# Trace-Reconstruction Attacks and Trace-Private Defenses for Confidential LLM Agents


## Abstract

An autonomous LLM *agent* chooses its own execution: how many retrieval hops to issue, which tool to invoke next, and when to terminate are runtime decisions conditioned on the private query and on the private documents the agent retrieves. The *shape* of the resulting execution trace (its step sequence, its per-step timing, and its egress sizes) is consequently a function of the private inputs. That shape remains observable to an honest-but-curious host even when the agent runs inside a trusted execution environment (TEE) composed with oblivious retrieval and output differential privacy, that is, with content, access, and output all protected.

We formalize this trace-shape channel as $(\varepsilon, \delta)$-trace-privacy, a differential-privacy notion whose sample space is the variable-length, adaptively generated agent trace rather than a single access or a single response. We instantiate a real multi-agent prior-authorization crew on LangGraph, replicating the agent roles of Microsoft's Prior-Authorization Multi-Agent Solution Accelerator, and mount a Trace-Reconstruction Attack against it: from trace metadata alone, with content, access, and output defenses in place, the passive host adversary infers whether a request concerns a sensitive, closely-scrutinized service at group-aware ROC-AUC 0.679 (and 0.96 on a second model); membership of an indexed patient case note is a weaker channel at 0.566. Our defense, TRACESHIELD, canonicalizes the agent's control flow to a data-independent execution plan and enforces public per-step deadlines and egress ceilings. The resulting release is $(0, 0)$-trace-private by construction, and a finite-$\varepsilon$ variant recovers graded adaptivity through randomized response over public plan templates. Deployed as an enforcing runtime over the live crew, TRACESHIELD drives the attack to chance (AUC 0.500) with no statistically significant utility degradation on our workload, at a measured latency cost of 15.6s to 36.0s per request, and emits a signed, integrity-verifiable Trace Receipt. A mechanism ablation shows that the secret is redundantly encoded across structure, timing, and size: closing any single coordinate leaves it almost fully recoverable, so the entire trace shape must be closed. Our adversary is the host of a *fully attested* TEE: attestation proves which code ran, but not how the agent moved, so a trusted-blind operator still fingerprints the workflow from its trace shape. We close the loop by binding each Trace Receipt to a hardware attestation, so the receipt proves the shaping ran inside the attested boundary: on an Azure Confidential VM, the SEV-SNP quote is verified by Microsoft Azure Attestation before it is bound. We release the crew, the attack, the defense, every recorded trace, the attestation-aware runtime, and a one-command confidential-VM deployment.


## 1 Introduction

Autonomous LLM agents are being deployed over confidential corpora such as clinical records, legal matter files, and proprietary code, under regulatory regimes that forbid exposing the underlying data to the infrastructure operator. The state of the art prescribes a layered defense. Run the agent and its retrieval inside a confidential VM (CVM), a trusted execution environment (TEE) on Intel TDX or AMD SEV-SNP attached to a confidential GPU, so that queries, documents, and model activations remain encrypted while in use [6, 19]. Conceal which record each access touches with oblivious retrieval [34, 39]. Bound what the final answer reveals about the corpus with output differential privacy [1, 8]. Content is sealed, access is oblivious, and output is private; every axis the literature knows how to protect is protected.

A side channel remains, and it is not an implementation bug but a direct consequence of agentic autonomy. The honest-but-curious host operating the infrastructure reads no payload, since the query, the documents, the activations, and the answer are all encrypted. It nevertheless observes the externally visible execution: the ordered sequence of boundary-crossing steps, the wall-clock timing of each, and the egress bytes each one emits. Because those observables are functions of the private inputs, they separate inputs. In our experiments, a prior-authorization request the agent resolves in 2.9 clinical-extraction passes with a clean coverage assessment is distinguishable, from trace metadata alone, from a request whose

content drives the agent to the depth cap and a longer, denser assessment; that observation by itself recovers a sensitive attribute of the request at ROC-AUC 0.679.

The adaptivity is intrinsic to the now-dominant reason-and-act loop [28, 31]: given a private request and the private referral and policy documents it encounters, the agent decides at runtime how many clinical-extraction passes to issue, which policy to retrieve, whether to loop back to repair an unsupported coverage criterion, and whether to revise the assessment after a medical-necessity review. A TEE encrypts the contents of guest memory and I/O, but not the pattern and timing of VM exits, tool invocations, and externally observable response sizes, so the trace shape crosses the confidential boundary in the clear. Recent microarchitectural and timing analyses of exactly these substrates confirm that such side information escapes the boundary [9, 13, 16, 36].

Each of the three standard defenses fails to close this channel for a specific reason. Per-access obliviousness protects one access; it says nothing about how many accesses occur or which tool follows which. Output DP bounds the answer, not the trace. Content encryption seals payloads, not structure. Stacking all three still leaves the joint, adaptive control-flow trace unprotected, and that gap sits exactly where agentic autonomy lives.

Figure 1 summarizes the end-to-end experiment, which we release in full.[1] A real multi-agent prior-authorization crew runs inside the confidential boundary and leaks a host-observable trace; an adversary attacks that trace, and our defense bounds the leak and certifies the bound. In summary, this paper makes the following contributions:

- *A formal notion.* We define $(\varepsilon, \delta)$-trace-privacy (Section 3), a differential-privacy notion whose object is the host-observable agent trace $T = \langle \text{step sequence}, \text{timing}, \text{sizes} \rangle$, a variable-length sequence whose length is itself data-dependent. The notion is defined separately for corpus-neighbors, which govern membership, and query-neighbors, which govern attribute inference. To our knowledge it is the first privacy notion aimed at the adaptive, multi-step control flow of an agent rather than at a single access or a single response.

- *A demonstrated attack.* We mount the Trace-Reconstruction Attack (Section 4) on the real crew: an adversary observing only trace metadata recovers whether the request concerns a sensitive, closely-scrutinized service at grouped, leakage-free AUC 0.679 and canary membership at 0.566, with content, access, and output defenses in place. Permutation, shape-fixed, and cross-topic controls attribute the leak to trace shape rather than to residual content or to template memorization.

- *A principled defense.* We design and deploy TRACESHIELD (Section 5), a planner that canonicalizes the agent's control flow to a data-independent shape and pads per-step timing and size to public ceilings. This release is $(0, 0)$-trace-private by construction (Proposition 1); run as an enforcing runtime over the live crew, it reduces the attack to chance with no statistically significant utility loss, at a latency cost we measure rather than assume away. A mechanism ablation shows why the whole shape must be closed: closing any single coordinate leaves the attribute leak nearly intact.

- *Verifiable evidence and an open artifact.* We introduce the Trace Receipt (Section 6): each request emits a signed, scope-self-describing receipt, anchored in a hash-chained ledger and checkable by a dependency-free verifier, that records the honest $(0, 0)$ guarantee for a canonicalized release and carries any residual as an empirical estimate. All 144 shielded receipts verify, tampering is detected, and the chain is intact.

- *Attestation-bound shaping on a real confidential VM.* We couple the defense to remote attestation (Section 7): the receipt binds the digest of a hardware attestation token and its launch measurement, so a verifier learns not only that the shaping was applied but that it ran inside an attested boundary. We deploy the whole system onto an Azure Confidential VM (AMD SEV-SNP) with one command; inside the guest the service reads the vTPM, submits SEV-SNP evidence to Microsoft Azure Attestation, and surfaces a verified quote that an auditor can re-verify live. The honesty invariants hold end-to-end: off attested hardware the same code reports a clearly labeled *simulated* verdict and the receipt keeps `hardware_attestation` false, which the verifier enforces.

We deliberately built the attack before the defense, because the attack substantiates the paper's central claim. Demonstrating that content protection, single-access obliviousness, and output DP are jointly insufficient against a metadata-only adversary is what motivates both the definition and the defense; a defense for a channel nobody has demonstrated would be a solution in search of a problem.

A note on scope before we proceed. Our empirical study of the channel adopts the confidential substrate's threat model, in which the adversary sees the trace's shape and timing but no payload, and measures the residual channel on real agent executions. The attack, the defense, the utility, and the latency are all measured on live runs of a real agent against a commercial LLM; for those measurements the hardware boundary is assumed rather than exercised, because the leak we study is

[1] The complete study of Figure 1, namely the crew, the attack, TRACESHIELD, the receipts, and every recorded trace, is released as an open-source artifact.

