GHOST TOOL CALLS: ISSUE-TIME PRIVACY FOR SPECULATIVE AGENT TOOLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-augmented agents are beginning to execute likely future tool calls speculatively to hide latency. When the agent abandons a speculated branch, those calls have already reached external services, carrying what the runtime guessed the user wanted. We call them *ghost tool calls*. Because no provider can be made to forget what it has logged, privacy must be decided when a speculative call goes out, not when its branch is dropped. We put the decision there: Speculative Tool Privacy Contracts label each proposed call and settle it before dispatch. What speculation gives away is both large and early. On a third-party agent benchmark, an observer who sees the agent's outgoing calls names what the user is deliberating about on 68% of tasks before the user confirms anything, against 21% from the calls the agent actually ends up making. Of the ghost calls that carry an intent at all, 31% name one the user never had. The leak recurs on two further domains and, independently, under a second speculator. The leak is separable from the speedup. Holding back only the sensitive speculative calls until the agent commits takes what an observer learns from the abandoned calls alone from 29.8% of tasks to 0.4%, while still allowing the agent to finish as many tasks and reuse every speculative call it would have. This costs latency: on live search, a median 6.8 points of the sequential baseline. The closest existing defense, restricting speculation to read-only calls, does not reach it: that helps only when the intent shows in which write the agent picks, and still leaks when the intent sits in what the agent reads.

1 INTRODUCTION

Tool-augmented language agents act through external systems: they search the web, retrieve documents, inspect calendars, query CRMs, and call application APIs. Each call creates a privacy surface beyond the final answer: its fact, timing, destination, and content can reveal information about the user or organization. To hide the latency of these calls, agent runtimes increasingly *speculate*, issuing the calls the agent is likely to need before it has decided to make them (§2).

Speculative tool execution widens that surface. A user might ask, "What's the penalty for breaking my lease early?" (Figure 1). A latency-optimized agent fans out a tree of speculative queries: a search for tenant law, which the agent eventually uses; a sideways search for available apartments; and a deeper chain through eviction law into a personal-loan-rates API. The final reply uses the tenant-law search. The apartment and loan-rate branches are abandoned, but the receiving providers have already logged them. The speculation explored a scenario that the user never requested: the agent guessed the user might be looking to leave, might be facing eviction, and might need a loan to cover it. This speculation is not visible to the user: the transcript that surfaces in the committed branch looks innocuous. But the abandoned branches present a potentially discrediting scenario to providers.

Speculation also persists across turns, as in Figure 1's second turn. Next to the committed trace lives a graph of branches the runtime explores to anticipate what the user might need. A call whose result the committed trace uses is *promoted*. A speculative agent dispatches more calls than it uses, and we call the abandoned externally observable ones *ghost tool calls*: issued, never promoted, and impossible to retract. Each ghost call deposits *observer-visible state* (provider logs, caches, audit sinks) from which an external observer can recover *agent-inferred intent*: the topics, entities, destinations, and follow-up actions the runtime predicted the user might need.

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

The cost of speculative tool calls is usually discussed in terms of overhead: abandoned branches cost tokens and other compute resources. This paper asks what speculation costs the user in privacy, and whether that cost can be removed without giving up the gain. The question has been overlooked: we are not aware of prior work that formulates or evaluates privacy policy at the speculative-issue boundary. Existing defenses decide what the agent may do or may keep, never what an issued call has already revealed (§2). The issue-time principle follows: for externally observable tools, the runtime must decide privacy policy before a speculative call leaves the trusted boundary, since commit-time cleanup has no privacy semantics once an observer has recorded the call.

We propose Speculative Tool Privacy Contracts to enforce this principle. A contract labels a proposed event by execution mode, observer, sensitivity, destination, confidence, and audit policy. Before dispatch, the monitor decides whether the call issues unchanged, transforms, routes to a trusted local substitute, defers until promotion, or blocks. The design treats observation before commitment as an effect distinct from state mutation and is deliberately simple: our contribution is the problem, its trace model, and the measurement of what speculation discloses and what withholding costs.

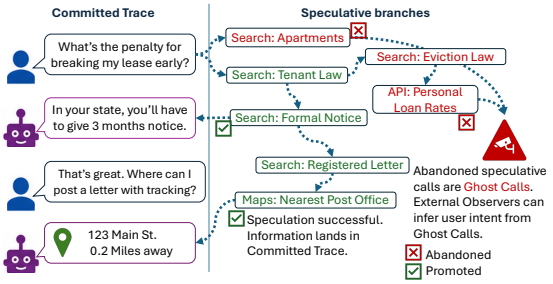

Figure 1: Motivating trace: the two-turn dialogue the user sees (left) and the runtime's tree of dispatched tool calls (right). Green branches feed the answer; red ones are abandoned ghost calls.

Our benchmarking setup combines the speculator released with Speculative Actions on τ^2-bench retail and airline and τ-Knowledge banking, with AgentDojo, HotpotQA, generated corpora, and live web search. At one guess per step it abandons 46% (retail) to 83% (banking) of its branches, leaving 1,703 provider-observed ghost tool calls over 522 τ^2 runs; at five, retail alone leaves 4,387. We ask three questions of them. *Q1: how much does pre-commit disclosure leak?* Before the user confirms, a schema classifier over tool identity already names the intent on 68% of τ^2 retail tasks, against 21% from the committed plan alone; 31% of intent-bearing ghost calls imply an intent the user never had; and the leak recurs on all three domains and under a second speculator. *Q2: which policy reduces the leak most?* Only issue-time controls that suppress or transform the intent-bearing projection, and the action must match the channel: read-only speculation removes the leak when intent rides on which write is called, and leaves it when intent rides in the text of a read. *Q3: what privacy–utility trade-off results?* A middle ground exists: on retail, Shadow + reissue cuts ghost-only recovery to 0.4% while serving as many promoted calls from speculation as Naive, and on banking, where intent rides in reads, it closes the channel this speculator leaves SIA open on, while still serving 64% of them.

We make three contributions:

- We identify ghost tool calls and the issue-time vs commit-time distinction, and formalize them in a trace model that compares the issued trace against a committed-only counterfactual.
- We propose Speculative Tool Privacy Contracts, an issue-time runtime abstraction with concrete labels, five actions, and a pre-commit soundness invariant.
- We measure the privacy–utility trade-off on τ^2-bench retail and airline and τ-Knowledge banking, with paired generated-corpus controls, AgentDojo, HotpotQA, and live Brave and DuckDuckGo providers, and find a middle ground: withholding sensitive speculative calls until commitment removes nearly all ghost-call leakage while keeping most of speculation's benefit, provided the action matches the channel that carries intent. We release the runtime, frozen frontiers, and adversaries.

2 Background and Motivation

Speculative tool execution. In the standard synchronous loop the model proposes a call, the runtime executes it, and the model waits, walking a committed trace one call at a time. The standard escape is lookahead, as in branch prediction and prefetching; agent runtimes apply it at the tool boundary, issuing likely next calls while the planner is still deciding. That is *speculative tool execution*, and the predicted invocations are *speculative tool calls*. The defining property is abandonment risk: parallel

calls all belong to a committed plan, whereas a speculative call may be abandoned after it has been sent. Speculative Actions pre-executes predicted API calls and reports up to 20% lower latency (Ye et al., 2026), Speculative Interaction Agents (SIA) issue read-only calls before the turn is confirmed and discard the ones later invalidated (Hooper et al., 2026), and other research runtimes dispatch tool calls ahead of the model's decision (Sui et al., 2026; Guan et al., 2026; Hua et al., 2025; Nichols et al., 2025); a separate line speculates only over the call's tokens, not whether it is issued (Xia et al., 2026). Production stacks speculate too, on three different boundaries. LiveKit and Deepgram speculate toward the model provider, which our threat model places out of scope (LiveKit, 2026; Deepgram, 2025). incident.io speculates at the tool boundary this paper studies, dispatching tool calls from keywords in the user's message and canceling the ones it does not need (incident.io, 2025), and a voice-platform vendor prescribes the same pattern against an external weather API, calling the mispredicted request one whose "only cost was a wasted API call" (Stream, 2025). That accounting is what this paper disputes: the call is wasted for the agent and retained by the provider. The pattern predates agents: Chrome has prefetched result pages for unselected suggestions for years, so search engines already receive queries no user submitted (The Chromium Projects, 2024). Agents do not change whether speculative requests are abandoned, but what an abandoned one reveals, since it carries the runtime's inference about why the user might need it. We measure what that costs, and no published system we know of joins deployment, an agent, a tool boundary, an external receiver, and abandonment in one measurement (App. C.1). The speculator we evaluate shows the pattern in its authors' published runs: 1,381 of 3,245 speculative calls are never made by the real agent, and 24 hand a customer's ID and email to the human-handoff tool in conversations that never escalate.

