middleware attaches four hooks to the `LiteLLM` proxy lifecycle. A **pre-call** hook detects high-value structured fields (regex-based, user-extensible), tokenizes each under a locally-held key with a per-request tweak and records the `(token, original)` pair in a request-scoped vault. A **post-call** hook scans inbound tool-call arguments and output for known tokens and, after validating a request-scoped macaroon (Birgisson et al., 2014) capability caveat, restores original values. An **async logging** hook writes an append-only transparency record for every event, and a **capability guard** rejects detokenization attempts whose capability caveat does not match the current request context. Two tokenization modes share this plumbing: **P0-A** (opaque tokens `[[TOK:APIKEY:07]]`) and **P0-B** (FF1 FPE under NIST SP 800-38G (Dworkin, 2016), explicitly avoiding FF3/FF3-1 (Durak & Vaudenay, 2017); the live evaluation uses a SHA-256 format-preserving hash approximation for portability—the SER = 0.00 result is independent of the underlying permutation).

B.2. Attack model and workloads

We model all four attack classes as a mock upstream in `LiteLLM`'s routing table. **AC-1** rewrites tool-call arguments in the returned JSON; **AC-1.a** uses typosquatting to evade domain-gate checks; **AC-1.b** applies a conditional predicate (per-session counter + `YOLO` flag) that is indistinguishable from benign behavior during finite-sample probing. **AC-2** passively scans the request body against the Mine regex pattern set (Liu et al., 2026); we additionally simulate a stronger LLM-aided extractor variant. Workloads combine a Mine-fixture corpus reconstructed from Liu et al. (2026), a ShareGPT subset with context-aware secret injection, and small HumanEval and synthetic samples covering the secret-type distribution of Mine Table 1. Baselines are B1 (naked `LiteLLM`), B2 (client-side regex redaction), and B3 (the high-risk policy gate of Liu et al. (2026)).

B.3. Evaluation protocol

We report four metrics. **Secret Exposure Rate (SER)** is the fraction of ground-truth structured secrets recoverable by a malicious router's regex scanner (lower is better, 0.00 means none recovered). **Tool-call Integrity Preservation (TIP)** is the fraction of tool-call arguments matching provider intent byte-for-byte (higher is better). **Latency Overhead (LO)** is the percent increase in end-to-end request latency at P50/P95 vs. naked `LiteLLM` (values over 5% at P95 we treat as a deployment blocker). **Downstream Utility (DU)** is the LLM-as-judge normalized response quality when prompts carry tokenized rather than plaintext secrets (reference 1.00 on unmodified prompts). DU uses GPT-4o as judge with a five-criterion rubric scoring task-relevant content preservation, reasoning coherence, instruction-following completeness, absence of tokenization artifacts, and factual fidelity relative to the plaintext-prompt baseline (each 0–2, normalized to

$[0, 1]$). To calibrate judge reliability, we draw a 100-sample held-out set and have a second independent LLM judge (Claude-3.5-Sonnet) score it; Cohen's $\kappa = 0.81$, consistent with the inter-judge agreement reported in Zheng et al. (2023). Because the mock-provider DU measurement establishes only a *necessary* condition on utility preservation, we also ran a 300-sample live-API evaluation. Diverse synthetic prompts across five task categories (code generation, data analysis, agentic pipelines, config/debug, and Q&A) with injected credentials (AWS, OpenAI, GitHub PAT) were submitted to `gpt-4o-mini` (temperature 0) under plaintext (LIVE-PLAIN), P0-A (LIVE-P0A), and P0-B (LIVE-P0B) conditions, scored by GPT-4.1 under the same five-criterion rubric. Results: LIVE-PLAIN 0.936, LIVE-P0A $0.947\,[0.937,\ 0.957]$, LIVE-P0B $0.929\,[0.917,\ 0.941]$ (95% bootstrap CI, 10 000 resamples; resampling unit: prompt–condition pairs, independent by construction). The judge receives the tokenized prompt (not plaintext) for P0-A/P0-B, mirroring end-user experience; this may contribute to the P0-A $>$ plain direction. Both P0-A and P0-B pass TOST equivalence under the pre-specified $\Delta = 0.05$ DU non-inferiority margin ($t_{\min}(299) = 8.73$, $p < 0.001$; per-test: P0-A t_{lo}=9.76, t_{hi}=15.26; P0-B t_{lo}=11.58, t_{hi}=8.73). Paired t-tests tell the same story: P0-A yields $t(299) = -2.75$, $p = 0.012$, $d = -0.16$ (Bonferroni-significant at $\alpha/3 = 0.017$; small effect, consistent with the judge-design note above), while P0-B yields $t(299) = 1.42$, $p = 0.31$, $d = 0.08$. Across all five task categories, both modes remain ≥ 0.84 (data-analysis 0.843–0.860 across conditions, reflecting task difficulty rather than tokenization artifact).

The prototype itself runs under 408 lines of Python core logic on top of `LiteLLM`'s hook API on a four-core x86_64 container (16 GB RAM) with a local mock provider at 100 QPS over 60 seconds. *Scope*: the local mock eliminates WAN jitter, so the reported latency figures should be read as lower bounds for production (FPE computation cost is stable, but round-trip jitter dominates at inter-continental latencies). The <5% P95 threshold is therefore conservative relative to any real-deployment floor.

B.4. Interpretation and extended results

Workload: 500 requests across 60 seconds (100 QPS); secret types drawn from the 12-family Mine corpus (Liu et al., 2026) (AWS, OpenAI, Anthropic, GitHub PAT, PEM blocks, EVM/Solana wallets, Slack, GCP, Stripe/Twilio) plus a ShareGPT agentic subset with injected credentials. Format-variant secrets are drawn from a held-out set of prefix-drifted and version-updated key formats not seen during detector development.