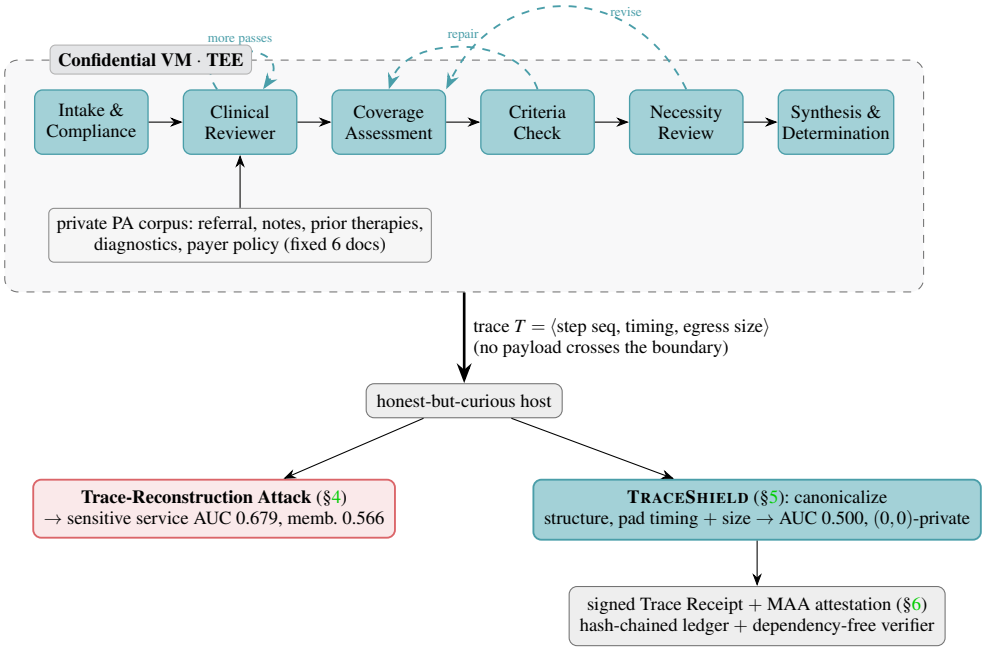


Figure 1: The released experiment. A real six-agent *prior-authorization* crew (PA-CREW, on LangGraph, replicating the agent roles of Microsoft's Prior-Authorization Multi-Agent Solution Accelerator) runs inside a confidential boundary over a private corpus of a member's referral documents and the payer's coverage policies. Its control flow is adaptive: how many clinical-extraction passes it takes, whether it loops to repair an unsupported coverage criterion, and whether it revises after a medical-necessity review are decided at runtime by the LLM over the private data (dashed edges). That adaptivity leaks through the host-observable trace $T$, whose payloads never cross the boundary. The Trace-Reconstruction Attack recovers whether the request concerns a sensitive, closely-scrutinized service at AUC 0.679 from $T$ alone; TRACESHIELD canonicalizes the control flow and pads timing and size to public ceilings, driving the attack to chance ($(0,0)$-trace-private, Proposition 1) and emitting a signed receipt bound to a hardware attestation. Every box maps to a released module: the crew to `agents.py` and `graph.py`, the recorder to `instrumentation.py`, TRACESHIELD to `shield_runtime.py`, attestation to `attestation.py`, and the attack, receipt, ledger, and verifier to the package.

a property of the trace, not of the silicon. Separately, and to close the deployment gap, we *do* provision a genuine confidential substrate: the artifact ships a one-command deployment onto an Azure Confidential VM (AMD SEV-SNP) with vTPM and Secure Boot, and the service inside it obtains a hardware attestation verified by Microsoft Azure Attestation (Section 7). We do not run a confidential GPU, and we do not re-run the full 288/144-run empirical study on that VM; we keep the two apart and label them so, and Section 10 states what each does and does not establish.

## 2 Background and Related Work

Retrieval-augmented generation [18] has moved from a fixed retrieve-then-generate step to agents that interleave reasoning, retrieval, and tool use [28], and that critique and revise their own work [31]. In medicine the pattern has advanced quickly, with multi-agent orchestration now proposed for clinical decision making and benchmarked on real workloads [14]. That autonomy is the source of the channel we study, because the control flow becomes a function of the private data the agent reads.

TEEs protect data in use by design, from hardware isolation primitives that resist memory-access side channels [6] onward. They are not free of side channels. A steady stream of attacks extracts secrets through ciphertext, interposer, fabric, and microarchitectural side channels [9, 15, 16, 19, 23, 29], and auditing the resulting state has itself become a research topic [3]. We take the strong side of that debate and assume the contents are protected, then study the structural channel that remains.

A separate line hides which record an access touches. Oblivious RAM and oblivious search conceal the access pattern [34, 39], and leakage-abuse attacks show why that matters [5]; more recent work warns that even TEE-backed encrypted databases leak through their access structure [12]. In parallel, differentially private prompt learning bounds what a released model or answer reveals [8], resting on the com-

position theorems of differential privacy [1, 10, 27], and the community has begun to ask how such deployments should be described and registered in the first place [25]. Each of these lines secures one axis of one step. None bounds the adaptive multi-step trace.

Traffic analysis and timing attacks have long recovered secrets from metadata [11, 33], and the idea that a program's iteration count or branch profile leaks its inputs is classical rather than new to LLMs. Very recent work carries it to LLM serving, where token-timing, length, and cache channels reconstruct responses [13, 36], and a fast-moving line reconstructs prompts or conversations from encrypted network traffic to a model or a retrieval agent [2, 17, 24]. Concurrent work [30] names timing, step count, and egress volume as attacker-visible "observable channels" in agent pipelines and evaluates padding and delay defenses. Our work differs on three axes. First, the *adversary vantage*: all of that work assumes a network observer (an ISP or a TLS wiretap) or a co-tenant, whereas our adversary is the host of the TEE itself, the very party that attestation is meant to make trustworthy, observing the internal step boundaries of a workload it cannot decrypt. Second, we give the channel a *formal* $(\varepsilon, \delta)$ *budget* over the variable-length trace rather than an empirical leakage number, and prove a $(0,0)$ construction. Third, we *bind the defense to attestation*: none of this work couples a shape defense to a hardware-rooted, verifiable receipt. Those attacks target a single model response; we target the agent's multi-step plan, whose length is itself data-dependent, as the object of leakage. On the agent side, attacks extract training data and RAG corpora [4, 26], exploit prompt injection [22], and steal the prompts that drive a service [7]. Closest in time, a wave of 2026 work measures extraction risk in LLM APIs [21], attributes poisoned knowledge in RAG [38], and attacks graph-based RAG through poisoning and query-efficient graph extraction [20, 37]. All of these concern the payload content in a channel. We concern ourselves with the structural and timing shape of the trace, we give it a formal DP budget, and we back it with a verifiable receipt. Verifiable inference through zkML proves which computation ran [35] but is impractically slow for per-request LLM use, so we rely on a signed receipt over an enforced budget instead.

Table 1 places these lines side by side.

Table 1: Where the channels sit. Each prior line protects one axis of one step; trace-privacy bounds the adaptive multi-step shape. A filled circle marks a channel a line addresses, an open circle one it addresses only partially or, for our row, assumes from the substrate rather than provides. This paper contributes the trace-shape and multi-step columns and assumes content, access, and output.

| | content | access (1-step) | output | trace shape | multi-step |
|---|---|---|---|---|---|
| Conf. TEE [6] | ● | | | | |
| ORAM/oblivious [39] | ● | ● | | ○ | |
| DP for LLMs [8] | | | ● | | |
| LLM side-chan. [13] | | | ○ | ○ | |
| Agent leak. [38] | ○ | | ○ | | ● |
| **This paper** | ○ | ○ | ○ | ● | ● |

## 3 Threat Model and $(\varepsilon, \delta)$-Trace-Privacy

Consider a health plan (or a provider's authorization service) that runs a prior-authorization agent $\mathcal{A}$ over a private corpus $D$ of a member's referral documents and the payer's coverage policies. On a request $q$, the agent executes an adaptive plan, that is, a sequence of steps (clinical-extraction passes, policy retrieval, criteria verification, medical-necessity review, and a determination) whose number, order, and parameters depend on $q$ and on the contents of $D$ the agent encounters. The agent and its retrieval run inside a confidential boundary, and we assume content confidentiality, single-access obliviousness, and output DP are all already supplied by the substrate. This defender-favorable assumption is deliberately strong: the leak we find survives even when all three of these protections are in place.

The adversary is an honest-but-curious host that operates the infrastructure outside the boundary. It cannot read payloads such as queries, documents, model activations, or answers, but it does observe the externally visible execution: the ordered sequence of boundary-crossing round-trips, both model calls and any external tool calls, their wall-clock timing, and the egress byte size of each. We are deliberately conservative about what counts as observable. Retrieval in the deployment we model runs inside the confidential VM over encrypted memory, so the number of documents an access returns crosses no boundary and is not host-visible. We therefore exclude result cardinality from the observable surface. The size we count is egress only: the bytes of the model round-trip that actually leave the confidential VM, never in-VM document bytes. Under an external oblivious store the per-access size and count would in any case be padded to a constant, so the residual data-dependent surface is the same three coordinates we model. We summarize those observables as the trace

$$T = \big\langle (s_1, \ldots, s_n),\ (\tau_1, \ldots, \tau_n),\ (b_1, \ldots, b_n) \big\rangle,$$

where $n$ is the data-dependent number of egress steps, $s_i$ is the $i$-th step's externally distinguishable type, $\tau_i$ its timing, and $b_i$ its egress size.