Why existing defenses miss it. These runtimes reason about correctness or state, none about what an issued call discloses. They separate safe from unsafe speculative calls by task-state mutation, discarding a losing branch that left task state unchanged, and transactional runtimes can delay or compensate writes (Mohammadi et al., 2026); neither recalls an observation, because the observer-visible state of §1 persists at each external observer whether or not the branch is committed. Asynchronous function calling (Feng et al., 2026) never abandons a dispatched call, so it creates no ghost calls. Standard unsafe-tool framing misses the temporal mismatch: a call that is appropriate once the user has asked is not appropriate before. Access control answers whether the agent *may* call a tool; speculative privacy asks whether it may reveal a tentative branch, destination, or argument before commitment. IFC and guardrail systems (Costa et al., 2025; Cai et al., 2026; Zhong et al., 2025; Mou et al., 2026; Doshi et al., 2026; Xie & Li, 2026; Denning, 1976; Myers, 1999) do decide before dispatch, but on the *committed* plan: the release they price is one the agent has chosen. Nothing prevents IFC from carrying issue-time labels; what these systems lack is a notion of speculative commitment. The closest baseline is SIA, whose read-only restriction does not transform what reaches the observer: it removes the inference only when intent rides on which write tool is chosen, and leaves it intact when intent rides in the arguments of a read (Proposition 2). Tool-privacy benchmarks (Hu et al., 2026; Zharmagambetov et al., 2025; Qiao et al., 2025; Huang et al., 2026; El Yagoubi et al., 2026; Debenedetti et al., 2024; Ruan et al., 2024; Zhan et al., 2024; Yao et al., 2025) cannot host this experiment: none runs a speculator, so none contains an abandoned call to measure.

Adjacent privacy lines. Speculative side channels in hardware and decoding (Wei et al., 2024; Kocher et al., 2019), and traffic fingerprinting of agent interactions (Zhang et al., 2025), carry the Spectre lesson to a different object; query privacy (Chor et al., 1998; Howe & Nissenbaum, 2008; Zeng et al., 2024) reduces what a service learns from an issued query, where we ask whether to issue it; contextual integrity (Nissenbaum, 2004) supplies the timing-and-purpose distinction. Table 36 (Appendix E) places this work against each line; the contract adds the observation axis to whichever speculation regime a runtime adopts.

3 Ghost Calls and the Privacy Boundary

3.1 Trace model

Let an agent runtime maintain candidate branches and label each tool event e before dispatch. The *issued trace* S is the ordered log of every tool event the runtime dispatches under its policy; its committed part $S_C \subseteq S$ is the ordered subsequence whose results or side effects the runtime ultimately uses. A *speculative frontier* is the set of candidate calls issued before the runtime has

committed to the branch requiring them. A speculative call is *promoted* if its result becomes part of S_C and *abandoned* otherwise. A *ghost tool call* is an abandoned speculative event with a nonempty externally observable projection.

Disclosure channels. A tool event can disclose through three channels. *Arguments*, the literal payload of the call, often carry the sensitive fact directly: a speculative search for "tenant rights lease termination penalty" reveals the user's frame to the provider whether or not the result is used. *Destination*, the endpoint, host, or collection, can disclose topical intent independent of payload: a retrieval routed to an HR-leave, oncology, immigration, or security collection encodes the agent's interpretation of the task even with an empty query, so collection layout and access-control design become part of the privacy surface. *Metadata*, including tool name, sizes, order, timing, retry counts, cache state, and branch traces, is deployment-sensitive but exploitable when destinations discriminate, caches are shared, or logs preserve branch status.

Observer projections. For observer class i, write $O_i(\cdot)$ for the projection of a trace into the channels observer i can recover; different observers preserve different subsets (Table 2). Then $O_i(S)$ is what i records and $O_i(C)$ is the *committed-only counterfactual*: what i would have seen had the runtime issued only C from the start. This is not S restricted to C, since speculative events perturb provider-side timing, cache state, retry counters, and rate-limit ledgers. The *marginal exposure* Δ_i is taken under an alignment π that pairs each committed event of S with its counterfactual execution in C: unaligned events, the ghost calls, contribute their whole (arguments, destination, metadata) tuple, and aligned events contribute only the fields on which the two differ, which is where the timing and rate-limit perturbations live. Throughout, C is the committed trace of the unconstrained non-speculative agent, so the comparison is against what the user's own plan would have disclosed and not against a sanitized version of it; $C = S_C$ only when the policy leaves committed calls unchanged. When the question is semantic rather than field-level, we measure *inference advantage*: the increase in an adversary's probability of recovering a sensitive label given $O_i(S)$ versus $O_i(C)$.

Promoted versus abandoned. *Ghost calls* (abandoned speculative events) lie in $S \setminus S_C$ by construction, so their projections contribute to Δ_i whenever observer i records anything about them. Promoted speculative events appear once on the wire whether the agent speculated or not, so they enter Δ_i only against observers that preserve ordering or timing. Throughout, *ghost tool call* names the strict abandoned-event case and *pre-commit disclosure* the broader category that also covers promoted speculative events under timing-sensitive observers. The monitor in §4 evaluates every speculative event at issue time regardless of later promotion, because a disclosure cannot be retracted. A worked trace with S, C, $O_i(S)$, $O_i(C)$, and Δ_i is in Appendix B.1.

Proposition 1 (No commit-time erasure). *Assume observer i persists observations outside the runtime's commit boundary. If a ghost call g emits a nonempty projection $O_i(g)$ at issue time, no policy that executes only after branch commitment can remove that projection from observer i's prior view.* Once a provider, network observer, shared cache, or audit log has received the external trace of g, the runtime cannot retroactively unsend it. Rollback works for state effects because a runtime can delay or compensate writes (Mohammadi et al., 2026); observation effects sit outside that boundary. Issue-time is therefore the only enforcement point for externally observable speculative dispatch.

Proposition 2 (Read-only is not observation-free). *Read-only authorization does not imply privacy preservation under speculation: read-only ghost calls can emit nonempty observer projections through any of the three channels above.* Designs that permit speculation only for read-only tools (Hooper et al., 2026) protect against state mutation, not observation.

3.2 Threat model

Adversary. An observer of class i that records what it receives and keeps it: tool providers first, then the network, shared caches, runtime logs, and audit sinks of Table 2. It is passive and honest-but-curious, and we grant it the deployed contract in the Kerckhoffs sense, with the tool schemas and label space. Colluding observers are modeled as one observer holding the union of their projections. On τ^2 most calls reach the retailer the user already talks to, but escalation summaries and banking searches reach separate sinks. Adversaries that instead manipulate the agent's context or exploit its

denial feedback (Chinaei, 2026; Dong et al., 2026; Shim et al., 2026) are complementary to, and orthogonal from, the timing question we ask.

Security goal. An observer should learn no more from a speculative run than from the committed-only counterfactual: for every observer i, the inference advantage of $O_i(S)$ over $O_i(C)$ should be zero. This deliberately does not protect what the committed plan itself discloses. An agent that cancels an order must tell the provider so. It protects only the disclosure that speculation adds.

Trusted boundary and scope. The platform that runs the agent, local or remote, is trusted. The monitor sits between the planner and *every* externally observable IO path; retry queues, telemetry exporters, tracing spans, caches, rate-limit ledgers, and auth-token minting all sit behind it, so no raw speculative argument reaches them before a policy decision (Figure 3). Out of scope are the provider of a hosted planner or speculator, which sees the prompt and every candidate branch before the monitor runs (privacy from it needs a local planner), and any deployment that breaks complete mediation (A1, §4). The labeler runs inside the trusted boundary by requirement: a hosted labeler would itself observe the arguments the contract protects, on the critical path.

Does the decision itself leak? Suppressing or deferring a call changes call count, order and timing, so a policy can leak through its own decisions even when every argument is hidden. Making counts and timing independent of the secret, *obliviousness*, needs padding or cover traffic, which this runtime does not implement, so we scope the contract to the payload and destination channels and measure the decision channel (Appendix C.10): on the 150-task corpus a pattern-only adversary recovers no label under any policy. That is evidence for these policies, not a general guarantee, and decisions more strongly correlated with the secret would need padding.

4 Speculative Tool Privacy Contracts

Speculative Tool Privacy Contracts make the *observer-visible-before-commit* effect explicit and attach policy decisions to it.

Decision function. For a proposed event e under contract κ, the monitor computes $D(e, \kappa) \to \{\text{ALLOW}, \text{REWRITE}, \text{SHADOW}, \text{DEFER}, \text{BLOCK}\}$ as a fail-closed pipeline (any missing label or uncovered case maps to DEFER or BLOCK, never silently to ALLOW): BLOCK if the tool or destination is outside κ's allow-list; for a speculative event, DEFER if a configured disclosure budget or confidence threshold is not met (our Gate implementation BLOCKs instead; App. B), and SHADOW, REWRITE, or DEFER if its argument or destination label is in κ's protected set; otherwise audit and ALLOW. A committed event keeps its real arguments unless κ also sanitizes committed calls. The audit hook records the decision and transformed fields; raw speculative arguments never persist unless κ explicitly grants that observer. Pseudocode in Appendix B.

Actions. ALLOW dispatches the event unchanged. REWRITE applies a declared transformation (entity redaction, time-range coarsening, intent-phrase replacement) before dispatch. A rewritten speculative call also ends its branch, since the agent will not commit to its arguments; the rule is an opt-in runtime flag, enabled for every τ^2 Rewrite result we report except the default one in §5, Q2. SHADOW sends a protected speculative call to a trusted-local substitute, not the provider. SHADOW + REISSUE, our deployable policy, reissues promoted calls only after commitment. Plain SHADOW reuses the substitute on promotion; its 0.000 recovery on the generated corpus is a suppression reference, not a usable operating point. Neither suppresses unrelated committed calls (Table 8). DEFER queues the event for re-evaluation on branch promotion; abandoned branches terminate without external dispatch. BLOCK drops the event with a policy denial. Every event traverses four states with explicit guards (Issued $\to$ Decided $\to$ Dispatched $\to$ Completed); promotion of a previously DEFERRED event re-runs $D(e, \kappa)$ as a committed event before external dispatch.