Against AC-2, Phase 0 tokenizes secrets before the prompt leaves the client, so any regex or LLM-based scanner receives ciphertext rather than plaintext. B1/B3 provide no defense (a payload-modification gate is not engaged by a passive scan); B2 provides partial defense but degrades on format variants and damages utility by removing prompt context the model needs. The zero SER for fully matched families is *deterministic*; the 95% CI reflects detection-pipeline variance, not probabilistic protection. The statistically meaningful result is the format-variant SER, 0.02 ± 0.01, which measures the detector's false-negative rate on out-of-distribution formats and characterizes the actual protection boundary. Against AC-1 and AC-1.b, Phase 0 provides no benefit by design: TIP equals the B1 baseline (B1 TIP = 0.00 because the malicious mock router rewrites all tool-call arguments in this evaluation configuration). Tool-call integrity is a Phase 3 concern (§5); Phase 0 introduces no regression on TIP. Latency overhead remains below 5% at P95; downstream utility is preserved above 0.90, substantially above the regex-redaction baseline (Chowdhury et al., 2025; Li et al., 2024).

B.5. Protocol overhead benchmarks: Phases 1–4

Sections 4 and 5 also make performance claims about the unimplemented layers, so we benchmark representative operations for those layers on the same hardware. Phase 1 ring-verify (16-member RSA-2048) takes 1.76 ms in pure Python (consistent with ≈0.8 ms at 10–50× native-C speedup, ✓); Phase 2 DPoP proof generation takes 0.062 ms (✓) and yields a 108,745× abuse-window reduction (✓); Phase 3 sanitizable-signature verification takes 2.52 ms (✓); and Phase 4 Merkle-log insertion reaches 244,509 entries/s (✓). Taken together, these numbers keep all four layers within the D8 budget introduced in Section 4. *Scope*: Phase 0 does not address AC-1.b (conditional-delivery injection), which requires Phase 3 chameleon signatures. An adaptive AC-2 attack that prompts the model to echo tokenized fields recovers only token identifiers (e.g., `[[TOK:APIKEY:07]]`), not the underlying credential; detokenization still requires the client-held vault key. We also do not address natural-language leakage (Li et al., 2024; Chowdhury et al., 2025) or existence-hiding.

Detector coverage and scope. The Phase 0 detector covers 12 credential format families from Liu et al. (2026) (AWS, OpenAI, Anthropic, GitHub PAT, PEM blocks, EVM/Solana wallets, Slack, GCP, Stripe/Twilio): SER = 0.00 ± 0.00 for all 12; SER = 0.02 ± 0.01 for format-variants (prefix drift); SER = 1.00 for custom enterprise tokens and natural-language secrets (out of scope; user-extensible via the `PatternRegistry` single-line API). The primary security boundary is

the format-variant SER, 0.02 ± 0.01: prefix-shift events such as `sk-` → `sk-proj-` → `sk-ant-` have repeatedly placed real credentials outside the matched pattern set as provider APIs evolved, so time-to-patch matters as much as steady-state coverage. Semantic credentials (where the model must reason about the credential value) use a selective pass-through flag; pass-through restores semantic access at the cost of AC-2 protection for those fields. The false-positive rate on our benchmark corpus was 0.0%; tokenization is always reversible within the active request via the vault log.

Adversarial evasion evaluation. We constructed a 20-secret adversarial test set (5 instances each across the 4 primary families: AWS access key, AWS secret, OpenAI, GitHub PAT) and applied five adversarial transformations: base64 encoding (ADV-1), split across two JSON fields (ADV-2), unicode homoglyph substitution (ADV-3), novel-prefix variants (`sk-ant-`, `gho_`, `AKID`; ADV-4), and plaintext embedded in natural-language sentences (ADV-5). SER by category is ADV-1 = 1.00 (baseline) / 0.00 (with +base64-decode pre-pass; n=20, 10-line extension, see Appendix H); ADV-2 = 1.00; ADV-3 = 0.95; ADV-4 = 0.75; and ADV-5 = 0.00. The base64 pre-pass is an *implemented prototype extension* (10-line addition to the tokenizer, verified on the adversarial set): it decodes every candidate base64 token and re-scans for credential patterns, reducing ADV-1 SER from 1.00 to 0.00 at minimal latency cost (<0.1 ms per request). ADV-2 (field-split) requires schema-aware concatenation across adjacent JSON fields; a general-purpose middleware tokenizer operating on prompt strings has no a priori knowledge of which fields will be paired, and speculative exhaustive-pair scanning is exponential in field count—this is an architectural constraint, not a prototype gap, and closing it requires application-layer schema hints outside Phase 0's scope. ADV-3 and ADV-4 fall on the same format-variant boundary, whereas ADV-5 (prose-embedded plaintext) is fully covered at SER = 0.00.

Residual leakage. Phase 0 still leaks metadata. In P0-B (FF1 FPE) mode, ciphertext length equals plaintext length, so a router inferring secret length gains at most $\lceil \log_2 L_{\max} \rceil \approx 7$ bits per field; in P0-A (opaque token) mode, all tokens are fixed-length, eliminating length leakage. Format-class identity is still visible (a router learns that an AWS key is present, not its value), and timing variance is bounded by our measured $< 5\%$ P95 overhead. Traffic-analysis mitigations (padding, dummy requests) remain future work.

Supply-chain circularity. A supply-chain compromise of the client's `LiteLLM` installation could bypass Phase 0 tokenization, so the recommended production deployment is a *sidecar proxy* that resolves this circularity (App → Phase-0 sidecar → LiteLLM → Router → Provider), adding one local HTTP round-trip (measured two-hop P95: 0.265 ms, $< 0.5\%$ additional overhead at a 60 ms baseline; architecture and full benchmark in Appendix H). Both the hook-API and sidecar modes provide identical AC-2 protection; only the supply-chain threat model differs.

Streaming API extension. Production SSE streaming is handled by a streaming-aware post-call hook (Algorithm 1, Appendix H) that accumulates tool-call argument deltas and detokenizes at stream termination, leaving user-visible time-to-first-token unaffected; measured P95 0.053 ms on 30 live SSE calls (≈94× below the 5 ms bound, ✓). Appendix C summarizes attack-class coverage.

C. Attack Class to Phase Coverage

Table 5 maps each attack class to the phase that structurally addresses it and the mechanism responsible. Phase 0 is the only implemented layer; the remaining mappings follow from the design arguments of §5.

Table 5. Attack class coverage per stack phase. ✓= addressed structurally; ∼ = partially mitigated; × = out of scope for that phase.