The adversary is passive: it counts and times round-trips but does not single-step the guest or probe microarchitectural state. Active adversaries do succeed against these substrates [9, 16, 23], and a single-stepping or page-fault adversary

would read intra-VM branches that an application-level defense cannot hide; we treat that stronger adversary as out of scope precisely so that any leakage we find is a floor, present even for the weakest realistic host. The adversary's goal is inference about the private inputs. Membership asks "is patient record $r$ in $D$?", a property of a single corpus record. Attribute inference asks "does the request concern a sensitive matter?", a case-type property that spans the query and the handful of records it draws on, and so is a group edit rather than a single-clause one. We keep the two as separate relations because they protect different secrets, and because the attribute is a group property we rely on Proposition 1, which covers changes of any size, rather than on a single-edit bound.

Formally, write $D \simeq D'$ for corpus-neighbors that differ in one patient record with the query held fixed, and $q \simeq q'$ for query-neighbors that differ in one clause of the request with the corpus held fixed. Let $\mathcal{M}$ be the randomized mapping from inputs to the observable trace induced by running $\mathcal{A}$ under a defense.

**Definition 1** ($(\varepsilon,\delta)$-trace-privacy)**.** *$\mathcal{M}$ is $(\varepsilon,\delta)$-trace-private with respect to a neighboring relation $\mathcal{R}$ if, for every pair $(x,x') \in \mathcal{R}$ and every measurable set $S$ of traces, $\Pr[\mathcal{M}(x) \in S] \le e^{\varepsilon}\Pr[\mathcal{M}(x') \in S] + \delta$. We report a corpus budget $(\varepsilon_D,\delta_D)$ under $D \simeq D'$, which governs membership, and a query budget $(\varepsilon_q,\delta_q)$ under $q \simeq q'$, which governs attribute inference; a change that spans several records and the query is a group edit, handled by group privacy, and at the $(0,0)$ release we certify this is exact for any group size, since a constant release is indistinguishable under any change.*

The sample space requires care, because the trace is not fixed-length. A trace of $n$ steps lives in $\Sigma^n \times \mathbb{R}^n_{\ge 0} \times \mathbb{N}^n$, and the full space is the disjoint union over $n \ge 0$ with its coproduct $\sigma$-algebra; $\mathcal{M}$ is a Markov kernel into it, with the data-dependent stopping time $n$ a measurable random index. This is the feature that separates trace-privacy from the notions it borrows from. Access-pattern privacy and oblivious RAM hide which cell a *fixed* program touches, and differentially private release and website-fingerprinting defenses pad the size or timing of a *single* response; in each case the number and order of operations are fixed by the program and are not themselves secret. An agent breaks that assumption, because the length $n$, the identity of the $i$-th step, and whether an $(i{+}1)$-th step happens at all are chosen at runtime from $D$ and $q$. The object to be bounded is therefore a distribution over variable-length, adaptively generated sequences, jointly across the timing and size coordinates the plan exposes at once. Prior single-axis, single-step notions do not reach that joint, variable-length, adaptive object. The definition states a worst-case bound; Section 9 reports, separately, the empirical leakage an actual attacker achieves, which is what a deployment monitors.

A subtlety governs how such a bound can be attained. One patient record can change an agent's entire plan, so the sensitivity of the raw trace to a corpus-neighbor is unbounded, and no amount of rounding or reordering applied to the raw trace is differentially private on its own: two neighbors that straddle a bucket boundary land in disjoint buckets, which yields an infinite likelihood ratio. The only construction that caps the loss without added noise is to stop reading the data, that is, to emit a trace that is a constant function of the inputs. That observation, developed in Section 5, makes the guarantee we certify a genuine one rather than an accounting exercise. It is also why we do not claim a small composed $\varepsilon$ for heuristic obfuscations; we measure their real effect instead. A per-tenant odometer, in the pay-as-you-go style [27], tracks how much non-data-independent release a session has spent and refuses further such release once a policy cap is reached.

## 4 The Trace-Reconstruction Attack

The attack instantiates the inference game of Section 3. The adversary collects observable traces labeled with a binary secret, either the membership of a canary patient record or the presence of a sensitive attribute, and it trains a classifier on trace features alone. The game itself is the classic membership setup [32], moved from model outputs to the trace.

From each trace we extract aggregate features: the step count, the count of distinct tools, and the sum, mean, and maximum of the per-step timing and egress-size vectors, together with per-tool counts. Attacker A1 is a logistic regression over those features. Attacker A2 adds tool-order bigrams and so captures which step tends to follow which. We report the attacker-favorable maximum of A1 and A2, and we compute ROC-AUC through the Mann–Whitney statistic with a 1,000-fold bootstrap 95% confidence interval. Because our evaluation suite (Section 9) is balanced and repeats each task cell, we report the headline number group-aware: we hold out an entire (clinical service, topic) task group at test time, so the classifier gains no advantage by memorizing a template. We also report a multi-split mean and standard deviation so that no single split carries the claim.

Two controls separate genuine signal from pipeline artifact, and both are necessary if the central claim is to hold. The permutation control shuffles the labels, so a correct pipeline must yield AUC near 0.5. The shape-fixed control replaces every trace's features with population constants, which erases the shape while keeping the labels intact, and it too must fall to 0.5; passing it shows that the recoverable signal lives in the trace shape rather than in any residual content leak.

We keep the attacker deliberately simple. A logistic regression over hand-built features is a weak learner, so the AUC it attains is a lower bound on what the channel leaks, and a stronger adversary can only do better; this makes the attack claim conservative rather than optimistic. The defended side needs no such caveat, because Proposition 1 shows that a canonicalized request emits a constant observable, against which the Bayes-optimal adversary has zero advantage re-

gardless of its features or its compute. Attacker strength thus matters only for measuring the undefended and partially defended leak, not for the guarantee we ultimately certify.

## 5 TRACESHIELD: a Leakage-Budgeted Planner

TRACESHIELD rests on the single idea that the sensitivity argument of Section 3 forces on us: to bound a channel whose sensitivity to one record is unbounded, the release must stop depending on the data. Its primary mechanism is therefore *structure canonicalization*. The runtime forces the live agent (the crew of Figure 1) onto a fixed public plan: a constant number of clinical-extraction passes followed by one coverage assessment, one criteria check with no repair loop, one medical-necessity review with no revision, and one determination. The egress step sequence and its length are therefore identical for every request regardless of $D$ and $q$. This is enforced at runtime: the agent genuinely executes the constant plan rather than having its recorded trace rewritten after the fact.

The two remaining data-dependent coordinates, per-step timing and egress size, are then closed by a hard, data-independent, fail-closed cap: a public per-step deadline $\tau^\star$ and a public egress-size cap $b^\star$. A step that finishes early waits to $\tau^\star$, and a step that would run past it is aborted and emits the canonical value; egress is bounded to $b^\star$ by construction. Either way the host observes exactly $\tau^\star$ and $b^\star$, so the constant is enforced rather than merely assumed. This matters for the guarantee below: because $\tau^\star$ is a real deadline and not merely a number larger than the observed maximum, the release is constant even on a request whose true work would have exceeded it. The cost is real on both sides: early finishers incur a wall-clock wait, and a small rate of aborted overruns can degrade an answer. We report this cost rather than treat the timing overhead as free.

**Proposition 1** (Canonicalized release is unconditionally trace-private)**.** *Let $t^\star$ be the fixed observable produced by structure canonicalization together with timing and size padding to the public ceilings $\tau^\star, b^\star$. If a request is served by emitting $t^\star$, then $\mathcal{M}$ maps every input to the same point mass at $t^\star$; hence for every pair of inputs, differing in any number of patient records and in any part of the query, the two output distributions are identical. Canonicalized release is therefore $(0,0)$-trace-private with respect to both the corpus relation $D \simeq D'$ and the query relation $q \simeq q'$ simultaneously, and with respect to group changes of any size. Consequently the Bayes-optimal adversary, of any computational power and with any auxiliary information, has membership and attribute advantage exactly zero on served requests.*

The proof is immediate: a constant function of the inputs induces identical output distributions on neighbors, so the privacy loss is zero and no $\delta$ is needed. Two points deserve emphasis, because they delimit exactly what we claim. First, the guarantee attaches to the *constant* release: padding timing and size to a public ceiling makes those coordinates constant, and the latency it costs is the price of the $(0,0)$ statement. Second, we make *no* differential-privacy claim for cheaper half-measures such as reordering tool tokens, coarsely bucketing timing, or padding the hop count to a data-dependent schedule. Rounding a raw, data-dependent trace is not post-processing of a private output and has unbounded worst-case loss at bucket boundaries, so rather than present these as certified mechanisms we *measure* the leakage each one leaves (Section 9, Table 7).