Theorem 1 (Pre-commit soundness under complete mediation). Assume (A1) complete mediation: every speculative or committed event reaches the dispatcher only through the monitor, and no telemetry, retry queue, cache, or audit sink receives raw event fields before the monitor decides; (A2) label correctness: labels correctly classify each event's destination, arguments, and effect class; (A3) action preservation: rewrites and shadow substitutes remove the fields they are declared to remove (a Rewrite of an intent-revealing argument yields a generic one outside the protected set; a Shadow

substitute emits nothing externally). *Under* A1–A3, no field that κ forbids at observer i reaches i from a speculative event before commitment, whether the event is later committed, abandoned, deferred, or promoted. If in addition (A4) the run's committed calls match those of C, as when every committed call receives the result a non-speculative agent would, no forbidden field enters Δ_i; without A4 a substitute could steer the agent to a committed call C lacks. It is a specification rather than a deep result: it holds by construction under A1–A3, and violating any one admits a direct counterexample. Labels are a deployment input, like an access-control allow-list: the monitor acts correctly on the labels it receives, and *Labeler quality* (§5) prices imperfect ones. The guarantee is field-level; the inference-level goal and the decision channel are scoped in §3.2, and §5 tests that goal directly: before the first committed write, Shadow + reissue's issued trace supports no more inference than the committed-only trace on retail and airline. Appendix B gives the event schema, label predicates, action semantics, and proof.

5 EVALUATION

We answer Q1–Q3 of §1. The evidence comes from three generated corpora and a 65-task AgentDojo subset, and from two third-party agent benchmarks: τ^2-bench retail and airline (Barres et al., 2025), together with the banking domain of τ-Knowledge (Shi et al., 2026), which the same harness hosts, driven by the speculator released with Speculative Actions (Ye et al., 2026), and HotpotQA (Yang et al., 2018). Claude Opus 4.7 is the primary adversary on the generated corpora, banking, HotpotQA and the width re-scoring; the τ^2 retail rows use a deterministic schema classifier over tool identity, and Table 29 gives the Opus contrast there ($0.294 \to 0.018$). Per-corpus tables are in Appendix C.

Setup. The prototype is a Python contract dispatcher over five tool adapters (web search, document retrieval, calendar, CRM, email search) carrying effect, argument-privacy and destination-privacy labels, with live Brave Search and DuckDuckGo endpoints replacing the mock in the real-provider studies. Our own planner emits a median of 8 candidates per task on the generated corpora, 82–89% of them abandoned on average (Appendix C.24). On τ^2 the agent, user simulator and speculator are Claude Haiku-4.5, utility is the benchmark's own database-state grader, and speculation runs against a sandboxed clone of the database, so a promoted read is served from its speculative result and a promoted write re-executes at commit. The speculator's authors find three guesses per step more accurate than one on HotpotQA and pool several speculators' candidates on τ-bench (Ye et al., 2026), so widths above one are its intended regime. Naive is captured live at each width (114 tasks $\times$ 2 trials) and the allow/suppress policies and Rewrite are replayed over the captured frontiers, which matches live recovery and task success for them at one guess; policies that change the frontier or the agent's own calls run live on a 30-task slice (Appendix D).

Policies. The main text carries five (glossary: Table 5, Appendix B): the endpoints **Naive** and **No-Spec**; **SIA** (Hooper et al., 2026), the only directly attributable prior speculative-agent baseline; and the two contract actions that survive multiple-comparison correction, **Rewrite** and **Shadow**, the latter both in its suppress-only form and as **Shadow + reissue**, which re-issues a promoted call against the real provider once the agent commits. Nine further baselines share the same frontier (Appendix C): allow-listing (ACL), post-hoc scrubbing (Late-Scrub), redaction of PII and topic keywords from arguments before dispatch (Pre-Scrub), confidence gating (Gate, Table 19), refusal (the Drop policy, which DEFERs sensitive arguments and BLOCKs sensitive destinations), Rewrite + reissue, and three IFC variants including a FIDES-like Typed-IFC (Costa et al., 2025).

What we measure. Privacy is closed-set sensitive-attribute inference accuracy (*recovery*): the adversary sees one observer projection and picks from N task labels, against a $1/N$ random-guess floor. It is weaker than prompt inversion or re-identification (Zhang et al., 2024), so every recovery figure is a lower bound. It asks only whether an observer learns what the user is deliberating about. We measure issued-versus-committed inference advantage directly, with the committed plan fixed (Q1, *Mechanism*) and over each τ^2 run. *Ghost-only* recovery on abandoned calls isolates disclosure before commitment and of intents never committed; we report it beside full speculative recovery. Empty traces count as misses, so suppression can score below chance; tables report raw accuracy, and the chance-normalized $(r - 1/N)/(1 - 1/N)$ given for transformations leaves every ranking unchanged. Utility is task success, calls served from speculation, round trips left at commit, and wall-clock saving.

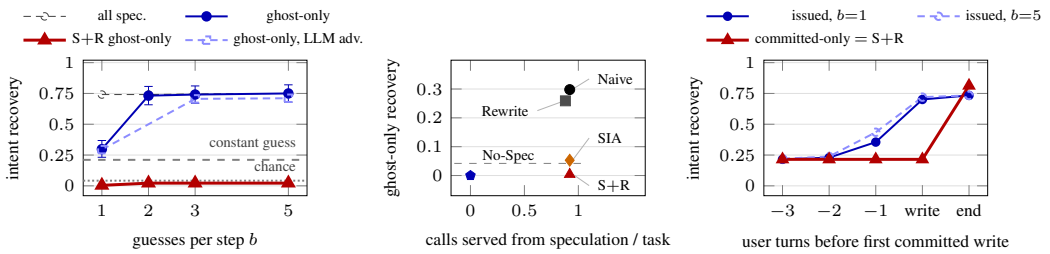

Figure 2: τ^2-bench retail, 114 tasks $\times$ 2 trials (Naive live, the others replayed; checked live at b=1); 457 ghost calls at b=1 and 4,387 at b=5 (per-task counts: Figure 8). **Left:** ghost-only recovery exceeds the constant guess at one guess and comes within a point of the whole speculative projection at two; the Opus 4.7 adversary tracks it. Dotted: chance; dashed: constant guess. **Middle:** the trade-off at b=1 (lower right is better); Late-Scrub coincides with Naive, Gate with No-Spec. **Right:** recovery from everything the provider has received, by user turn before the agent's first committed write (the 107 tasks that commit one). The committed-only trace, matched exactly by SIA and Shadow + reissue, supports only the constant guess until the write; Naive's issued trace reveals the intent a turn earlier, and by the end its ghost writes cost it accuracy.

Q1: How much does pre-commit disclosure leak? Speculation discloses what no committed trace does: the user's intent before they confirm it, intents they never had, and details sent to receivers outside the conversation.

What a ghost call contains, and who receives it. On one τ^2 retail task the speculator guesses the agent will escalate and issues `transfer_to_human_agents` with a summary saying a named customer "is experiencing financial hardship", with order numbers and refund totals; the agent never escalates, yet the escalation endpoint now holds a hardship report. The misattribution is systematic: 72 of the 229 intent-bearing ghost calls at one guess (31%) imply an intent the task never had, 18% at five. On banking, ghost knowledge-base searches go to a retrieval backend, and in the speculator's published retail runs 24 calls hand customers' IDs and emails to the handoff tool (§2).

Before the user confirms. Until the agent commits its first intent-bearing write, which τ^2 retail's policy lets it make only after the user confirms, the committed-only trace supports only the constant guess (0.211), which an Opus adversary reading the committed lookups does not improve on (App. D.5), while Naive's issued trace identifies the intent on 0.675 of the 114 retail tasks ($p = 1.1 \times 10^{-14}$; 0.70 on the 107 that commit a write, Figure 2, right), mostly in the turn before the write, and on 0.375 of the 24 runs that never commit one. SIA and Shadow + reissue add nothing on retail or airline, meeting §3.2's goal; once the write is committed it names the intent, and the issued trace is no more informative (App. D.5).

How much, and at what width. Benchmark, simulated users, grader and speculator are all third-party (Appendix D). On retail the speculator abandons 46% of its branches at one guess, leaving 2.0 provider-observed ghost calls per task that alone recover the intent on 0.298 of tasks, against 0.211 for always guessing the most common intent; half of them are intent-free reads, which is why one guess leaks least. In the speculator's intended regime, a second guess lifts ghost-only recovery to 0.732, and three and five reach 0.741 and 0.750, the level of the whole speculative projection (Figure 2; Holm $p \leq 2 \times 10^{-15}$); a task's ghost calls carry 1.5–1.6 of eight write intents against 1.3 in its gold set, so they disclose that task's intent rather than sweeping all of them. Our Opus 4.7 adversary, which reads arguments rather than tool identity, tracks the same curve, and a frequency-calibrated observer, which we adopted after the banking criterion failed and report as exploratory, gains 0.37 bits on retail and 0.67 on banking from Naive ghost traces and none from Shadow + reissue (App. D.3). The leak recurs on airline (0.481 of write-task intents, chance 0.100; App. D.1) and under a second, GPT-4o-mini prefetching speculator (0.404 on retail; App. D.4).

Mechanism. On 30 hand-crafted tasks with fixed committed plans, speculative recovery climbs from 0.44 under No-Spec to 0.97 under Naive, with Late-Scrub, SIA and ACL at Naive's level (Table 8); under a real LLM planner the gap (0.49 against 0.34) points the same way but is not resolved at 30 tasks (CIs overlap; Table 9). Exposure tracks the speculator's error rate, not the act of speculating: HotpotQA's speculator abandons only 4 of 123 calls over 15 questions, 3 provider-observed, so it has almost no ghost-only projection and our HotpotQA numbers describe the speculative one.