Attack class	Ph. 0	Ph. 1	Ph. 2	Ph. 3
AC-2: passive secret scan	✓	×	×	×
AC-1: tool-call JSON rewrite	×	×	×	✓
AC-1.a: typosquatting rewrite	×	×	×	✓
AC-1.b: conditional delivery	×	×	×	✓
Credential resale / replay	∼	×	✓	×
Fake router identity	×	✓	×	×
Metadata / timing side-channels	×	×	×	×
Natural-language leakage	×	×	×	×

Metadata and natural-language leakage are out of scope for the entire stack as currently designed; they are addressed by orthogonal techniques (Li et al., 2024; Chowdhury et al., 2025).

D. Incentive Analysis: Payoff Structure

The per-phase individual-rationality claims in §1 and §5 are grounded in compliance-driven utility rather than universal rationality. We make the payoff structure explicit here.

Stakeholder interest matrix. Table 6 below consolidates the 13 stakeholder dimensions referenced in §3.

Table 6. Stakeholder interest dimensions. · marks indifference; *crit* marks a red line whose violation predicts non-adoption. U-ind.: individual developer; U-ent./vert.: enterprise/vertical builder; R-aggr.: commercial router aggregator; R-ent.: enterprise internal gateway; P: frontier model provider.

Dimension	U-ind.	U-ent./vert.	R-aggr.	R-ent.	P
D1. Structured-secret confidentiality in prompts/responses	high	*crit*	·	med	med
D2. Tool-call integrity: client executes what provider intended	*crit*	high	·	high	med
D3. Session-level assurance: response bound to originating session	med	high	·	med	med
D4. Auditable evidence trail for compliance and incident review	med	*crit*	med	*crit*	low
D5. Upstream-topology privacy: provider graph not disclosed	·	·	*crit*	low	high
D6. Cache/aggregation preservation: no per-request state forced	·	·	*crit*	med	·
D7. Zero-code integration: no invasive framework changes	high	high	*crit*	*crit*	*crit*
D8. Latency overhead $\leq 5\%$ of baseline	med	med	*crit*	med	high
D9. No new legal liability from signing execution-relevant content	·	·	high	high	*crit*
D10. Credential-abuse reduction: leaked key loses durable value	med	high	·	med	*crit*
D11. Regulatory evidence for SOC 2, HIPAA, PCI-DSS	med	*crit*	med	*crit*	low
D12. Pricing-flexibility preservation: per-token margins intact	·	·	high	low	*crit*
D13. Post-quantum agility: mechanisms replaceable independently	low	med	low	low	med

Phase 0 (client) — normal-form analysis. Let c_0 denote the integration cost (one-time middleware deployment, ≈ 408 LOC) and ρ_0 the expected credential-exposure loss per period. Table 7 gives the normal-form payoff matrix for Phase 0 adoption, where the client faces a router that is either honest or malicious.

Table 7. Phase 0 adoption game (client). Payoffs are per-period expected utility: ρ_0 is exposure loss; c_0 is integration cost; $\epsilon \approx 0$ is the overhead penalty when the router is honest.

Client action	Router honest	Router malicious
Adopt Phase 0	$-c_0 - \epsilon$	$-c_0$ (exposure blocked)
Do not adopt	0	$-\rho_0$ (exposed)

Adopting Phase 0 weakly dominates non-adoption: when the router is honest, the client pays only $c_0 + \epsilon$ (small, one-time + negligible per-request); when the router is malicious, adoption avoids $\rho_0 \gg c_0$. Since Phase 0 requires no partner changes and our evaluation shows c_0 is dominated by the Day-1reduction in exposure risk ($\rho_0 \gg c_0$ for any deployment with sensitive credentials), adoption is a dominant strategy for any client with $\rho_0 > 0$, *independent of router or provider behavior*. This is the payoff intuition behind the "individually rational" claim in §1: under these assumptions, Phase 0 adoption remains attractive in each column of Table 7.

Phase 1 (compliance-bound router). Let $p \in (0, 1]$ be the periodic audit probability, Π the expected non-compliance penalty (license revocation, contract termination, regulatory fine), and c_A the attestation integration cost.

Table 8. Phase 1 router payoff matrix.

Router action	Audited (p)	Not audited $(1-p)$
Adopt attestation	$-c_A$ + premium	$-c_A$
Do not adopt	$-\Pi$	0

Phase 1 adoption condition. For a compliance-bound router with audit probability $p > 0$, if $p \cdot \Pi > c_A$, then adopting Phase 1 ring-signature attestation is individually rational (strictly preferred over non-adoption in expected payoff). This property is unilateral: it holds regardless of whether clients have adopted Phase 0 or providers have adopted Phase 2.

For the enterprise internal-gateway class (R-ent.) audited under SOC 2 CC6.1 or FedRAMP AC-17, Π includes contract termination and compliance citation costs, so the condition $p \cdot \Pi > c_A$ is plausibly satisfied for any p above a few percent;

the condition is not universally satisfied and is not claimed to be. For commercial aggregators (R-aggr.) whose D5 (topology privacy) would be violated by auditable evidence, adoption is not individually rational; this is the basis for limiting the scope claim to compliance-bound and enterprise-facing routers.

Sensitivity analysis. To calibrate the condition $p \cdot \Pi > c_A$ against realistic parameters:

• *HIPAA context*: The IBM 2024 Cost of a Data Breach report gives an average *total* breach cost of $\approx \$4.45$M across all industries and breach causes; we use this as an upper-bound proxy for Π, acknowledging that the fraction specifically attributable to a routing-layer compliance failure is unknown and likely lower. Even discounting Π by an order of magnitude to $\$445$K to reflect uncertainty in attribution, and given annual audit probability $p \approx 1$ for covered entities, the condition $p \cdot \Pi > c_A$ still holds with ring-signature integration cost $c_A \approx \$20$K–$\$50$K. The sensitivity analysis therefore remains directionally valid even under conservative penalty assumptions.
• *FedRAMP context*: Π includes contract termination and re-authorization costs ($\approx \$500$K–$\$1$M); audits occur at least annually ($p \approx 1$); c_A is similar. Condition holds.
• *Unenforced context*: if $p \approx 0$ (no audits, no contractual enforcement), the condition fails and the analysis predicts non-adoption, consistent with the observed behavior of gray-market resellers.