The constant release is one end of a design space, and a finite budget recovers some adaptivity. Fix $K$ public plan templates $t_1, \ldots, t_K$, each a fixed data-independent profile (for $K = 3$: a shallow, a medium, and a deep plan). For a request the runtime computes the template $t_i$ closest to the agent's realized depth and then *releases* a template by $K$-ary randomized response, keeping $t_i$ with probability $e^\varepsilon/(e^\varepsilon + K - 1)$ and otherwise choosing a different template uniformly, after which it emits the released template's public profile.

**Proposition 2** (Template randomized response is $\varepsilon$-trace-private)**.** *Releasing a template by $K$-ary randomized response with the keep probability above, and then emitting that template's fixed public profile, is $\varepsilon$-trace-private with respect to both neighbor relations. As $\varepsilon \to 0$ the release approaches the uniform distribution over templates and the observable becomes data-independent, while $K = 1$ recovers the $(0,0)$ constant of Proposition 1.*

The argument is the textbook randomized-response bound: for any two true templates, the output distributions differ by at most $e^\varepsilon$ at every outcome, no matter how a neighboring input changes the true template. The mechanism is therefore $\varepsilon$-differentially private for a corpus edit and a query edit alike. Emitting the released template's profile is post-processing and cannot increase the loss. The utility cost is now graded rather than all-or-nothing: with probability $e^\varepsilon/(e^\varepsilon + K - 1)$ the runtime provisions the agent at its natural depth, and otherwise it over- or under-provisions, so smaller $\varepsilon$ trades answer quality for privacy along the frontier of Section 9. A per-tenant odometer records how much budget a session has spent. When the policy cap is reached, the fail-closed governor blocks the request or routes it to a human gate rather than silently serve a data-dependent trace. Either way, the runtime writes the outcome into the receipt, so a breach is never hidden.

Structure canonicalization has a real utility cost, because the agent can no longer deepen its search when a case warrants it, and timing padding has a real latency cost; the size of both depends on how much the workload needs adaptive depth. The evaluation reports both against a measured baseline rather than asserting they are free. To sweep the privacy-versus-coverage trade-off cheaply, TRACESHIELD also runs as a

post-hoc transform over recorded traces, canonicalizing a chosen fraction of them; we use this mode only to map the frontier, and every headline number comes from the enforcing runtime.

## 6 Trace Receipts

Bounding the leak is not enough for a regulated deployer, who needs evidence rather than assurances. Each request therefore emits a Trace Receipt, a signed and self-describing record that states the enforced scope, the guarantee the release carries, the mechanisms applied, the budget status (`within` or `breach`), and an execution-environment and policy reference. By design, the receipt never prints a small composed $\varepsilon$ for heuristic obfuscation. For a canonicalized request it records the honest fact that the released trace is data-independent, so the guarantee on served requests is $(0,0)$ by Proposition 1. Any residual leakage, such as the number a deployer measures when only structure is canonicalized and timing and size are left real, is carried as an empirical per-deployment estimate, never as a per-request bound. Output DP, if present, is reported as not composed unless the scope specifies composition.

A receipt is canonicalized with sorted keys and a fixed number format, then signed with Ed25519, and a dependency-free verifier checks the signature, the scope, the budget status, and these honesty invariants without touching any payload. Receipts are anchored in a per-tenant hash-chained ledger, so omission or reordering is detectable. On its own, the verifier establishes the integrity and non-repudiation of a signed claim, not a hardware-rooted proof that the confidential VM actually ran the canonical plan. We close that gap with an optional attestation binding (Section 7): when the service runs on an attested confidential VM, the receipt carries the digest of the Microsoft Azure Attestation (MAA) token and the SEV-SNP launch measurement, and only then does its signed `hardware_attestation` flag read true. The verifier accepts a true flag only if the bound, MAA-verified evidence is present and was issued by a genuine Azure attestation host. Off attested hardware the flag stays false and the signer is a clearly labeled demo key; the verifier rejects any receipt that claims otherwise. Figure 2 shows a real receipt emitted by a shielded request in our evaluation.

| `request_id`: 764c06e482d7a9b4f6c9bc3770ddd449 | |
|---|---|
| scope | `control-flow` |
| guarantee | data-independent trace, $(\varepsilon,\delta)=(0,0)$ |
| mechanisms | structure-canon, timing-pad, size-pad |
| plan template | `CANON` |
| budget status | `within` |
| residual leak | attribute AUC $\approx 0.77$ if only structure is canonicalized (empirical) |
| ledger | sequence 583, hash `b1d8ce80`… |
| signature | ed25519, `5oWQ7fsm`… |

Figure 2: A real Trace Receipt (abridged) from a shielded request. The verifier checks the Ed25519 signature, the scope, the budget status, and the honesty invariants, without access to any payload. The certified guarantee is the $(0,0)$ of a data-independent release; any residual is an empirical estimate.

## 7 Attestation, Chain of Trust, and TCB

A signed receipt is only as trustworthy as the key that signs it and the environment that holds that key. This section makes the trust root explicit and states what the host of a CVM can and cannot see, which is the deployment reality behind the threat model of Section 3.

**What the host observes on a CVM.** On an AMD SEV-SNP (or Intel TDX) confidential VM, hardware encrypts and integrity-protects guest memory, so the host cannot read queries, documents, model I/O, or the retrieval index while they are in use. What the host *can* still observe is exactly our channel: the timing and byte size of each boundary-crossing round-trip the guest makes, together with coarse VM-exit and interrupt cadence. Every model call and every external tool call leaves the encrypted VM as ciphertext of an observable size at an observable moment. Our deployment topology makes this precise: the six agents share one CVM, so purely in-VM computation is invisible, and the leak we bound is carried only by the inter-agent steps that actually cross the confidential boundary as egress. Attestation does not touch this surface. It proves *which code* was launched, not *how that code moves* at run time; a fully attested, trusted-blind host still fingerprints the workflow branch from trace shape. That gap motivates TRACESHIELD, and binding TRACESHIELD to attestation lets a receipt certify both facts at once.

**Chain of trust.** The attested trust root is a standard confidential-VM chain, which our artifact walks end-to-end. The silicon vendor root (AMD ARK/ASK signing a VCEK, or the Intel PCK) signs the hardware report; the SEV-SNP quote attests the launch measurement and guest security version over the encrypted VM; the Azure paravisor's vTPM binds an attestation key to that hardware report (its `report_data` endorses the vTPM AK public key), and the AK signs a fresh TPM quote whose qualifying data carries a caller nonce for freshness; MAA validates this evidence and issues an RS256 JWT whose claims include the compliance status and the measurement values. The service reads the guest vTPM NV index that holds the HCL report, extracts the raw hardware report, fetches the VCEK chain from the Azure THIM endpoint, submits the bundle to MAA, and then *independently* re-verifies the returned token: it fetches the provider JWKS, checks the RS256 signature and the `*.attest.azure.net` issuer, and rejects a token whose issuer is not a genuine at-

testation host. Only a token that survives this check binds a hardware root; a self-signed or non-Azure token verifies cryptographically but is reported as *not* hardware-rooted. Finally, the receipt-signing step embeds the token digest and launch-measurement digest, so the receipt is cryptographically tied to the attested boundary rather than merely asserting one.

**Trusted computing base.** The TCB a deployer must trust is: the AMD (or Intel) TEE and its firmware; the Azure HCL paravisor that provides the vTPM; Microsoft Azure Attestation as the verifier of hardware evidence; and, inside the guest, the application code that enforces the shape and signs receipts. That guest code comprises TRACESHIELD, the recorder, the receipt signer, and the attestation client, plus the crew itself; Section 8 quantifies the code size. We deliberately keep the enforcement path small and payload-blind: the recorder and shield see only step type, timing, and egress size, and the receipt/attestation code touches no query, document, or answer, which bounds what a bug in the TCB could disclose. The commercial LLM endpoint is *outside* the TCB, because it is called across the boundary and is itself a host-visible egress; our guarantee is therefore about the trace the guest emits, not about the model provider. We do not include a confidential GPU in this deployment; extending the chain to an attested GPU (H100 CC via SPDM) so that model execution also runs in-TEE is the natural composition and is orthogonal to the trace-shape result.