Policy	ghost-only [95% CI]	spec.	task success	served/task	commit/task
Naive	0.298 [0.232, 0.368]	0.741	0.575	0.92	6.88
No-Spec	0.000 [0.000, 0.000]	0.000	0.575	0.00	7.79
SIA	0.053 [0.018, 0.096]	0.053	0.575	0.92	6.88
Shadow + reissue	**0.004** [0.000, 0.013]	0.022	0.575	0.92	6.88

Table 1: One run set, four policies: τ^2-bench retail, 114 tasks $\times$ 2 trials at one guess per step, schema classifier (Opus 4.7: Table 29). Naive is captured live, the other three replayed over its frontiers. Recovery is against a 0.042 chance floor and a 0.211 constant-guess baseline; chance-normalized the ghost-only column reads $+0.268$, -0.043, $+0.011$ and -0.039. Task success, 0.575 [0.496, 0.654], is the Haiku-4.5 agent's own rate under τ^2's exact database-state grader, identical across rows because the agent receives the same tool results (§5; live check in Tables 27, 28). *Served/task*: promoted calls answered from speculation. *Commit/task*: provider round trips made at commit. Bold marks the best *deployable* policy here and in Tables 31 and 7; No-Spec is a floor, not a candidate.

Q2: Which policy reduces the leak most? Only issue-time transformation or suppression, and the action has to match the channel that carries intent.

Paired-frontier replay. Replaying one Naive frontier per task and replicate through every policy (150 tasks $\times$ 3; Table 10, Figure 5) varies only the dispatch decision. Of eleven contrasts with Naive under Holm correction (Holm, 1979), only **Rewrite** (-0.076) and **Shadow** (-0.104), which act on every speculative call, survive (Holm $p \leq 0.015$); SIA, the scrubbers, ACL and both taint baselines do not, and a FIDES-style Typed-IFC (Costa et al., 2025) falls inside Naive's interval.

Suppression must still serve the committed plan. Plain Shadow's 0.000 is suppression: its substitute also serves the committed call, so every Shadow trace on the generated corpus is empty. Shadow + reissue is the deployable form: on τ^2 it takes ghost-only recovery from 0.298 to 0.004 at one guess and holds it at 0.022 at every larger width (Holm $p = 6 \times 10^{-15}$), and the Opus adversary agrees (Holm $p \leq 6.5 \times 10^{-14}$). Refusing to speculate the protected calls, a read-only rule extended with argument labels, is the Drop action, and on retail, airline and banking it leaves the provider exactly the trace Shadow + reissue does in replay (Tables 27, 30, 31). That is structural: either action keeps a protected call off the wire until commitment and sends it then if the agent needs it, and on retail and airline most protected calls are writes (75% and 79%), which re-execute at commit under every policy. Withholding actions are interchangeable for the observer.

Transforming actions are not. By default Rewrite lets the speculator keep building on each rewritten lookup, and those branches are abandoned once the real call commits: on the live 30-task slice at one guess, default Rewrite more than doubles the ghost calls (4.5 against 2.0 per task) and raises ghost-only recovery from Naive's 0.300 to 0.533 ($p = 0.016$; Table 28). Ending the branch at the rewritten call (§4) instead lowers recovery, from 0.298 to 0.259 at one guess and from 0.73–0.75 to 0.59–0.61 beyond ($p \leq 0.039$; App. D). Which withholding action is chosen does not matter; a transforming action must also end the branch it rewrites, or it exposes more than no protection.

The action must match the channel. On τ^2 the arguments Rewrite transforms do not carry the intent; on HotpotQA, where intent rides on the query text, it takes the Opus adversary from 1.00 to 0.13 over 15 questions, on the speculative projection since that speculator abandons almost nothing. The same holds at a real provider: against the live DuckDuckGo API on the 30-task corpus, Rewrite takes recovery to the chance floor while Naive stays an order of magnitude above it (Table 17). The mirror image is SIA, indistinguishable from Naive on the synthetic corpora but down at 0.053 on τ^2, where intent rides on *which* retail write tool is called and refusing to speculate writes happens to suppress the carrying channel. Move the intent into a read and SIA no longer closes the channel: on banking, where the speculator's knowledge-base queries carry it, SIA leaves 0.247 against Naive's 0.335 and Shadow + reissue's 0.036 ($p = 3.7 \times 10^{-9}$ against Shadow + reissue). For the frequency-calibrated observer of App. D.3, which we report as exploratory, SIA's residual gain is 0.125 bits against Naive's 0.67. The second speculator's knowledge-base queries carry no intent, so there SIA and Shadow + reissue coincide; on its banking write tasks Shadow + reissue still removes the leak ($p \leq 1.5 \times 10^{-4}$), and its all-task score reflects benign account lookups rather than disclosure (App. D.4). A contract protects only the channel it transforms (Proposition 2).

Q3: What privacy–utility trade-off results? A middle ground: on retail, Shadow + reissue keeps speculative reuse and task success at 0.4% recovery; each withheld read costs a round trip.

Speculative reuse and timing. Because τ^2's in-process tools and cached model responses make wall-clock timing uninformative, Table 1 reports round trips: Shadow + reissue and Naive each serve 0.92 promoted calls per task from speculation; No-Spec serves none. On a separate multi-tool workload with live DuckDuckGo search, Shadow + reissue sends as many provider calls as No-Spec (4.25 and 4.24 per task, Naive 12.72; Appendix C.20). It keeps a 7.7% p50 saving against sequential No-Spec, compared with 12.1% for unprotected speculation; compared directly on the same runs, protection costs a median 6.8 points of the sequential baseline (Appendix C.20). That cost is the withheld-read round trip described below. It is also no more attributable than the saving it is subtracted from: Taint-F, issuing 0.14 provider calls per task, saves as much as Naive issuing 12.69 (both over all timed rows). Against a baseline that dispatches independent committed calls in parallel, a sweep of tool latency from 100 ms to 2.5 s on this workload puts speculation's saving at 7–18% (mean 13%), at 2.8–3.0× the provider calls (`results/latency-sweep/summary.csv`). On τ^2, 71% of tool-calling turns issue a single call whose result later turns consume, so committed parallelism has nothing to overlap. Speculation answers 0.41 such turns per task outright.

Task utility. Since promoted writes re-execute at commit, only a promoted read could return a different value. No-Spec re-executes the promoted reads at commit, and in the 228 runs each returns the same value, since the six that follow a committed write are catalogue lookups no retail write modifies. On the live 30-task slice every policy except ACL and Pre-Scrub scores 0.600 (12 of 14; Table 27), agreeing with replay on every task. Blunter contracts do break the task: our access-control baseline refuses the agent's own committed calls and scores 0.000, a limit of that configuration rather than of access control. Issue-time PII redaction scores 0.200.

When is withholding free? Shadow + reissue keeps the latency gain only for promoted reads the labeler passes as benign: writes re-execute at commit under every policy, and each withheld read costs one round trip. Where intent is carried by which tool is called, as on retail, that share is 1.00: all 209 reusable calls pass the issue-time labels. Where it is carried by the argument of a read it falls. On banking 38 of 59 pass, so Shadow + reissue serves 0.20 of Naive's 0.30 calls per task; on the search and retrieval corpora about half pass under both the CPU labeler and the gold annotator (Table 23). The labeler's operating point sets that share (Figure 7, Table 25).

Labeler quality. Theorem 1 assumes correct labels (A2). Retail uses a local schema labeler, not an oracle; it protects 74.9% of 1,013 speculative calls. On generated tasks, increasing protect recall from 0.66 to 0.89 cuts excess leakage over No-Spec from $+0.098$ to $+0.004$, while speculative reuse falls from 0.68 to 0.49 (Figure 7). On 80 training-disjoint tasks, a 22M-parameter CPU probe labels a frontier in 28–56 ms, cutting speculative recovery from 0.733 to 0.450 against a 0.338 no-speculation floor, 72% of the gap, and serving 55.5% of promoted calls (App. C.23).

Ablations and robustness. On generated corpora, masking tool names or anonymizing destinations shifts recovery by at most 0.03: arguments carry the signal there, tool choice on τ^2 retail (App. C.9). At 20% label false negatives, Rewrite (0.06) remains below noisy Naive ($p = 0.007$); Shadow (0.08) is lower but not significantly ($p = 0.087$). The ranking holds across three LLM adversaries, a 7B model, keyword and TF-IDF baselines, alternative metrics, and AgentDojo; at a real HTTP boundary only the direction is resolved.

6 Conclusion

Speculative tool execution creates an issue-time privacy failure: a not-yet-needed call discloses intent before commitment, and only issue-time controls matched to the disclosure channel that carries it suppress the leak. How much speculation buys depends on how much tool time the planner can overlap; how much it exposes does not. Dispatch is therefore the last boundary at which the runtime can still prevent the disclosure.

AI use statement

We used generative AI in the system under study and as a writing aid, so we separate the two uses. *As method:* the benchmark corpora were generated with an LLM (§C.2); the planners, the agents, the adversaries, and the gold annotator used for labeler validation are all LLMs, with models and prompts specified in the appendix and released in the artifact. Every such component is evaluated rather than

trusted: the labeler is validated against a gold annotator (Tab. 22), and the adversary is cross-checked against non-LLM baselines (Tab. 15) and against a schema classifier on τ^2 (Tab. 29). We also tested an LLM-as-judge rubric for task success, found that it could not separate an informative answer from a content-free one, and therefore report utility only from τ^2's own state-based grader. We have reviewed all AI-assisted work and take responsibility for the final content of this paper, including all text, claims, and artifacts.