These are order-of-magnitude estimates, not empirically validated cost figures; they illustrate that the individual-rationality condition is non-vacuous for the compliance-bound subclass we target.

Phase 2 (provider). Providers already operate OAuth 2.0 Token Exchange (Jones et al., 2020) and DPoP (Fett et al., 2023) in adjacent systems; the marginal integration cost is low. The benefit is reduced credential-resale abuse (D10), which the provider cares about at *crit* level. Adoption is individually rational for any provider whose credential-resale cost exceeds the marginal DPoP integration effort.

Free-rider dynamics. A natural concern is whether Phase 0 client adoption reduces the router's incentive to adopt Phase 1 — a free-rider dynamic. This concern does not apply as stated, because Phases 0 and 1 provide benefits to *different parties*: Phase 0 is a client benefit (D1, structured-secret confidentiality) in which the router has no stake, while Phase 1 is a router benefit (D4, auditable compliance evidence for SOC 2 and FedRAMP mandates) that the client cannot substitute. A client running Phase 0 does not close the router's compliance gap; the router still needs cryptographic attestation evidence to satisfy its own regulatory mandates. The genuine free-rider structure is *between routers*: if sufficiently many compliance-bound routers adopt Phase 1 attestation that enterprise clients begin requiring it as a procurement condition, non-adopting routers face exclusion from those contracts. This is a classic adoption-cascade dynamic and is favorable for the protocol: early adopters capture premium compliance-bound contracts, creating selection pressure on laggards. We also note that Phase 0 and Phase 2 interact constructively rather than substitutively: Phase 0 reduces the blast radius of a leaked credential (the token, not the plaintext, is what a passive scanner obtains), while Phase 2 reduces the *durability* of any credential that is leaked; the two mechanisms address orthogonal dimensions of the same credential-abuse threat.

Strategic interaction between phases. The individual-rationality claims above treat each stakeholder's adoption decision as independent of whether other phases have been deployed. A more complete treatment would model the strategic interaction: for example, does provider adoption of Phase 2 change the router's calculus for Phase 1? In our qualitative analysis, Phase 2 adoption increases the value of Phase 1 for compliance-bound routers, because a router that can attest its membership and demonstrate it uses short-lived credentials satisfies more of its compliance mandate than either mechanism alone. We do not model this interaction formally; the individual-rationality analysis is a necessary condition for adoption, not a game-theoretic equilibrium characterization, and a formal treatment of the adoption game is future work.

Adverse selection and the scope of each phase. A natural and important concern is the following: malicious routers will not voluntarily adopt Phases 1–4, because these phases introduce accountability mechanisms that bad-faith operators actively want to avoid. Does this make Phases 1–4 security theater? The answer depends on precisely what each phase is designed to accomplish.

Phase 0 is the primary structural defense against *malicious* routers: it requires no router action whatsoever, and its protection — preventing the plaintext value of structured secrets from appearing in the channel — holds unconditionally regardless of the router's intentions or compliance status. A gray-market reseller that declines to adopt Phase 1 attestation still faces a client whose Phase 0 middleware never exposes the plaintext credentials the attacker would otherwise harvest.

Phases 1–4 are not designed to constrain malicious actors; they are designed to raise the floor of the *compliant majority*. A compliance-bound router that adopts Phase 1 attestation does so because its enterprise contract or regulatory mandate requires it — and the individual-rationality analysis predicts adoption for precisely this population. The adversarially interesting population (gray-market resellers, compromised nodes) is treated as outside the voluntary adoption path by design; constraining them is a market-structure and legal problem, not a cryptographic one.

This is an asymmetric but not defective design: the combination of Phase 0 (client-side, unconditional, effective against malicious routers) and Phases 1–4 (router/provider-side, conditional on compliance incentives, effective against the compliant majority) produces a genuine improvement. Clients are protected unconditionally by Phase 0. The compliant majority of the router ecosystem is progressively drawn toward verifiable accountability by the compliance-incentive structure of Phases 1 and 2. The residual risk is a population of non-adopting bad-faith routers whom clients can increasingly filter out as Phase 1 attestation becomes a procurement requirement. We do not claim the stack eliminates the bad-faith population; we claim it renders them distinguishable and economically marginal relative to the compliant market.

Scope. These are existence proofs that individually rational adoption paths exist under stated conditions, not empirically validated equilibria; formal mechanism-design treatment and direct stakeholder elicitation are future work.

E. Extended Mechanism Drawer

Table 1 in the main text covers the mechanisms that survive the red-line test of §4. For completeness, the full design-space survey we performed additionally considered the following mechanisms, all of which we eliminated early for the reasons noted.

- *Per-response general-purpose zero-knowledge proofs (Groth16, Plonk, Halo2)*: eliminated against D7 (prover cost not tolerable at per-request budget) and D8 (latency).
- *Fully homomorphic evaluation of LLM inference*: eliminated against D7/D8 (orders-of-magnitude slowdown).
- *Router-side secure multi-party computation*: eliminated against D5/D7/D8 (topology privacy, coordination overhead, latency).
- *Oblivious transfer and private information retrieval*: considered for provider-side response delivery; eliminated as orthogonal to the core threat.
- *Watermarking of generated content*: considered for accountability; eliminated because the threat is payload modification and secret exfiltration, not content plagiarism, and because watermark-based accountability for tool-call execution semantics has no obvious security game.
- *Verifiable delay functions and randomness beacons*: considered for session-level freshness; found redundant with TLS 1.3 freshness.
- *Cryptographic accumulators*: retained as an optional component of the Phase 1 upstream-set commitment layer; not promoted to the main table.
- *Redactable signatures, aggregate signatures, proxy signatures, blind signatures, threshold signatures*: considered in the signature family; each either duplicates a surviving mechanism or violates a red line.
- *Intel SGX, Intel TDX, and AMD SEV-SNP remote attestation*: considered for provider-side and router-side attestation; treated as a substitutable implementation of the Phase 2 vault rather than a distinct layer.
- *Trusted timestamping services*: redundant with transparency logs in the Phase 4 layer.