**Honest degradation.** The same binary runs off Azure. When no vTPM is present the attestation client returns a *simulated* verdict, the receipt keeps `hardware_attestation` false, and the runtime self-report and verifier both refuse to assert a quote. This lets the public artifact be exercised anywhere while never asserting a hardware claim the platform cannot support.

## 8 Implementation

The workload is the real multi-agent prior-authorization crew of Figure 1, PA-CREW, built on LangGraph, replicating the agent roles of Microsoft's Prior-Authorization Multi-Agent Solution Accelerator (an intake and compliance check, a clinical reviewer, a coverage-assessment agent, and a synthesis decision). An intake node validates the request packet and routes it; a clinical reviewer issues adaptive extraction passes over the referral sources and makes a real LLM sufficiency decision after each one; a coverage-assessment node maps the extracted evidence to each payer policy criterion; a criteria-verification node checks that every criterion is source-supported and can loop back; a medical-necessity review flags gaps and can trigger a revision; and a synthesis node issues the final APPROVE or PEND determination. Every agent decision is a live call to a commercial LLM (gpt-4o-mini, via the OpenAI API), whereas retrieval over the private corpus is local and deterministic and therefore contributes no randomness.

The instrumentation layer records only the host-observable trace, namely the egress step sequence, its per-step wall-clock timing, and its per-step egress size. It deliberately excludes in-VM result cardinality, which crosses no boundary, and it counts only bytes that leave the confidential VM, never in-VM document bytes. It records nothing else, so no request, document, assessment, or determination is ever written down. TRACESHIELD, the attack, the receipt format, the ledger, and the verifier total roughly 1,100 lines of dependency-free Python; the crew and the experiment harness add about 1,500 more. Everything is released as an open-source artifact.

**Deployment.** We package TRACESHIELD, the PA-CREW crew, the receipt signer, the verifier, and the attestation module as a single container and provide a one-command deployment onto an Azure Confidential VM (AMD SEV-SNP, DCasv5) with vTPM, Secure Boot, and guest attestation enabled (Bicep plus an `az`-CLI wrapper; teardown included). On that VM guest memory is hardware-encrypted, and the service performs the in-guest attestation of Section 7. So that the posture can be confirmed independently, the service exposes a read-only self-report of its isolation kind, guest kernel, hardware-TEE state, and the receipt's implementation-scope flags, and a live endpoint that re-collects evidence, submits it to Microsoft Azure Attestation, and re-verifies the returned token. The same image runs unchanged off Azure: with no vTPM present it reports a *simulated* verdict and never asserts a quote, so the self-report is honest in both settings.

## 9 Evaluation

Our evaluation answers four questions. Does the trace leak under a metadata-only adversary, with content, access, and output defenses in place (RQ1)? Does the signal truly live in the trace shape (RQ2)? Does TRACESHIELD bound it, and at what cost in utility and latency (RQ3)? And are the receipts sound (RQ4)?

We build a balanced suite over 3 clinical specialties: cardiology, oncology, and psychiatry. Each specialty contributes two prior-authorization topics, and each topic appears in two framings, a routine service and a sensitive, closely-scrutinized one (a specialty or high-cost service); the framing is the attribute label. Every case holds its document count fixed at six regardless of framing or membership, so that a depth difference the agent later shows reflects the LLM's genuine sufficiency decision over content complexity rather than a corpus-size artifact. The sensitive request does name its service and is therefore longer than the routine one, but this cannot leak into the attack: the request stays inside the boundary and is never a trace feature, and the egress size we record is the model's output

only, so a longer request never enlarges any host observable. We do note that the sensitive referrals are also denser than the routine ones, so part of the size and timing signal is that a harder case genuinely requires a longer assessment; that is the behavior we set out to measure rather than a confound to remove. Each case carries, or does not carry, a detail-dense canary patient case note that replaces one distractor, so membership too leaves the document count unchanged. The crew adjudicates the request and issues a determination over its private corpus; we collect 288 adaptive undefended runs and 144 runs under the enforcing TRACESHIELD in its canonical setting. Utility is scored by an independent, stronger judge model (gpt-4o, distinct from the crew's gpt-4o-mini) along faithfulness, completeness, and clarity, and latency is real wall-clock time. We report attack AUC group-aware, training the attacker on all-but-one (service, topic) group and testing on the held-out group so that no task template is shared between train and test; we also report a multi-split mean and standard deviation. The full study cost a few tens of dollars in API usage.

*The trace leaks* (Table 2). From host-observable metadata alone, and with content, access, and output defenses in place, the group-aware attacker recovers the sensitive attribute at AUC 0.679 and canary membership at 0.566; the multi-split estimates agree, at $0.74 \pm 0.10$ and $0.57 \pm 0.05$. The two secrets differ in strength, and we treat them separately rather than average them into one headline number. Attribute inference is the stronger channel, and Figure 3 shows one reason: on the size-matched corpus the agent's mean research depth is still 2.9 hops on routine tasks but 4.7 on sensitive ones, because the real, LLM-made sufficiency decision keeps the agent retrieving when the content raises unresolved complications. The crew runs under a public maximum-depth cap, which sensitive cases reach, so depth alone is a coarse, bimodal signal; the sharper and less cap-dependent signal is egress size, since the agent writes more when the case is harder. Because the document count is held fixed across the two framings, that depth gap is a property of the agent's reasoning over content, not of a larger corpus. Table 3 locates the signal precisely: the attribute is redundantly encoded across egress size (AUC 0.64) and timing (0.63) as well as depth (0.57). Because size and timing carry it independently of structure, a defender who fixes only the control-flow structure does not close the channel; the ablation below confirms this.

Membership is a much weaker channel. Once the in-VM cardinality that an earlier design relied on is excluded as host-invisible, the canary leaves only a faint trace: a marginally larger draft when it is retrieved and cited. Membership AUC is therefore 0.566 group-aware, and no single feature separates it well (Table 3). We report it as a real but modest risk rather than a headline number, and it is attribute inference that carries the RQ1 claim.

The attribute leak is not an artifact of one service: per-service attribute AUC is 0.70, 0.61, and 0.85 for cardiology, oncology, and psychiatry, respectively (Table 5). Nor is it an artifact of one model. On a second crew driven by gpt-4o the same undefended attack reaches attribute AUC 0.96 and membership 0.58, so a different model with a different sufficiency policy exposes the same channel. The mechanism is a property of adaptive agents rather than of one configuration.

Table 2: The Trace-Reconstruction Attack on real host-observable traces: undefended (group-aware), under structure canonicalization alone (residual), and under full padding. Structure alone leaves the attribute leak; full padding closes it.

| Attack target | Undefended (grouped) | Struct. canon. (residual) | Full pad |
|---|---|---|---|
| Membership | 0.566 | 0.504 | 0.500 |
| Attribute | 0.679 | 0.765 | 0.500 |

Undefended multi-split: memb. $0.566 \pm 0.046$, attr. $0.736 \pm 0.102$
Controls: permutation 0.570, shape-fixed 0.500

Table 3: Single-feature ROC-AUC on undefended host-observable traces (pooled). The attribute secret is redundantly encoded across egress size, timing, and depth, which is why a single-axis defense cannot close the channel; membership is weak on every feature.

| Single trace feature | Memb. AUC | Attr. AUC |
|---|---|---|
| hop count | 0.510 | 0.571 |
| step count | 0.543 | 0.647 |
| total timing | 0.559 | 0.633 |
| total egress size | 0.561 | 0.644 |
| distinct tools | 0.500 | 0.500 |

*The recoverable information is structural rather than contentful*, which is what RQ2 asks, and three controls pin that down. The permutation control gives AUC 0.57 and the shape-fixed control gives 0.50 (Table 2); shuffling the labels or erasing the shape both collapse the attack to near chance, ruling out a pipeline artifact and a residual content leak. The group-aware protocol itself is the third control: because the held-out (service, topic) group shares no task template with training, the attribute AUC of 0.679 cannot be template memorization, but only the data-dependent plan generalizing to unseen tasks. What the attacker recovers is the trace shape, exactly the object $(\varepsilon, \delta)$-trace-privacy governs.