Ethics statement

This work studies a privacy leak, so it necessarily describes how an observer can infer a user's sensitive intent from a tool-call trace. We judged the disclosure net-positive: the channel arises from an optimization that deployed systems already ship, the inference uses only off-the-shelf models against traces a provider already holds, and the defenses we evaluate are implementable today. No new attack capability is created by publishing it.

The generated corpora are synthetic: their sensitive labels (health conditions, employment jeopardy, financial distress) are LLM-generated scenarios attached to LLM-generated user turns. The third-party benchmarks are public: τ^2-bench retail and airline ship synthetic customer databases with simulated users, τ-Knowledge banking adds a synthetic knowledge base, AgentDojo ships synthetic suites, and HotpotQA is a public question-answering dataset queried against a local corpus. No real user data, no human subjects, and no personal information from any real individual appear in any corpus or trace. The real-provider experiments issue synthetic queries to public search endpoints. We release the corpora, traces, and code so the measurements can be checked.

Reproducibility statement

The runtime, every policy, the labelers, the adversary prompts, and every frozen frontier and trace are released, with each model request recorded against the model identifier it was issued under; the Haiku and GPT-4o-mini adversaries are pinned to dated snapshots, while the Opus planner and adversary are recorded under a floating alias. Every number in this paper is generated from those records by the released scripts. Limitations are discussed in Appendix A.

References

Mahsa Amani, Seungeon Lee, Abhisek Dash, Asmaa El Fraihi, Yunah Jang, Elisabeth Kirsten, Qinyuan Wu, Krishna P. Gummadi, Manish Gupta, Abhilasha Ravichander, Muhammad Bilal Zafar, and Soumi Das. Characterizing web search by conversational LLM agents: From search decisions and strategies to results and responses, September 2026. URL https://arxiv.org/abs/2609.19244.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment. arXiv:2506.07982, June 2025. URL https://arxiv.org/abs/2506.07982.

Yuandao Cai, Wensheng Tang, Cheng Wen, and Shengchao Qin. Ghost in the agent: Redefining information flow tracking for LLM agents, April 2026. URL https://arxiv.org/abs/2604.23374.

Mohammad Hossein Chinaei. Causality laundering: Denial-feedback leakage in tool-calling LLM agents, April 2026. URL https://arxiv.org/abs/2604.04035.

Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM*, 45(6):965–981, 1998. doi: 10.1145/293347.293350.

Cloudflare. The 2025 Cloudflare Radar year in review. https://blog.cloudflare.com/radar-2025-year-in-review/, December 2025. Accessed 2026-09-25.

Manuel Costa, Boris Köpf, Aashish Kolluri, Andrew Paverd, Mark Russinovich, Ahmed Salem, Shruti Tople, Lukas Wutschitz, and Santiago Zanella-Béguelin. Securing AI agents with information-flow control, May 2025. URL https://arxiv.org/abs/2505.23643.

Edoardo Debenedetti, Jie Zhang, Mislav Balunovic, Luca Beurer-Kellner, Marc Fischer, and Florian Tramèr. AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. In *Advances in Neural Information Processing Systems*, volume 37, pp. 82895–82920. Curran Associates, Inc., 2024. doi: 10.52202/079017-2636.

Deepgram. Optimize voice agent latency with eager end of turn. Deepgram Flux documentation, 2025. URL https://developers.deepgram.com/docs/flux/voice-agent-eager-eot. Eager end-of-turn triggers speculative LLM calls at a documented cost of 50–70% more calls. Accessed 2026-09-20.

Dorothy E. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19 (5):236–243, 1976. doi: 10.1145/360051.360056.

Ben Dong, Zhonghao Guo, Tianyi Lu, and Qian Wang. The claws in plain sight: Unauthorized context disclosure through LLM agent tool calls, August 2026. URL https://arxiv.org/abs/2608.20658.

Aarya Doshi, Yining Hong, Congying Xu, Eunsuk Kang, Alexandros Kapravelos, and Christian Kästner. Towards verifiably safe tool use for LLM agents. In *Proceedings of the 48th IEEE/ACM International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER '26)*, pp. 201–205, Rio de Janeiro, Brazil, April 2026. Association for Computing Machinery. URL https://conf.researchr.org/details/icse-2026/icse-2026-nier/41/Towards-Verifiably-Safe-Tool-Use-for-LLM-Agents.

Faouzi El Yagoubi, Godwin Badu-Marfo, and Ranwa Al Mallah. AgentLeak: A benchmark for internal-channel privacy leakage in multi-agent LLM systems. *IEEE Access*, 14:94960–94978, 2026. doi: 10.1109/ACCESS.2026.3704541.

Guangyu Feng, Huanzhi Mao, Prabal Dutta, and Joseph E. Gonzalez. Concurrency without model changes: Future-based asynchronous function calling for LLMs, May 2026. URL https://arxiv.org/abs/2605.15077.

GitHub. Reviewing the audit log for your organization. GitHub Docs, https://docs.github.com/en/organizations/keeping-your-organization-secure/managing-security-settings-for-your-organization/reviewing-the-audit-log-for-your-organization, 2026. Events from the last 180 days. Accessed 2026-09-25.

Glama. The glama MCP gateway. Product documentation, https://glama.ai/mcp/gateway, 2026. Accessed 2026-09-25.

Google. Data retention and lag times. Google Workspace Admin Help, https://support.google.com/a/answer/7061566, 2026. Accessed 2026-09-25.

Yilin Guan, Qingfeng Lan, Fei Sun, Dujian Ding, Devang Acharya, Chi Wang, William Yang Wang, and Wenyue Hua. Dynamic speculative agent planning. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR 2026)*. OpenReview.net, 2026. URL https://openreview.net/forum?id=YZ5k2dVj6O.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Coleman Hooper, Minwoo Kang, Suhong Moon, Nicholas Lee, Eric Wen, John Wawrzynek, Michael W. Mahoney, Yakun Sophia Shao, Amir Gholami, and Kurt Keutzer. Speculative interaction agents: Building real-time agents with asynchronous I/O and speculative tool calling, May 2026. URL https://arxiv.org/abs/2605.13360.

Daniel C. Howe and Helen Nissenbaum. Trackmenot: Resisting surveillance in web search. In Ian Kerr, Valerie Steeves, and Carole Lucock (eds.), *Lessons from the Identity Trail: Anonymity, Privacy and Identity in a Networked Society*, pp. 417–436. Oxford University Press, 2008. doi: 10.1093/oso/9780195372472.003.0023.

Shijing Hu, Liang Liu, Zhu Meng, and Zhicheng Zhao. ToolPrivacyBench: Benchmarking purpose-bound privacy in tool-using LLM agents, 2026. URL https://arxiv.org/abs/2606.28061.

Wenyue Hua, Mengting Wan, Jagannath Shashank Subramanya Sai Vadrevu, Ryan Nadel, Yongfeng Zhang, and Chi Wang. Interactive speculative planning: Enhance agent efficiency through co-design of system and user interface. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025)*. OpenReview.net, 2025. URL https://openreview.net/forum?id=BwR8t91yqh.

Tao Huang, Chen Hou, Guosen Wu, and Jiayang Meng. Observable channels, not just storage: Evaluating privacy leakage in LLM agent pipelines, March 2026. URL https://arxiv.org/abs/2603.22751.

incident.io. Break chatbot speed limits with speculative tool calls. incident.io engineering blog, 2025. URL https://incident.io/building-with-ai/speculative-tool-calling. Issues tool calls speculatively from keywords in the user's message; reports 2–3 s saved per interaction. Accessed 2026-09-20.

Paul Kocher, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, Moritz Lipp, Stefan Mangard, Thomas Prescher, Michael Schwarz, and Yuval Yarom. Spectre attacks: Exploiting speculative execution. In *Proceedings of the 2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1–19. IEEE, 2019. doi: 10.1109/SP.2019.00002.

LiveKit. Preemptive generation. LiveKit Agents documentation, 2026. URL https://docs.livekit.io/agents/logic/turns/tuning/. Speculatively starts LLM inference on a stable partial transcript before the user's turn is confirmed and discards the generation if the final transcript differs; enabled by default. Accessed 2026-09-20.

Mintlify. The state of agent traffic in documentation. https://www.mintlify.com/blog/state-of-ai, March 2026. Accessed 2026-09-25.

Bardia Mohammadi, Nearchos Potamitis, Lars Henning Klein, Akhil Arora, and Laurent Bindschaedler. Atomix: Timely, transactional tool use for reliable agentic workflows. In *Proceedings of the ICLR 2026 Workshop on Agents in the Wild: Safety, Security, and Beyond (AIWILD)*. OpenReview.net, 2026. URL https://openreview.net/forum?id=UeRbEpSVUz.

Yutao Mou, Zhangchi Xue, Lijun Li, Peiyang Liu, Shikun Zhang, Wei Ye, and Jing Shao. ToolSafe: Enhancing tool invocation safety of LLM-based agents via proactive step-level guardrail and feedback. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 37125–37153, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1850. URL https://aclanthology.org/2026.findings-acl.1850/.

Andrew C. Myers. JFlow: Practical mostly-static information flow control. In *Proceedings of the 26th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '99)*, pp. 228–241. Association for Computing Machinery, 1999. doi: 10.1145/292540.292561.

Daniel Nichols, Prajwal Singhania, Charles Jekel, Abhinav Bhatele, and Harshitha Menon. Optimizing agentic language model inference via speculative tool calls, December 2025. URL https://arxiv.org/abs/2512.15834.