None of the above is wrong or uninteresting in its own right; the claim is only that under the specific red-line constraints imposed by the current LLM router ecosystem (§3), none of them produces a Pareto improvement over the mechanisms promoted to the main stack. A different red-line structure (for example, a deployment context in which provider liability is not an issue, or where a trusted-hardware assumption is acceptable) would select a different subset of this drawer.

F. Phase 3 Architecture Sketch: Controlled-Transform and Session Assurance

Phase 3 is the only bilateral layer in the stack and the layer that structurally addresses tool-call injection (AC-1, AC-1.a, AC-1.b). Because it requires provider–transform-authority coordination, no prototype exists; this appendix provides a concrete architecture sketch to establish that the design is technically specified, not merely aspirational.

F.1. Parties and roles

- *Provider* P: generates responses, holds a signing key sk_P, issues per-response chameleon-hash commitments.
- *Transform Authority* TA: a neutral third party (schema registry, regional compliance body, or federated consortium) holding the chameleon-hash trapdoor td; issues *rewrite tokens* RT authorizing specific schema adaptations.
- *Router* R_i: may rewrite only fields covered by a valid RT; all other fields are frozen by the chameleon commitment.
- *Client* C: verifies the designated-verifier signature (DVS) bound to its session; detects any rewrite not covered by RT.

F.2. Per-response wire format

The provider appends to each response resp a Phase-3 header:

$$\mathsf{H}_3 = \big(\mathsf{CH.Com}(\mathsf{resp}, r),\ \mathsf{DVS.Sign}(\mathsf{sk}_P, \mathsf{resp}, \mathsf{sess}),\ \mathsf{tls\_exp}\big)$$

where $\mathsf{CH.Com}(\mathsf{resp}, r)$ is a chameleon-hash commitment to the full response body under randomness r (Krawczyk & Rabin, 2000), and $\mathsf{DVS.Sign}(\mathsf{sk}_P, \mathsf{resp}, \mathsf{sess})$ is a designated-verifier signature binding the response to the client session sess (Jakobsson et al., 1996). DVS non-transferability preserves D9: the assertion is session-bound and cannot be extracted as publicly verifiable legal evidence.

F.3. Rewrite token issuance

When a router R_i needs to adapt a schema field (e.g., renaming a parameter for a downstream provider's API format), it requests a rewrite token from TA:

$$\mathsf{RT}_{i,j} = \mathsf{TA.Issue}(\mathsf{td},\ \mathsf{field}_j,\ \Delta_j,\ \mathsf{scope})$$

where Δ_j specifies the permitted transform (field rename, value normalization within a declared range) and scope binds the token to a session or time window. The rewrite token is a signed authorization; the transform authority's signature over $(\mathsf{field}_j, \Delta_j, \mathsf{scope})$ is verifiable by the client using TA's public key.

F.4. Authorized rewrite and client verification

The router applies the permitted transform by re-computing the chameleon hash:

$$\mathsf{CH.Com}(\mathsf{resp}', r') = \mathsf{CH.Com}(\mathsf{resp}, r) \quad \text{using td}$$

so the commitment is unchanged even though $\mathsf{resp} \neq \mathsf{resp}'$; this is the trapdoor property of chameleon hashes. The router appends $(\mathsf{RT}_{i,j}, r', \Delta_j)$ to the header.

At the client, verification proceeds as:

1. Verify $\mathsf{DVS.Verify}(\mathsf{pk}_P, \mathsf{H}_3, \mathsf{sess})$: the response was issued for this session.
2. For each $\mathsf{RT}_{i,j}$ in the header: verify TA's signature and that Δ_j is consistent with the received resp'.
3. Verify $\mathsf{CH.Com}(\mathsf{resp}', r') = \mathsf{CH.Com}_0$: the commitment matches after applying declared transforms.
4. Reject if any field in resp' differs from $\mathsf{resp}_{\mathsf{frozen}}$ without a covering RT.

A rewrite not covered by a valid RT will fail step 3 because the adversary cannot re-randomize the chameleon hash without the trapdoor td.

F.5. Trapdoor custody and threat model

The critical design constraint is trapdoor custody. If P holds td: any router can obtain authorization directly from the provider, reducing Phase 3 to a provider-permission system with no TA value. If R_i holds td: the mechanism degenerates (routers can rewrite anything by re-randomizing). The correct configuration is TA holds td and issues narrowly scoped RTs per field family; TA becomes a schema registry analogous to a Certificate Authority for permitted transforms. The governance problem for TA admission is analogous to Phase 1 key-set governance and admits the same bilateral-anchor bootstrapping path.

F.6. Why Phase 3 is bilateral but not blocked

Phase 3 requires P to issue chameleon-hash headers and TA to issue rewrite tokens; neither can provide value alone. However, given Phase 2 already in place (provider has DPoP infrastructure), extending to chameleon-hash commitments on the response body is an incremental auth-flow change, not a new cryptographic deployment. Given Phase 1 governance already in place (a recognized key set and compliance contracts), TA admission follows the same institutional path. The bilateral dependency is therefore *sequenced* rather than simultaneous: Phase 3 is individually rational for each party once the preceding unilateral phases have been adopted, consistent with the deployment ladder argument of §5.

F.7. Mixed-deployment compatibility matrix

Table 9 shows that under our wire-format assumptions, mixed configurations (some clients on Phase 0, some providers on Phase 2, some routers on Phase 1) remain operational with no incompatible cell.

Table 9. Mixed-deployment compatibility. Entry shows the effective protection level; ✓ with phase numbers indicates the corresponding subset of Table 6 dimensions is lifted. Under the wire-format assumptions of the preceding phases, no cell is incompatible.

Client \ R/P	$(0,0)$	$(0,2)$	$(1,0)$	$(1,2)$
0	$\checkmark_{0}$	$\checkmark_{0,2}$	$\checkmark_{0,1}$	$\checkmark_{0,1,2}$
1	$\checkmark_{0}$	$\checkmark_{0,2}$	$\checkmark_{0,1}$	$\checkmark_{0,1,2}$
3	$\checkmark_{0,3^*}$	$\checkmark_{0,2,3^*}$	$\checkmark_{0,1,3^*}$	$\checkmark_{0,1,2,3^*}$

3^*: Phase 3 engages only when both the provider and a transform authority are active. A client running Phase 0 automatically benefits from Phase 2 once the provider deploys it.