TRACESHIELD *closes the leak*, and we quantify its cost in two parts (RQ3). Structure canonicalization alone is cheap but insufficient. Forcing the constant plan reduces membership leakage, yet the attribute attack survives at AUC 0.77 on the structure-canonical observable (Table 2); as the attribution and ablation show, the secret is still carried by the real per-step timing and egress size, which structure canonicalization leaves untouched. Only when timing and size are

also padded to public ceilings does the observable become data-independent, at which point the attack is at chance, 0.500 for attribute and 0.500 for membership; Proposition 1 guarantees this collapse rather than merely reporting it. That closure is not free. Enforcing the public per-step deadline adds about 20.4s of wall-clock time per request, a figure computed from the measured per-step timings (our harness reports this cost rather than physically sleeping each run to the deadline). Mean latency therefore rises from 15.6s at baseline to 36.0s under full padding, whereas structure canonicalization by itself changes mean latency by -16%. About thirteen percent of steps exceed the deadline and would be aborted to the canonical value, a small additional utility cost we do not separately measure. Utility, judged by the independent gpt-4o model, does not degrade under enforcement: enforced runs in fact score somewhat *higher* than baseline (Table 4: overall 0.76 versus 0.84, paired Wilcoxon $p < 0.01$), with faithfulness and completeness moving the same way, consistent with the canonical plan forcing full extraction depth on routine cases that the adaptive baseline cuts short. We read this as no *detected* loss on this workload, not as proof of equivalence; the honest cost of canonicalization is exactly the adaptivity it removes, which this suite happens not to need.

The constant release is not the only setting, and Figure 4 shows the finite-ε template mechanism of Proposition 2 in action, with the per-budget keep probabilities and attack AUCs tabulated in Table 6. Even at a large budget it already releases only a coarse, noised depth bucket, so attribute AUC sits at 0.76 rather than the undefended 0.679, because timing and size are always emitted as constants; as ε falls the released template approaches uniform and the attack reaches chance (0.56 at ε=0.10). The budget interpolates between these extremes: larger ε provisions the agent at its true depth more often; smaller ε trades that fidelity for privacy.

A natural objection is that closing every coordinate is overengineering, and Table 7 answers it by closing each coordinate in isolation over the measured traces. Obfuscating the tool order provides essentially no protection, because the number, timing, and size of the steps are left intact. Padding egress size is the most effective single-coordinate defense yet still leaves attribute AUC near 0.71 through timing, and canonicalizing the whole control-flow structure while leaving timing and size real also leaves the attribute at 0.71. Only closing structure, timing, and size together drives the attack to 0.50. The ablation therefore justifies the design. The intuitive first mitigations, fixing the plan or padding a single axis, each leave the attribute substantially recoverable, and it is the redundancy of the secret across coordinates that forces the whole-shape defense.

*The receipts are sound*, which answers RQ4. All 144 shielded requests produce receipts that verify under the dependency-free verifier, the hash-chained ledger is intact, and a single mutated field is detected because the signature then fails to verify. Each receipt states the honest guarantee,

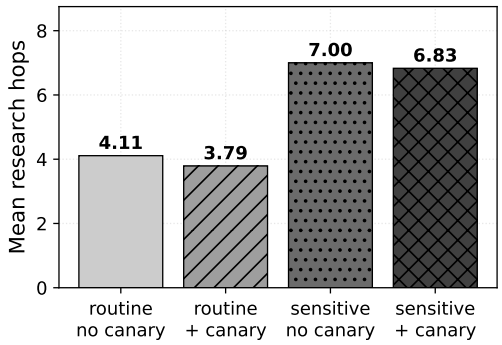


Figure 3: Mean clinical-extraction passes per cell on the size-matched corpus. Even with the document count held fixed across framings, the agent searches deeper on sensitive tasks (4.7 versus 2.9 hops), because the depth is decided by the LLM's genuine sufficiency decision over content, not by corpus size. The sensitive value sits at the public max-hop cap, so depth is a coarse signal; the sharper, cap-independent one is egress size (Table 3).

Table 4: Real utility (independent gpt-4o judge) and latency, baseline versus enforced TRACESHIELD, with the significance test and the full-padding latency cost.

| Condition | Overall | Faithful. | Complete. |
|---|---|---|---|
| Baseline (adaptive) | 0.757 | 0.823 | 0.764 |
| Enforced (canonical) | 0.837 | 0.875 | 0.858 |

Judge: gpt-4o; difference -0.080 (significant, $p \approx 0.00$)

Latency: baseline 15.6s, struct. canon. 13.1s (-16%), full pad 36.0s

that a canonicalized release is data-independent and therefore $(0,0)$-trace-private, and carries the structure-only residual as an empirical estimate rather than a per-request bound.

## 10 Discussion and Limitations

The largest caveat concerns which parts of the system run on real confidential hardware. The *deployment* is a genuine Azure SEV-SNP confidential VM with vTPM attestation verified by MAA (Section 7); the *empirical study* of the channel, however, is conducted on runs that assume the boundary rather than exercise it, and we keep the two apart rather than conflating a working attestation with a hardware-measured leak. For that study we assume the TDX or SEV-SNP boundary together with content, access, and output defenses and grant the adversary only the trace, which strengthens the RQ1 claim because the leak persists even when everything else is ideal. Two further scope limits follow from this split. We do not re-run the full attack and defense evaluation from inside the attested

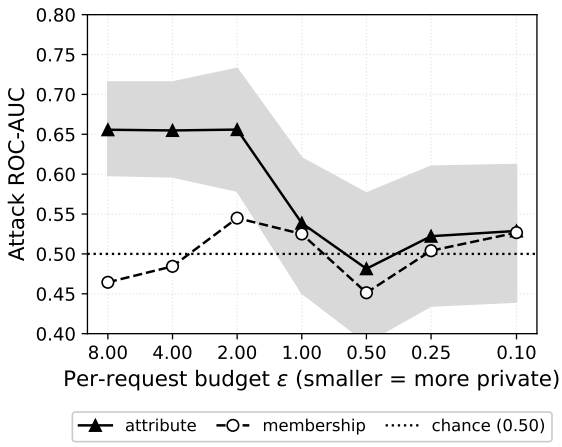


Figure 4: The finite-ε template mechanism (Proposition 2): attack AUC against the per-request budget ε. Even at a large budget only a coarse, noised depth bucket is released (attribute well below the undefended value, since timing and size are always constant), and as ε shrinks the attack reaches chance.

Table 5: The attribute attack generalizes across clinical specialties (group-aware, undefended traces).

| Service | $n$ | Memb. AUC | Attr. AUC |
|---|---|---|---|
| Cardiology | 96 | 0.572 | 0.703 |
| Oncology | 96 | 0.541 | 0.609 |
| Psychiatry | 96 | 0.572 | 0.849 |

VM, because doing so would not change the trace-shape result, which is a property of the agent's behavior rather than of the silicon. We also do not defend the content-level channels (KV-cache and token-length timing) that a co-tenant or network adversary can mount; those are orthogonal to trace shape and we do not claim to close them. We were also deliberately careful about which observables are real. Because retrieval runs inside the CVM, per-access result cardinality crosses no boundary, and we exclude it; we count only egress bytes. The surface we attack is therefore the one a host actually sees, not an in-VM quantity that would inflate the leak. The honest consequence is that our timing is application-level, driven by model round-trips, not microarchitectural; on real hardware the host would see additional, finer channels [16], which can only add leakage. The counterpart is our defense's overhead. Closing the timing channel means padding every step up to a public wall-clock ceiling, which is why full padding roughly doubles latency; we report that rather than hide it. Off-the-shelf network padding closes only the size and rate channel, leaving the data-dependent step count and per-step timing, which our structure canonicalization removes.

We are equally explicit about the shape of the finding. The

Table 6: The finite-ε template mechanism: keep probability and attack AUC across per-request budgets ε.

| ε | keep prob. | Memb. AUC | Attr. AUC |
|---|---|---|---|
| 8.00 | 1.00 | 0.521 | 0.760 |
| 4.00 | 0.96 | 0.536 | 0.750 |
| 2.00 | 0.79 | 0.543 | 0.704 |
| 1.00 | 0.58 | 0.548 | 0.648 |
| 0.50 | 0.45 | 0.563 | 0.575 |
| 0.25 | 0.39 | 0.565 | 0.574 |
| 0.10 | 0.36 | 0.561 | 0.563 |

Table 7: Mechanism ablation (pooled attacker-favorable split): residual attack AUC when each trace coordinate is closed alone over the measured traces. Closing structure, timing, or size in isolation leaves the attribute leak nearly intact; only closing all three drives it to chance.

| Coordinate closed alone | Memb. AUC | Attr. AUC |
|---|---|---|
| none (undefended) | 0.566 | 0.736 |
| blur tool order only | 0.565 | 0.734 |
| pad timing only | 0.588 | 0.747 |
| pad egress size only | 0.535 | 0.760 |
| canonicalize structure only | 0.556 | 0.707 |
| full (structure + timing + size) | 0.500 | 0.500 |

attribute channel is strong and robust: it holds under group-aware evaluation across unseen topics, across three services, and across a second model. Membership, by contrast, is weak once we exclude the in-VM cardinality that an earlier design relied on, so we lead with attribute and report membership as a modest residual risk. The strength of the attribute leak comes from a real behavior, namely that the agent writes more and takes longer on genuinely harder cases even when the corpus size is held fixed, and that same behavior is exactly the adaptivity a defender sacrifices under canonicalization. On this workload the sacrifice was free in measured utility, but that is partly because the tasks did not truly require adaptive depth. On harder tasks where adaptive depth is essential, a real utility cost would appear; this is the trade-off the coverage frontier exists to characterize, and quantifying it on adaptivity-critical workloads is the natural next step.