Helen Nissenbaum. Privacy as Contextual Integrity. *Washington Law Review*, 79(1):119–158, 2004.

Yuxuan Qiao, Dongqin Liu, Hongchang Yang, Wei Zhou, and Songlin Hu. Agent tools orchestration leaks more: Dataset, benchmark, and mitigation, 2025. URL https://arxiv.org/abs/2512.16310. To appear, Findings of the Association for Computational Linguistics: EMNLP 2026.

Yangjun Ruan, Honghua Dong, Andrew Wang, Silviu Pitis, Yongchao Zhou, Jimmy Ba, Yann Dubois, Chris J. Maddison, and Tatsunori Hashimoto. Identifying the risks of LM agents with an LM-emulated sandbox. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR 2024)*. OpenReview.net, 2024. URL https://openreview.net/forum?id=GEcwtMk1uA.

Quan Shi, Alexandra Zytek, Pedram Razavi, Karthik Narasimhan, and Victor Barres. τ-knowledge: Evaluating conversational agents over unstructured knowledge. arXiv:2603.04370, March 2026. URL https://arxiv.org/abs/2603.04370.

Minsun Shim, Ramisha Raida Karim, Ruthwik Jakkula, Kaiwen Zhou, Xin Liu, Xin Eric Wang, and Zhou Li. Confuse the model, control the flow: Understanding and mitigating privacy leakage from LLM agents with information flow control, September 2026. URL https://arxiv.org/abs/2609.14003.

Stream. How do you handle "speculative tool calling" in a voice loop to prevent the 3-second silence while the LLM decides which function to use? https://getstream.io/blog/speculative-tool-calling-voice/, December 2025. Accessed 2026-09-25.

Yifan Sui, Han Zhao, Rui Ma, Zhiyuan He, Hao Wang, Jianxun Li, Kaiqiang Xu, Kai Chen, and Yuqing Yang. Parallelizing tool execution and LLM generation for low-latency agent serving, March 2026. URL https://arxiv.org/abs/2603.18897.

The Chromium Projects. Omnibox prefetch and prerender for default search engines. Chromium design document, https://www.chromium.org/developers/design-documents/omnibox-prefetch-for-default-search-engines/, 2024. Accessed 2026-09-25.

Jiankun Wei, Abdulrahman Abdulrazzag, Tianchen Zhang, Adel Müürsepp, and Gururaj Saileshwar. When speculation spills secrets: Side channels via speculative decoding in LLMs, November 2024. URL https://arxiv.org/abs/2411.01076.

Heming Xia, Yongqi Li, Cunxiao Du, Mingbo Song, and Wenjie Li. ToolSpec: Accelerating tool calling via schema-aware and retrieval-augmented speculative decoding, April 2026. URL https://arxiv.org/abs/2604.13519.

Jin Xie and Songze Li. OCELOT: Inference-leakage budgets for privacy-preserving LLM agents, 2026. URL https://arxiv.org/abs/2606.12341.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik R. Narasimhan. τ-bench: A benchmark for Tool-Agent-User interaction in real-world domains. In *Proceedings of the Thirteenth International Conference on Learning Representations (ICLR 2025)*. OpenReview.net, 2025. URL https://openreview.net/forum?id=roNSXZpUDN.

Naimeng Ye, Arnav Ahuja, Georgios Liargkovas, Yunan Lu, Kostis Kaffes, and Tianyi Peng. Speculative actions: A lossless framework for faster AI agents. In *Proceedings of the Fourteenth International Conference on Learning Representations (ICLR 2026)*. OpenReview.net, 2026. URL https://openreview.net/forum?id=P0GOk5wslg.

Shenglai Zeng, Jiankun Zhang, Pengfei He, Yiding Liu, Yue Xing, Han Xu, Jie Ren, Yi Chang, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG). In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4505–4524, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.267.

Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 10471–10506, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.624.

Collin Zhang, John Xavier Morris, and Vitaly Shmatikov. Extracting prompts by inverting LLM outputs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024. URL https://aclanthology.org/2024.emnlp-main.819/.

Yixiang Zhang, Xinhao Deng, Zhongyi Gu, Yihao Chen, Ke Xu, Qi Li, and Jianping Wu. Exposing LLM user privacy via traffic fingerprint analysis: A study of privacy risks in LLM agent interactions, October 2025. URL https://arxiv.org/abs/2510.07176.

Arman Zharmagambetov, Chuan Guo, Ivan Evtimov, Maya Pavlova, Ruslan Salakhutdinov, and Kamalika Chaudhuri. AgentDAM: Privacy leakage evaluation for autonomous web agents. In *Advances in Neural Information Processing Systems (Datasets and Benchmarks Track)*, 2025. URL https://arxiv.org/abs/2503.09780.

Peter Yong Zhong, Siyuan Chen, Ruiqi Wang, McKenna McCall, Ben L. Titzer, Heather Miller, and Phillip B. Gibbons. RTBAS: Defending LLM agents against prompt injection and privacy leakage, February 2025. URL https://arxiv.org/abs/2502.08966.

Observer	Typical visibility
Provider	Tool, endpoint, tenant, arguments, timestamps, sizes, provider logs.
Network	Destination, timing, sizes, retries, traffic shape (TLS hides plaintext args).
Runtime log	Full trace including branch labels, transformed arguments, cache keys.
Co-tenant	Cache occupancy, lock contention, queueing delay, rate-limit residue.
Auditor	Whatever the runtime records.

Table 2: Observer projections used in the trace model. Each row lists fields that observer class can recover.

A LIMITATIONS

Labels are a deployment input. Theorem 1 assumes correct labels (A2), and a labeler is schema-specific: the rule labeler is written from each tool schema, and the trained probe dropped into τ^2 without retraining protects only 0.8% of calls (Appendix C.23). A deployment therefore supplies labels for its own tools, as it supplies an access-control policy. We price imperfect labels end to end (Figure 7) rather than claim a universal labeler.

What the theorem covers. A1–A3 establish that forbidden fields do not leave before commitment. The claim about Δ_i also needs committed-plan equivalence (A4), which replay over a frozen frontier supplies and our live τ^2 slice confirms, but a general adaptive agent need not satisfy. Neither claim bounds inference from decision patterns, which we measure instead on the 150-task corpus (Appendix C.10). Contracts are a policy abstraction implemented in a prototype runtime: complete mediation across egress, retries, telemetry, and caches (Figure 3) is the deployment's to provide, not a hardened mechanism we ship.

What the measurements cover. The main τ^2 evidence uses the speculator released with Speculative Actions and a Haiku-4.5 agent. A second, prefetching speculator reproduces the retail result and banking write-task protection but fails the prespecified all-task banking test. The failure comes from the metric, not a disclosure: benign account lookups that Shadow + reissue lets through make the adversary's default "no write" answer a hit on no-write tasks, and a frequency-calibrated observer, a metric we adopted after seeing the failure, detects no gain from those traces (Appendix D.4). SIA's banking residual does not replicate with a speculator whose knowledge-base queries carry no intent. The size of the ghost-only projection tracks each speculator's error rate and width, so other speculators leak more or less. On τ^2 the committed writes name most intents by the end of a run, so the issued trace's advantage there is one of timing and of intents never committed (Appendix D.5); we measure it with the schema classifier. An Opus adversary that reads the committed lookups gains nothing from them, leaving the advantage at $+0.482$; a naive Bayes fit to the same deployment's traffic raises the baseline to 0.368 and halves it (Appendix D.5). The primary evaluation, every τ^2 result included, is closed-set attribute inference over each benchmark's label menu. The open-set probes are smaller and confined to the generated corpora (30 tasks in Table 14, 150 in Appendix C.17): they score a free-form guess by lexical or semantic match, bound open-ended recovery on those corpora only, and are not run on τ^2. The real-provider studies (DuckDuckGo, 30 tasks; Brave, 9) and HotpotQA (15 questions) confirm the direction of the effect at a live HTTP boundary and where intent rides on query text; they are narrower than τ^2 and not powered to order policies. Wall-clock latency is measured on three workloads (Tables 16, 18 and 21), and τ^2 reports round trips instead. We establish what speculation costs when it happens, not how common tool-boundary speculation is in deployment.

B CONTRACT MECHANISM DETAILS

This appendix gives the event schema, label predicates, action semantics, decision-function pseudocode, prototype implementation notes, a worked contract example, and a worked example of the trace model on the motivating vignette of §1.

Event schema. Each tool event is a tuple $e = (\mathit{tool}, \mathit{args}, \mathit{destination}, \mathit{mode}, t)$, where *tool* identifies the adapter, *args* is a dictionary of named argument fields, *destination* is the externally observable endpoint or collection, *mode* ranges over the six execution modes of Table 3, and t is the issue timestamp. The measured prototype assigns one argument-projection label to the whole dictionary

Field	Meaning
Execution mode	committed, speculative, prefetch, retry, validation, shadow.
Effect class	pure-local, local-read, external-read, external-write, semi-stateful, irreversible.
Observer class	provider, network, runtime log, co-tenant, auditor.
Argument label	One nominal class for the argument dictionary in the prototype: public, pseudonymous, personal, relational, sensitive, or intent-revealing.
Destination label	One class from the same vocabulary for the endpoint or collection.
Branch confidence / budget	Planner-stated commit score (not assumed calibrated); optional per-task disclosure budget.
Audit policy	which observer (if any) receives the audit record of this event.

Table 3: Fields the monitor binds to each speculative event before dispatch.