F.8. Streaming-aware Phase-0 hook and sidecar deployment

Streaming algorithm. Algorithm 1 handles SSE streaming by accumulating tool-call argument deltas and detokenizing at stream termination; the text-content stream is forwarded chunk-by-chunk with no buffering.

Algorithm 1 Streaming-Aware Phase-0 Post-Call Hook

Require: SSE delta stream S, vault state V, request capability Cap
Ensure: detokenized response forwarded to client
$A \leftarrow \{\}$ {tool-call argument accumulator, keyed by call index}
for all delta d in S **do**
  if d.type $=$ `content_delta` **then**
    forward d to client immediately {no latency penalty}
  else if d.type $=$ `tool_call_delta` **then**
    $A[d.\texttt{index}] \mathrel{+}= d.\texttt{function.arguments}$
  else if d.type $=$ `done` **then**
    validate Cap against the active request context
    for all accumulated argument string a_j in A **do**
      for all token $t_i \in V$ matching any substring of a_j **do**
        $a_j \leftarrow a_j.\texttt{replace}(t_i,\ V.\texttt{lookup}(t_i))$
      end for
    end for
    emit $\texttt{detokenized\_tool\_calls}(A)$ to client
  end if
end for

Sidecar deployment. The recommended production deployment is a sidecar proxy that resolves the supply-chain circularity of the hook-API mode:

$$\texttt{App} \xrightarrow{\text{HTTP :8000}} \textbf{[Phase-0 sidecar]} \xrightarrow{\text{HTTP :8001}} \texttt{LiteLLM} \xrightarrow{\text{HTTPS}} \text{Router} \xrightarrow{\text{HTTPS}} \text{Provider}$$

The sidecar runs as a separate OS process with an independent Python virtualenv (vault, FPE, and macaroon libraries pinned and hash-verified at startup). Measured two-hop loopback P95: 0.265 ms (mean 0.151 ms), representing $< 0.5\%$ additional overhead at a typical 60 ms baseline request.

Vault key lifecycle. The Phase 0 vault holds root secret sk_V; losing this key renders all active session tokens permanently un-detokenizable. For individual developers, sk_V is stored in the OS-native credential store (macOS Keychain, Linux Secret Service, Windows Credential Manager). For enterprise deployments: (i) back up sk_V to an HSM or secrets manager (HashiCorp Vault, AWS Secrets Manager) independent of the `LiteLLM` process; (ii) key rotation either re-tokenizes all live tokens or uses CapRevoke to invalidate in-flight sessions cleanly; (iii) scope sk_V per deployment environment to limit blast radius of key compromise.

G. System and Threat Model

This appendix gives the system and threat-model details that sit behind §2–§2. The goal is to make the paper's assumptions explicit without requiring the reader to follow a fully formal treatment.

G.1. Parties and chains

A single agent interaction involves three classes of entities: a *client* C, a *router chain* $\mathcal{R} = (R_1, R_2, \ldots, R_k)$ of one or more application-layer proxies, and a *provider* P. Each hop R_i terminates the inbound TLS session and re-originates a new TLS session to R_{i+1} (or to P), meaning that WebPKI validation by the client extends only to R_1; the composition $(R_2, \ldots, R_k)$ is invisible to C by construction. A request req is a tuple $(\mathsf{meta}, \mathsf{sys}, p, \mathsf{tools})$ consisting of metadata, system context, prompt, and tool-call schema; a response resp is a sequence of $(\mathsf{content}, \mathsf{tc}_1, \ldots, \mathsf{tc}_m)$ where each $\mathsf{tc}_j = (\mathsf{name}_j, \mathsf{args}_j)$ is a tool call to be executed at the client.

G.2. Router capabilities and two attack classes

Given R_i, its capabilities after TLS termination are (i) read of all plaintext fields in req and resp, (ii) arbitrary rewrite of any field in resp before forwarding, (iii) side-channel forwarding of any observed content to an attacker-controlled sink, (iv) per-session state (e.g., counters, flags) that permits stateful behavior, and (v) re-origination of downstream TLS using any cipher suite it chooses.

Following Liu et al. (2026), we distinguish two core attack families. *AC-1* rewrites fields in the tool-call JSON; adaptive sub-variants *AC-1.a* (dependency-targeted typosquatting) and *AC-1.b* (conditional delivery gated by a stateful predicate) defeat respectively domain-based policy gates and finite-sample audits. *AC-2* passively scans request and response bodies for regex-matchable structured secrets and forwards matches out of band without modifying the payload.

G.3. Corruption and the weakest-link observation

We adopt a composable treatment of incentive-driven adversaries. Each R_i may be *Byzantine-active* (arbitrary rewrite plus side-channel), *Byzantine-passive* (side-channel only), or honest, with the corruption choice driven by incentive rather than by a fixed threshold. Let $\mathsf{Corr} \subseteq \{R_1, \ldots, R_k\}$ be the set of corrupted hops.

Weakest-link observation. For any nonempty Corr and any client-side detector D that observes only $(\mathsf{req}_C, \mathsf{resp}_C)$, there exists a conditional-delivery predicate φ such that the detector's flag probability on attacked traffic is at most negligibly greater than on honest traffic plus a probing-budget-dependent term, and honest downstream hops cannot detect the deviation because they possess no reference to the original upstream state.

This weakest-link observation is the appendix-level intuition behind the structural argument in §2; it restates the AC-1.b indistinguishability point from Liu et al. (2026) in a compact way. The practical consequence is that closing the loop cannot be done by passive observation alone: the defense must either prevent the adversary from *acquiring* the plaintext inputs it would exfiltrate (Phase 0) or prevent it from *producing* rewrites that pass downstream validation (Phase 3 with chameleon-signed frozen fields).

G.4. Out-of-scope

We explicitly do not address: (a) natural-language semantic leakage in the prompt, which is the subject of (Li et al., 2024; Chowdhury et al., 2025); (b) model-weight extraction; (c) provider-side malicious behavior, for which trusted-hardware or verifiable-inference approaches are complementary; (d) traffic metadata and timing side channels; and (e) prompt injection via the model's own reasoning, which is orthogonal.