Two boundaries of the guarantee deserve restating. First, the $(0,0)$ statement is for the fully canonicalized release; the cheaper structure-only defense leaves a measured residual, and the receipt states which of the two a deployment ran rather than overclaiming. Second, our attacker is passive and offline: a single-stepping or page-fault adversary reads intra-VM branches that no application-level defense can hide. Canonicalization defends against the host we model by construction, since a constant release is data-independent, but robustness against active and adaptive adversaries is future work.

## 11 Conclusion

Autonomous agents leak their secrets through the shape of what they do, not only through the content of what they say. We formalized $(\varepsilon, \delta)$-trace-privacy over the observable agent trace. On a real multi-agent prior-authorization crew, and on a second model, we demonstrated that this channel stays open even behind a TEE, oblivious retrieval, and output DP. We then showed that canonicalizing the control flow to a data-independent shape and padding timing and size closes the channel with a $(0, 0)$ guarantee, at a latency cost we measure rather than assume away, while emitting verifiable evidence. On an Azure Confidential VM that evidence is bound to a hardware attestation verified by Microsoft Azure Attestation, so the certified shape and the attested code are checked together. The ablation makes the design's necessity concrete: because the secret is written redundantly into structure, timing, and size, closing any one of them is not enough. As agentic AI moves into hospitals and other regulated settings, bounding and certifying the behavioral trace is a missing and necessary layer.

## Ethical Considerations

We analyzed the ethics of this work against the stakeholders it affects: the patients whose records a deployed system would process, the hospitals that operate such systems, the cloud and model providers that host them, and the research community.

*No human subjects and no real data.* The corpus is entirely synthetic and fictional, authored for this study, and contains no protected health information. No human subjects were involved, no personal data was collected, and the workload is not a clinical decision-support system and produces no medical advice. The experiments run against a system we built, not against any third party's service or users.

*Dual use and disclosure.* The Trace-Reconstruction Attack is offensive in form, so we considered its potential for misuse. The channel it exploits is not a vulnerability in any specific vendor's product; it is an inherent consequence of adaptive agent control flow, present in principle in any confidential-computing deployment of a reason-and-act agent. There is therefore no single vendor to notify, and the risk of publication enabling new harm is low relative to the benefit, because the attack requires only host-observable metadata that a cloud operator already possesses. We publish the attack alongside a deployable defense that closes it, so the net effect is to reduce rather than increase exposure. We do not release any tooling specialized for a particular production target, and the demonstration operates only on synthetic data.

*Beneficence.* The intended beneficiaries are operators of confidential agentic systems over sensitive corpora in regulated settings, who currently have no way to bound or certify this leakage. By making the residual channel measurable and the defense verifiable, the work supports data-minimization and auditability obligations rather than undermining them. We judged the balance of benefit over harm to be clearly favorable and proceeded on that basis.

## Open Science

We are committed to open science and release the full artifact required to evaluate the paper's contributions: the six-agent prior-authorization crew, the metadata-only trace recorder, the Trace-Reconstruction Attack with its statistical controls, the three TRACESHIELD defense modes, the Ed25519 Trace Receipts and hash-chained ledger, the independent verifier, the attestation module and its Azure/MAA verification path, a one-command Azure Confidential VM deployment, and the synthetic corpus. The artifact runs fully offline on a deterministic fixture provider (no key or network required) and, under a configured commercial-model key, performs a fresh live replication.

*Provenance, stated honestly.* The numeric results in this submission are anchored to a specific artifact state and separated by provenance, and the artifact enforces this separation rather than trusting prose. The fixture provider is a deterministic plumbing harness whose outputs are labeled `synthetic_fixture` and must not be read as evidence that a real agent leaks. A run against the configured models is labeled `live_replication` and is a fresh measurement, never a relabeling of any prior result. Attestation is hardware-backed only on a genuine confidential VM and is otherwise reported as `simulated`; the receipt verifier rejects any receipt that overclaims.

*Reproducibility.* The attack-derived tables and numeric macros are generated in place from an experiment journal by `scripts/make_paper_artifacts.py`, so a fresh run regenerates every attack, control, ablation, frontier, attribution, and per-service number and the manuscript reflects it exactly; utility and latency additionally require a judged live run. The offline functional gate is `docker compose up -build` (or `uv run pytest`); the enforcing study is `traceguard experiment`, whose journal drives the generator. The attestation deployment is exercised by `deploy/azure/deploy.sh`. To respect double-blind review, the artifact is provided through an anonymized repository whose link appears on the submission page, and no artifact URL or metadata reveals author identity.

## References


[1] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proc. 2016 ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, pages 308–318, 2016.

[2] Anonymous. NetEcho: Reconstructing LLM conversations from encrypted network traffic. *arXiv preprint arXiv:2510.25472*, 2025.

[3] Quinn Burke, Anjo Vahldiek-Oberwagner, Michael Swift, and Patrick McDaniel. It's a feature, not a bug: Secure and auditable state rollback

for confidential cloud applications. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[4] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *Proc. 30th USENIX Security Symposium*, pages 2633–2650, 2021.

[5] David Cash, Paul Grubbs, Jason Perry, and Thomas Ristenpart. Leakage-abuse attacks against searchable encryption. In *Proc. 2015 ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, pages 668–679, 2015.

[6] Victor Costan, Ilia Lebedev, and Srinivas Devadas. Sanctum: Minimal hardware extensions for strong software isolation. In *Proc. 25th USENIX Security Symposium*, pages 857–874, 2016.

[7] Zehang Deng, Haoyang Li, Wanlun Ma, Ruoxi Sun, Derui Wang, Minhui Xue, Haibo Hu, Sheng Wen, and Yang Xiang. The prompt stealing fallacy: Rethinking metrics, attacks, and defenses. In *Proc. 35th USENIX Security Symposium*, 2026.

[8] Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. Flocks of stochastic parrots: Differentially private prompt learning for large language models. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

[9] Nicolas Dutly, Friederike Groschupp, Ivan Puddu, Kari Kostiainen, and Srdjan Capkun. AEX-NStep: Probabilistic interrupt counting attacks on intel SGX. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[10] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *Proc. 51st IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 51–60, 2010.

[11] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail. In *Proc. 2012 IEEE Symposium on Security and Privacy (S&P)*, pages 332–346, 2012.

[12] Zachary Espiritu, Seny Kamara, Tarik Moataz, and Valentin Ogier. Leafblower: A leakage attack against TEE-based encrypted databases. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[13] Zibo Gao, Junjie Hu, Feng Guo, Yixin Zhang, Yinglong Han, Siyuan Liu, Haiyang Li, and Zhiqiang Lv. I know what you said: Unveiling hardware cache side-channels in local large language model inference. In *Proc. 34th USENIX Security Symposium*, 2025.

[14] Zhuohan Ge, Haoyang Li, Yubo Wang, Nicole Hu, Chen Jason Zhang, and Qing Li. ClinicalAgents: Multi-agent orchestration for clinical decision making with dual-memory. In *Proc. 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2026.

[15] Philipp Giersfeld, Benedict Schlüter, and Shweta Shinde. BREAKFAST: Confused deputy attack on infinity fabric to break AMD SEV-SNP. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[16] Tristan Hornetz, Hosein Yavarzadeh, Albert Cheu, Adria Gascon, Lukas Gerlach, Daniel Moghimi, Phillipp Schoppmann, Michael Schwarz, and Ruiyi Zhang. TDXRay: Microarchitectural side-channel analysis of intel TDX for real-world workloads. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[17] Junpyo Jeong et al. Network-level prompt and trait leakage in local research agents. In *Proc. 35th USENIX Security Symposium*, 2026.

[18] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 9459–9474, 2020.

[19] Mengyuan Li, Yinqian Zhang, Huibo Wang, Kang Li, and Yueqiang Cheng. CIPHERLEAKS: Breaking constant-time cryptography on AMD SEV via the ciphertext side channel. In *Proc. 30th USENIX Security Symposium*, 2021.

[20] Jiacheng Liang, Yuhui Wang, Changjiang Li, Tanqiu Jiang, Rongyi Zhu, Neil Zhenqiang Gong, and Ting Wang. GraphRAG under fire. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[21] Ruixuan Liu, David Evans, and Li Xiong. Beyond indistinguishability: Measuring extraction risk in LLM APIs. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[22] Yupei Liu, Yuqi Jia, Runpeng Geng, Jinyuan Jia, and Neil Zhenqiang Gong. Formalizing and benchmarking prompt injection attacks and defenses. In *Proc. 33rd USENIX Security Symposium*, 2024.