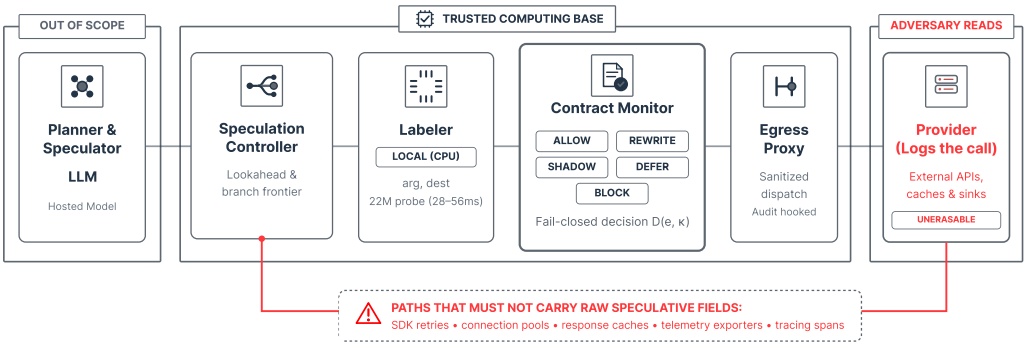

Figure 3: Where a contract acts. The hosted planner and speculator model is out of scope (§3.2). Every candidate branch passes the speculation controller, the local labeler (the trained probe shown; rule-based on τ^2), and the contract monitor, which decides allow, rewrite, shadow, defer, or block *before* the call reaches the egress proxy, optionally re-issuing it once the agent commits. That decision is the only thing that changes between the policies we compare, and the adversary reads whatever the provider logs, which cannot be erased. Red: the paths that would carry raw speculative fields around the monitor (SDK retries, connection pools, response caches, telemetry exporters, tracing spans) and break complete mediation.

and one label to the destination, from the nominal vocabulary in Table 3; the abstract contract also admits finer per-field labels when an implementation supplies them.

Why a multi-field vocabulary and not a scalar risk level? Collapsing the vocabulary to a single (mode, risk-level) pair would force every policy to encode its decision inside a policy-specific risk function. That relocates the complexity. Different policies read different fields: Rewrite reads argument sensitivity, Shadow reads destination and argument sensitivity, Gate reads branch confidence, Drop reads sensitivity anywhere. Sensitivity and confidence are also orthogonal in our measurements (the Gate lever in Table 19 lives in the high-sensitivity, low-confidence quadrant a scalar would flatten), and one destination yields multiple observer projections, so policies must track Δ_i per observer.

Label predicates. The six classes in Table 3 are nominal, not a claimed total order: a pseudonymous identifier and a relational fact need not be comparable. The abstract contract κ can specify forbidden argument and destination label sets per observer; the evaluated profile uses one provider-observer set. Shadow + reissue protects argument labels SENSITIVE and INTENT-REVEALING and destination label SENSITIVE; its allow decision is exactly a set-membership test. On τ^2 a deterministic rule-based labeler assigns these aggregate labels from $(tool, args, destination)$ before the policy acts. On the generated corpora each adapter emits a constant argument label (INTENT-REVEALING; Appendix C.23), so there Rewrite, Shadow and Drop act on every speculative call; selective protection on those corpora is evaluated with the local probe (Table 23). Theorem 1 applies to the measured whole-argument projection only when A2 makes that aggregate label correct for every field the contract forbids; finer per-field labeling is an interface extension, not a measurement claimed here.

Branch confidence. The planner emits a per-branch confidence $c \in [0, 1]$ alongside each speculative candidate, prompted to estimate its commit probability. Gate consumes this score directly: a candidate with $c < \kappa.\tau$ is suppressed. Our implementation BLOCKs it rather than DEFERring it, so a gated candidate is dropped instead of being reconsidered at promotion; that makes the sweep in Table 19 a lower bound on what a deferring gate would serve. Any score in $[0, 1]$ is admissible as a ranking input, but τ has no promotion-probability interpretation without calibration and no protected-ghost-budget interpretation without a separate risk estimate. Pooled over candidates in the 30- and 150-task captured frontiers, mean stated confidence is 0.385 and 0.348 against pooled realized promotion rates of 0.189 and 0.121 (Table 26 gives the mean per-task rates, 17.9% and 11.4%). At $\tau = 0.7$, only 52 of 748 and 126 of 3,578 candidates pass, with realized ghost rates 0.058 and 0.159 among those admitted. These descriptive rates are not guarantees; the separate isotonic recalibration check splits by task across all replicates (`scripts/audit_gate_calibration.py`). Gate is a baseline, not a certified policy.

Action semantics. ALLOW(e) dispatches e to its destination unchanged. REWRITE(e) applies the declared argument transformation and dispatches the result; Theorem 1's A3 requires that transformed fields actually leave the contract's forbidden set. SHADOW(e) routes e to a trusted-local substitute and marks the result non-provider-visible; the substitute is a function of $(\textit{tool}, \textit{args})$ and never touches an external observer. DEFER(e) buffers e; if the speculative branch is promoted to the committed plan, the monitor re-runs $D(e, \kappa)$ as a committed event and may dispatch then. BLOCK(e) drops e with a policy denial and records an audit entry per κ.audit_policy.

Disclosure budget. A contract κ may impose a per-task budget b on the number of SENSITIVE-argument or INTENT-REVEALING-argument dispatches. The monitor increments a counter on every ALLOW or REWRITE; once the counter reaches b, subsequent qualifying candidates fall through to DEFER. Budgets are reset at task boundaries.

Worked contract example. For a web-search tool, an illustrative contract might specify: *allowed_tools* = $\{$*web_search*$\}$; *allowed_destinations* = $\{$*api.brave.com*, *api.duckduckgo.com*$\}$; protected argument labels = $\{$SENSITIVE, INTENT-REVEALING$\}$; $\tau = 0.7$ for speculative dispatch; and audit to runtime log. Under this contract, a speculative *web_search*(*q*="tenant rights lease termination penalty"), with its argument dictionary labeled INTENT-REVEALING, triggers REWRITE to a generic query before dispatch. A committed call for the same query would also REWRITE under this illustrative profile; the measured Rewrite + reissue profile instead releases the real query only at commitment.

Decision-function pseudocode. The body summarizes the monitor's logic in §4; the full fail-closed pipeline is:

```
function decide(event e, contract k):
  l = labels(e)   # aggregate in measured prototype
  if l is missing:
    return DEFER if e.is_speculative else BLOCK
  # endpoint authorization is not a privacy label
  if e.tool not in k.allowed_tools:
    return BLOCK
  if e.destination not in
       k.allowed_destinations:
    return BLOCK

  protected_arg = l.argument in k.protected_args
  protected_dest = l.destination in k.protected_dests
  if e.is_speculative:
    if k.has_budget and budget_exceeded(k, e):
      return DEFER
    if k.has_threshold and e.conf < k.threshold:
      return DEFER
    if protected_arg or protected_dest:
      if k.protected_action == SHADOW:
        return SHADOW(k.substitute_for(e))
```

```
      if k.protected_action == REWRITE
           and protected_arg and not protected_dest:
        return REWRITE(scrub(e.args, k))
      return DEFER
    audit(e, k, ALLOW)
    return ALLOW

  # committed-policy choice: ordinary reissue or
  # the plain Rewrite variant's sanitization
  if k.sanitize_committed and protected_arg:
      return REWRITE(scrub(e.args, k))
  audit(e, k, ALLOW)
  return ALLOW
```

This is the contract-level decision rule, not a claim that every optional knob was active in every experiment. The concrete endpoint allow-list and the destination's privacy class are separate checks. The measured prototype labels the argument dictionary as one projection and tests exact membership in the protected sets above; it does not implement a total-order comparison or a per-field lattice. Budget and confidence checks, when configured, occur only before speculative dispatch. The measured Shadow + reissue profile chooses SHADOW for a protected speculative argument or destination and releases the real call at commitment; plain REWRITE sanitizes committed arguments too, whereas Rewrite + reissue does not. A deferred call or a Shadow + reissue promotion re-runs the policy as a committed event; plain Shadow uses its substitute instead.

Proof of Theorem 1. Consider each speculative event before its branch resolves. Before dispatch it emits nothing. Under A1 every externally observed field passes through the monitor, including fields that could otherwise escape through a retry, cache, or telemetry path. The five decisions exhaust the cases: BLOCK and DEFER emit no tool-call field; SHADOW emits only to a trusted local substitute under A3; REWRITE emits the transformed fields, which A3 says do not carry a forbidden label; and ALLOW is reachable only when the argument and destination labels pass the contract's protected-set tests, which A2 says are correct. Thus induction over any sequence of speculative decisions yields no forbidden field in the pre-commit external projection. At promotion, a deferred event or one under Shadow + reissue is re-evaluated as a committed call; plain Shadow instead reuses its local substitute. A committed call may issue its real arguments, so the first claim deliberately stops at commitment. For the corollary about Δ_i, A4 pairs each such committed call with a field-identical event in C; those fields cancel under π, while the remaining speculative events have no forbidden field by the first claim. A4 is essential: a substitute that changes the agent's eventual tool choice can satisfy A1–A3 yet produce a committed field absent from C. Counts and timing need not cancel even under A4. □

Prototype implementation. The prototype is a Python runtime on a tool-adapter abstraction with a contract-policy hook on the dispatch path. The static-plan harness exposes five mock tools (web search, document retrieval, calendar lookup, CRM lookup, email search); each tool carries a contract with effect, argument-privacy, and destination-privacy labels. For the real-provider experiment we swap in a Brave Search adapter in place of the mock web-search tool. The real-planner path uses an LLM planner that emits speculation candidates and a committed plan; both flow through the same contract-action dispatcher; where a policy consumes branch confidence (Gate), its default threshold is τ=0.7. Two observer helpers record (i) tool name, a bounded argument summary, dispatcher decision, and 100 ms duration bucket; and (ii) timestamps with synthetic cache-hit flags over canonicalized tool-name and argument keys. The static experiments use simulated harness latencies (web search 200 ms, document retrieval 100 ms, calendar 50 ms, CRM 80 ms, email 120 ms).