H. Phase 0 Design Sketch

This section sketches the Phase 0 layer implemented by the prototype in §6. It is meant as a design-level description rather than a full formal specification; the supporting reasoning appears later in Appendix I.

H.1. Design outline

A *capability-indexed format-preserving tokenization* (CI-FPT) scheme Π is a tuple of algorithms:

$$\Pi = (\mathsf{Setup}, \mathsf{VaultInit}, \mathsf{CapIssue}, \mathsf{Tokenize}, \mathsf{Detokenize}, \mathsf{CapRevoke})$$

with the following informal semantics:

- $\mathsf{Setup}(1^\lambda)$ chooses public parameters pp, including a type-indexed family of FPE instances $\{\Pi^\tau_{\mathsf{FPE}}\}_\tau$ and an HMAC family H for tweak derivation.
- $\mathsf{VaultInit}(\mathsf{pp})$ generates a vault identifier vid and a root secret sk_V held only by the local vault process.
- $\mathsf{CapIssue}(\mathsf{sk}_V, \mathsf{cid}, \mathsf{scope}, \mathsf{ctx})$ returns a macaroon-style capability whose caveats include a request context ctx containing a fresh local request identifier rid, a time window $[t_0, t_1]$, and a set of allowed type labels.
- $\mathsf{Tokenize}(\mathsf{sk}_V, \mathsf{Cap}, m)$ validates Cap against the active local request context, detects high-value typed fields $\{(\tau_i, v_i)\}$ in the message m via a user-supplied detector, and for each field computes a tweak $\mathsf{tw}_i = H(\mathsf{cid} \| \tau_i \| i \| \mathsf{rid})$ followed by $c_i = \Pi^{\tau_i}_{\mathsf{FPE}}.\mathsf{Enc}_{\mathsf{sk}_V}(v_i, \mathsf{tw}_i)$; it returns the rewritten message m' together with a tokenization trace recorded in the vault.
- $\mathsf{Detokenize}(\mathsf{sk}_V, \mathsf{Cap}, \mathsf{resp}, T_{\mathsf{tok}})$ validates Cap against the same live request context, locates each c_i in resp using the trace, recomputes tw_i, and decrypts via $\Pi^{\tau_i}_{\mathsf{FPE}}.\mathsf{Dec}$.
- $\mathsf{CapRevoke}(\mathsf{sk}_V, \mathsf{cid})$ marks cid as revoked; subsequent detokenization attempts fail.

The key property of the construction is that the tweak tw_i *binds* the ciphertext to a locally issued request context via rid; replaying a token in a different request produces a different tweak, and replaying it after expiry fails because the vault no longer authorizes detokenization outside the issuing request window.

H.2. Operational correctness intuition

Under the assumption that the idempotent response path does not alter the ciphertext strings c_i (which follows from the format-preservation property of FPE and a canonicalization wrapper at the detection layer), $\mathsf{Detokenize}$ recovers v_i from an honestly issued Cap bound to the same live request context.

H.3. Desired security properties

We state three game-based goals in explicit "goal / capability" form:

- **Value indistinguishability under tokenization/detokenization oracle access**: *goal*—distinguish which of two same-type secret values was tokenized in a challenge prompt; *capability*—chosen-message access to $\mathsf{Tokenize}$ and $\mathsf{Detokenize}$ oracles on arbitrary non-challenge inputs. Concretely, after choosing (v_0, v_1, τ) of the same type, the adversary receives a challenge tokenization of one of them and must guess which value was used.
- **Capability unforgeability under issuance/tokenization/detokenization oracle access**: *goal*—forge a fresh valid capability; *capability*—oracle access to $\mathsf{CapIssue}$, $\mathsf{Tokenize}$, and $\mathsf{Detokenize}$ on adversarially chosen inputs. Success means producing a capability whose cid was never issued and that still verifies.
- **Cross-request replay narrowing under chosen-context access**: *goal*—make a token or capability captured in one local request context succeed in another or after revocation; *capability*—chosen-context access to tokenization and detokenization oracles together with captured cross-context artifacts. Success means causing $\mathsf{Detokenize}$ under $\mathsf{rid}_2 \neq \mathsf{rid}_1$, or after revocation of rid_1, to return the original plaintext of a token produced under rid_1.

H.4. Informal security takeaways

Value indistinguishability intuition. Under the pseudorandom-permutation security of the underlying FPE family and the pseudorandom-function security of the HMAC tweak derivation, the value-indistinguishability advantage of any computationally bounded adversary is bounded by the sum of these two advantages plus a collision term negligible in the HMAC output length.

Capability-unforgeability intuition. Under the existential unforgeability of the HMAC family as a MAC, the capability-unforgeability advantage of any computationally bounded adversary is bounded by the MAC advantage.

Cross-request replay intuition. Under freshness of locally generated request identifiers, correct enforcement of capability caveats, the strong PRP security of FPE, and the PRF security of HMAC, the cross-request replay advantage of any computationally bounded adversary is bounded by the identifier-collision probability, the PRP advantage, and a negligible term.

Phase 0 passive-exposure takeaway. Against a Byzantine-active corrupted router chain mounting AC-2, the probability of distinguishing any two type-equal secret values embedded in tokenized prompts is bounded by a sum of the value-indistinguishability, capability-unforgeability, and cross-request replay advantages, each individually negligible under their respective assumptions.

I. Informal Reasoning Sketches

The following sketches are informal reasoning outlines meant to show why the design claims are plausible. They intentionally avoid a full game-based treatment; detailed reductions, bound algebra, and edge-case handling are left to future cryptographic work. Where the argument is thin or depends on extra assumptions, we say so explicitly.

I.1. Value indistinguishability intuition

We chain four games. G_0 is the standard value-indistinguishability game: the challenger generates $(\mathsf{sk}_V, \mathsf{vid}) \leftarrow \mathsf{VaultInit}(\mathsf{pp})$, issues a challenge capability Cap^* on a fresh cid^*, and the adversary $\mathcal{A}$ submits (v_0, v_1, τ) with $|v_0| = |v_1|$; the challenger tokenizes v_b for a random bit b and returns the challenge ciphertext.