[23] Jesse De Meulemeester, David Oswald, Ingrid Verbauwhede, and Jo Van Bulck. Battering RAM: Low-cost interposer attacks on confidential computing via dynamic memory aliasing. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[24] Microsoft Security Research. Whisper leak: A novel side-channel attack on remote language models. *arXiv preprint arXiv:2511.03675*, 2025.

[25] Priyanka Nanayakkara, Elena Ghazi, and Salil Vadhan. Making privacy public: Toward a differential privacy deployment registry. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[26] Ali Naseh, Yuefeng Peng, Anshuman Suri, Harsh Chaudhari, Alina Oprea, and Amir Houmansadr. Riddle me this! stealthy membership inference for retrieval-augmented generation. In *Proc. 2025 ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, 2025.

[27] Ryan M. Rogers, Aaron Roth, Jonathan Ullman, and Salil Vadhan. Privacy odometers and filters: Pay-as-you-go composition. In *Advances in Neural Information Processing Systems 29 (NeurIPS)*, pages 1921–1929, 2016.

[28] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

[29] Benedict Schlüter, Christoph Wech, and Shweta Shinde. FABRICKED: Misconfiguring infinity fabric to break AMD SEV-SNP. In *Proc. 35th USENIX Security Symposium*, 2026.

[30] Yuhao Shen et al. Observable channels, not just storage: Evaluating privacy leakage in LLM agent pipelines. *arXiv preprint arXiv:2603.22751*, 2026.

[31] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023.

[32] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *Proc. 2017 IEEE Symposium on Security and Privacy (S&P)*, pages 3–18, 2017.

[33] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In *Proc. 10th USENIX Security Symposium*, 2001.

[34] Emil Stefanov, Marten van Dijk, Elaine Shi, Christopher W. Fletcher, Ling Ren, Xiangyao Yu, and Srinivas Devadas. Path ORAM: An extremely simple oblivious RAM protocol. In *Proc. 2013 ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, pages 299–310, 2013.

[35] Haochen Sun, Jason Li, and Hongyang Zhang. zkLLM: Zero knowledge proofs for large language models. In *Proc. 2024 ACM SIGSAC Conf. on Computer and Communications Security (CCS)*, pages 4405–4419, 2024.

[36] Roy Weiss, Daniel Ayzenshteyn, Guy Amit, and Yisroel Mirsky. What was your prompt? a remote keylogging attack on AI assistants. In *Proc. 33rd USENIX Security Symposium*, 2024.

[37] Shuhua Yang, Jiahao Zhang, Yilong Wang, Dongwon Lee, and Suhang Wang. Query-efficient agentic graph extraction attacks on GraphRAG systems. In *Proc. 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

[38] Baolei Zhang, Haoran Xin, Yuxi Chen, Zhuqing Liu, Biao Yi, Tong Li, Lihai Nie, Zheli Liu, and Minghong Fang. Who taught the lie? responsibility attribution for poisoned knowledge in retrieval-augmented generation. In *Proc. 2026 IEEE Symposium on Security and Privacy (S&P)*, 2026.

[39] Jinhao Zhu, Liana Patel, Matei Zaharia, and Raluca Ada Popa. Compass: Encrypted semantic search with high accuracy. In *Proc. 19th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 915–938, 2025.

# A Artifact Appendix

**Abstract.** The artifact is the complete system of this paper: the six-agent PA-CREW prior-authorization crew on LangGraph (replicating Microsoft's Prior-Authorization Multi-Agent Solution Accelerator), the metadata-only trace recorder, the Trace-Reconstruction Attack with its statistical controls, the three TRACESHIELD defense modes, Ed25519 Trace Receipts on a hash-chained ledger, the attestation module and its Azure/MAA verification path, a FastAPI + React research console, and a one-command Azure Confidential VM deployment. It runs fully offline on a synthetic, fictional prior-authorization corpus using a deterministic fixture provider; a live-model replication and a hardware-attested deployment are optional.

**Scope and honesty labels.** The artifact evaluates three provenance classes and never conflates them. *Paper-reported (unverified)* values are transcribed from the manuscript and shown only as reference. *Synthetic-fixture* runs are deterministic plumbing tests, not evidence for the empirical claim. *Live replication* is a fresh run against the configured commercial models, labeled as a new run rather than the original record. Likewise, the attestation verdict is hardware-backed only on a real confidential VM and is otherwise a clearly labeled *simulated* result; the receipt verifier rejects any overclaim.

**Requirements.** Offline gate: Docker 24+, or Python 3.11/3.12 with `uv`. Live replication: a rotated commercial-model API key, injected at runtime only. Attested deployment: an Azure subscription with confidential-VM (DCasv5) quota, the `az` CLI, and an SSH key. No specialized hardware is needed for the offline and fixture gates.

**Set-up and functional gate.**

```
# console at :8080
docker compose up --build
# or, without Docker:
uv sync --extra dev
uv run traceguard doctor
uv run pytest
uv run traceguard experiment \
    --provider fixture \
    --n-per-cell 2 --seed 20260710
```

Each run writes an immutable directory under `artifacts/runs/` with resolved config, dataset and code hashes, events, traces, receipts, and a signed manifest.

**Major claims and how to exercise them.** (C1) The trace leaks under a metadata-only adversary: the *Attack lab* console page, or `POST /api/analysis/attack`, runs the leave-one-group-out attack and reports AUC with controls. (C2) TRACESHIELD closes the channel: the *Defenses* page runs the same case through adaptive, structure-only, and full-pad and compares observables; full-pad collapses the attack to chance. (C3) Receipts are sound: the *Receipts & ledger* page verifies signatures and detects tampering and chain breaks. (C4) Attestation is real and honest: the *Attestation* page (or `GET /api/attestation` and `POST /api/attestation/verify`) collects evidence, validates the MAA token, and reports a verdict, which is *simulated* off Azure and hardware-backed on a confidential VM.

**One-command attested deployment.**

```
cd deploy/azure
# set subscription, region, size, REPO_URL:
cp deploy.env.example deploy.env
./deploy.sh       # prints the frontend URL
# open http://<fqdn>/attestation, Verify
./teardown.sh     # remove all resources
```

The deployment provisions a SEV-SNP CVM with vTPM and Secure Boot, runs the container with the guest vTPM mounted, and yields a URL whose `/attestation` page shows a Microsoft Azure Attestation-verified SEV-SNP quote and its measurement values.

# B Attestation and Verification Details

**Evidence collection.** Azure confidential VMs hide the raw SEV-SNP guest device behind the HCL paravisor, so evidence flows through the guest vTPM rather than a `/dev/sev-guest` ioctl. The service reads NV index `0x01400001`, which holds the HCL report, and extracts the raw hardware report (1184 B for SEV-SNP, 1024 B for TDX) at offset 32. It fetches the VCEK certificate chain from the Azure THIM endpoint (`169.254.169.254/metadata/THIM/amd/certification`). The general-purpose IMDS "attested document" is deliberately *not* used: it is signed by the host, not the CPU, and is not a TEE quote.

**MAA submission and token.** The evidence bundle is posted to `{provider}/attest/SevSnpVm`; the provider returns an RS256 JWT whose claims include `x-ms-compliance-status` (expected `azure-compliant-cvm`), `x-ms-sevsnpvm-launchmeasurement`, `-idkeydigest`, `-guestsvn`, and `-reportdata`. Policy should key on the compliance status and the ID-key digest, which Microsoft rotates slowly; the launch measurement changes with every paravisor or UEFI update and is recorded, not pinned.

**Independent verification.** The client re-verifies the token rather than trusting the issuer blindly: it fetches the JWKS from `{provider}/certs`, extracts the RSA public key from the `x5c` certificate, checks the RS256 signature, and requires the `iss` host to end in `.attest.azure.net` and to match the pinned provider. A token that verifies cryptographically but carries a non-Azure issuer is reported as *not* hardware-rooted. The console's "Verify" action reproduces this check live and returns the signature, issuer, and hardware-rooted booleans.

**Receipt binding and enforced invariants.** If and only if a token is hardware-backed, MAA-verified, and Azure-issued, the receipt embeds `token_sha256` and `measurement_sha256` and sets its signed `hardware_attestation` flag true. The verifier applies this rule symmetrically: it accepts a true flag only with that bound evidence, and continues to reject any receipt that marks a TEE, oblivious retrieval, output DP, or transport padding as implemented, since those remain assumptions the artifact does not provide. This is the mechanism that keeps the public artifact from turning a deployment assumption into a claim.