G_1 replaces the HMAC tweak-derivation $H(\cdot)$ with a lazily sampled truly random function R; the transition advantage is at most $\mathbf{Adv}^{\mathsf{PRF}}_{H}(\mathcal{A})$ by a standard PRF reduction that programs R using H.

G_2 replaces each invocation of $\Pi^{\tau}_{\mathsf{FPE}}$ with an independent truly random tweakable permutation over $\mathcal{D}_\tau$; the transition advantage is at most $\sum_\tau \mathbf{Adv}^{\mathsf{PRP}}_{\Pi^{\tau}_{\mathsf{FPE}}}(\mathcal{A})$ by a hybrid over type labels.

G_3 observes that the challenge tweak $\mathsf{tw}^* = R(\mathsf{cid}^* \| \tau \| \cdot)$ has never appeared as input to any prior oracle evaluation with probability $1 - q^2/2^\lambda$ (birthday bound over q oracle calls on an λ-bit output space); conditioned on this event, the challenge ciphertext is uniformly distributed in $\mathcal{D}_\tau$ independent of b, so $\mathcal{A}$'s advantage in G_3 is zero.

Summing the transition gaps yields that the value-indistinguishability advantage is at most $\mathbf{Adv}^{\mathsf{PRF}} + \sum_\tau \mathbf{Adv}^{\mathsf{PRP}^\tau} + q^2/2^\lambda$.

Caveat. the sketch does not handle the case where $\mathcal{A}$ queries $\mathsf{Tokenize}$ on a message containing both v_0 and v_1 simultaneously (joint embedding), or the case where $\mathcal{A}$ queries $\mathsf{Detokenize}$ on a response containing a ciphertext from a prior oracle call (token recurrence). A full proof must case-split on these events and bound their probability.

I.2. Capability-unforgeability intuition

We give a direct reduction $\mathcal{B}$ to MAC existential unforgeability. $\mathcal{B}$ receives a MAC challenger for H and simulates the capability-unforgeability environment for $\mathcal{A}$: for each $\mathsf{CapIssue}(\mathsf{sk}_V, \mathsf{cid}_i, \cdot)$ query from $\mathcal{A}$, $\mathcal{B}$ queries its MAC oracle on the first-level message $m_i = (\mathsf{cid}_i \| \mathsf{scope}_i \| \mathsf{ctx}_i)$ and assembles the resulting tag into a capability using the macaroon chaining structure. $\mathsf{Tokenize}$ and $\mathsf{Detokenize}$ are simulated using the FPE family under a separately sampled sk_V. When $\mathcal{A}$ outputs a forgery $\mathsf{Cap}^\dagger$ on a fresh $\mathsf{cid}^\dagger$, its first-level tag is a valid MAC on a message never queried to the MAC oracle; $\mathcal{B}$ outputs this as its own forgery. The MAC advantage transfer is exact modulo the simulation fidelity of the FPE oracle, which is bounded by the PRP advantage.

Caveat. the sketch omits (a) the macaroon discharge-chain verification logic in $\mathsf{Detokenize}$ and how it interacts with the MAC reduction, (b) the possibility that $\mathcal{A}$ forges a *caveat-extended* capability rather than a fresh cid, and (c) the CapRevoke race condition where a capability is revoked between issue and detokenization.

I.3. Cross-request replay intuition

We argue in three steps.

Step 1. By freshness of locally generated request identifiers, two distinct live request contexts share the same rid with probability at most $q^2/2^\lambda$ (birthday bound over q issued identifiers on a λ-bit space).

Step 2. Conditioned on $\mathsf{rid}_1 \neq \mathsf{rid}_2$, the HMAC-derived tweaks $\mathsf{tw}_i^{(1)}$ and $\mathsf{tw}_i^{(2)}$ for the same index i are computationally independent with probability $1 - \mathbf{Adv}_H^{\mathsf{PRF}}$, by the PRF security of H.

Step 3. Conditioned on independent tweaks, the FPE permutation under $\mathsf{tw}_i^{(1)}$ and its inverse under $\mathsf{tw}_i^{(2)}$ compose to a uniformly random permutation over $\mathcal{D}_\tau$; applying the request-context-2 inverse to the request-context-1 ciphertext recovers the original plaintext with probability at most $1/|\mathcal{D}_\tau|$ (PRP security under independent tweaks), which is negligible for types with $|\mathcal{D}_\tau| \geq 2^{40}$.

Caveat. small-domain types ($|\mathcal{D}_\tau| < 2^{40}$, e.g., short numeric identifiers) require the domain-extension construction noted in §I.5; the cross-request replay bound for those types is not immediate and requires a separate argument for the domain-extended FPE. The domain-extension section provides the construction but not the full security reduction.

I.4. Phase 0 passive-exposure intuition

Whatever advantage a Byzantine-active adversary $\mathcal{A}$ achieves at distinguishing embedded plaintext values in tokenized prompts decomposes, via a standard meta-reduction argument, into three cases: (i) $\mathcal{A}$ wins by breaking value indistinguishability of the FPE (bounded by the value-indistinguishability advantage); (ii) $\mathcal{A}$ wins by forging a capability to obtain a detokenization oracle on its chosen request context (bounded by the capability-unforgeability advantage); (iii) $\mathcal{A}$ wins by replaying a token from request context 1 in request context 2 (bounded by the cross-request replay advantage). Each summand is negligible under its respective assumption. The meta-reduction is a union bound and requires that the three cases are exhaustive, which holds by case analysis on the adversary's winning condition: any winning strategy must either (i) distinguish the ciphertext value, (ii) obtain a detokenization oracle, or (iii) cross request contexts.

I.5. FPE small-domain considerations

The PRP-to-PRF switching loss in the value-indistinguishability sketch is quadratic in the number of queries per tweak. For type domains of cryptographic size (for example, 256-bit API keys, 40-character wallet addresses, 64-character PEM blocks) this term is immediately negligible. For small-domain types (six-digit order identifiers, five-character postal codes) the raw FF1 bound is not tight, and a domain-extended construction (AES-CTR plus a Feistel wrapper that embeds the secret into a larger synthetic domain before FPE evaluation) is the standard remedy; we avoid FF3 and FF3-1 entirely in light of known small-domain attacks (Durak & Vaudenay, 2017).