B.1 WORKED TRACE EXAMPLE

We instantiate the trace model on the motivating example from §1. A user query "What's the penalty for breaking my lease early?" induces a speculative frontier of three tool calls, of which the agent later commits to one.

Issued trace S.

1. $e_1 = (\textit{web search}, q_1 =$ "tenant rights lease termination penalty", external, speculative, $t = 0.1\,\text{s})$
2. $e_2 = (\textit{web search}, q_2 =$ "available apartments near me", external, speculative, $t = 0.15\,\text{s})$
3. $e_3 = (\textit{loan-rates API}, q_3 =$ "current personal loan rates", external, speculative, $t = 0.25\,\text{s})$

The agent commits to a tenant-rights answer that uses only the first search.

Committed part S_C and counterfactual C. e_1 is promoted; e_2 and e_3 are abandoned. The committed-only counterfactual issues the same search as e_1 after commitment.

Provider projection vs. counterfactual. Each provider records its own issued query plus timestamp and destination. The search provider sees e_1 and e_2; the loan-rates provider sees e_3. Had the agent issued only the committed call from the start, the search provider would have seen only its committed execution, and the loan-rates provider would have seen nothing. On the argument and destination channels, the marginal exposure from unaligned events is $\{e_2, e_3\}$; the timing difference for e_1 is an additional component of Δ_{prov}. From the ghost calls the adversaries can infer apartment shopping and personal-loan interest, neither of which the user asked the agent to express.

Contract action. Under issue-time REWRITE on the argument channel, the monitor dispatches e_2 with a generic query that does not reveal an apartment search. Under issue-time SHADOW on the destination channel, the monitor routes e_3 to a trusted-local substitute so the loan-rates provider never receives the speculative request. Both transformations apply at issue time, to speculative events, two of which turn out to be ghost calls, and shrink Δ_{prov} before any externally observable event.

C Evaluation Tables

C.1 Where speculative dispatch is deployed

The body's deployment claim rests on the sources below. We separate what a source states from what the paper infers, and we do not claim a single system exhibits every property at once.

System	Boundary	Deployed	Agent	Third-party receiver	Abandonment stated
Chrome omnibox prefetch (The Chromium Projects, 2024)	search	yes	no	yes	yes
Stream voice agents (Stream, 2025)	tool	prescribed	yes	yes	yes
incident.io (incident.io, 2025)	tool	yes	yes	first-party	yes
Speculative Actions (Ye et al., 2026)	tool	research	yes	mock service	yes
SIA (Hooper et al., 2026)	tool	research	yes	mock service	yes
LiveKit (LiveKit, 2026)	model	yes	yes	yes	yes
Deepgram (Deepgram, 2025)	model	yes	yes	yes	yes

Table 4: Deployment evidence for speculative dispatch. *Boundary* is what the speculative request reaches: a search engine, a tool provider, or the model provider. The model-boundary rows are out of scope for our threat model, which trusts the planner's provider. No row has every column, which is the honest state of the evidence.

Three quotations carry the argument. Chromium's design document states that a prefetched result page "will be fetched with the `Purpose: Prefetch` header" and that the feature is suppressed when "there were too many recent prefetches that the user did not navigate to," so the browser both issues and counts abandoned third-party requests. A voice-platform vendor describes the mispredicted case in full: "Once the full transcription arrives, the LLM realizes the context was negative. LLM simply ignores the tool output in its response … The only cost was a wasted API call." And incident.io reports that "sometimes we speculate incorrectly and have to throw away that work." Each vendor prices the abandoned call as wasted compute. None prices it as disclosure, which is the gap this paper addresses.

A wider pattern shares the observation boundary without the branch-prediction mechanism: agents that issue calls under genuine uncertainty and answer from a fraction of what they retrieved (Amani et al., 2026). We scope that case out, since the runtime never predicted and never abandoned a branch, but the receiver's view is the same.

Ghost calls in the Speculative Actions release. The Speculative Actions repository publishes the result files behind its τ-bench retail experiments: six speculator configurations over 112 to 115 tasks

Policy	Issue-time action	Origin
Naive	dispatch every candidate	speculative endpoint
No-Spec	dispatch nothing before commitment	non-speculative endpoint
SIA	allow read-only tools only	(Hooper et al., 2026)
Rewrite	replace sensitive arguments before dispatch	contract action
Shadow / Shadow + reissue	withhold; serve a trusted local substitute	contract action

Table 5: The five policies in the main text. *Shadow + reissue* adds one rule: when the agent promotes a withheld call, the real call is issued as a committed call. Nine further baselines (ACL, Late-Scrub, Pre-Scrub, Gate, Drop, Taint, Taint-F, Typed-IFC, and Rewrite + reissue) are in Appendix C; Taint and Taint-F are the whole-argument and field-level information-flow baselines, described with Typed-IFC in Appendix C.18.

each, 683 runs in all, in which a guesser predicts the user's next message and a decider runs the agent on that guess, dispatching its tool calls to the environment. Counting a speculative call as promoted when the same call, name and arguments, appears anywhere in the task's real trajectory, and as a ghost call otherwise, Figure 4 gives 1,381 ghost calls among 3,245 speculative ones. 383 of them carry an order identifier the service rejects as unknown, and 24, in 17 conversations the real agent never escalates, hand `transfer_to_human_agents` a summary with the customer's user ID and email address. The environment is the benchmark's simulated retailer, so this shows what the speculator emits, not that a deployed service received it.

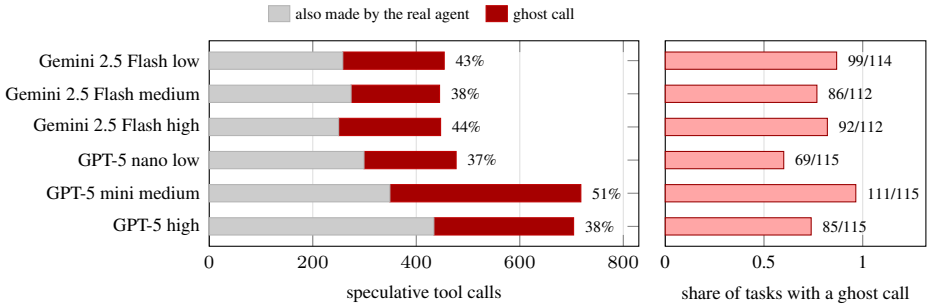

Figure 4: Ghost calls in the published Speculative Actions results for τ-bench retail, one result file per speculator configuration as released (task counts differ where a run omitted tasks). **Left:** speculative tool calls, split into those the real agent also makes and ghost calls, with the ghost share; in total 1,381 of 3,245 (43%). **Right:** tasks with at least one ghost call, 542 of 683 in total. Ghost calls carrying an order identifier the service rejects number 67, 63, 66, 18, 112 and 57 per configuration (383 in total).

Receivers keep what they get. Receiver-side retention is documented rather than assumed. GitHub keeps an organization's audit events for 180 days (GitHub, 2026); Google Workspace keeps most audit logs for six months and administrators "cannot delete log event data" (Google, 2026); an MCP gateway advertises "every JSON-RPC message logged with full payloads" (Glama, 2026). Agent traffic at such receivers is no longer marginal: AI coding agents made 45.3% of requests to one documentation host over 30 days (Mintlify, 2026), and user-directed AI crawling grew more than 21-fold between January and early December 2025 (Cloudflare, 2025).

This appendix collects the per-corpus tables behind every measurement in §5.

C.2 Experiment matrix

Table 6 maps each corpus to its planner, provider, policy set, and result location.

C.3 The same policies across corpora

Table 7 places the τ^2 numbers of Table 1 next to the generated corpora and HotpotQA. Those corpora differ in label space, planner, and adversary, so the columns are not comparable as one measurement, and the ordering is *not* stable across them: SIA and Rewrite invert. SIA is indistinguishable from Naive on the generated corpora and drops to 0.053 on τ^2; Rewrite is the reverse. That inversion is the

Corpus	N	Repl.	Planner	Prov.	Policies	Adversary	Reported in
Static-plan 30	30	3	static	mock	9	Opus 4.7	Tab. 8
Real-planner 30	30	3	real LLM	mock	11	Opus 4.7	Tab. 9
Pooled 150 (paired)	150	3	real LLM	mock	12	Opus 4.7	Tab. 10, Fig. 5
Pooled 150 (calls/policy)	150	3	real LLM	mock	12	Opus 4.7	Tab. 11
Brave 9	9	1	real LLM	Brave	2 (Naive, Rewrite)	Opus 4.7	App. C.14
Async-latency 30 (DDG)	30	3	real LLM	DDG	4 (subset)	Haiku-4.5	Tab. 16
AgentDojo 65	65	1	real LLM	mock	5 (+ SIA)	Haiku-4.5	Tab. 20
Channel-isolation 9	9	1	real LLM	mock	Naive	Opus 4.7 (view)	Tab. 13
τ^2 retail, $b\leq5$	114	2	τ^2	retail	14	schema, Opus	Fig. 2
τ^2 retail, live slice	30	1	τ^2	retail	14	schema	Tab. 27
τ^2 airline, $b\leq2$	50	2	τ^2	airline	8	schema	Tab. 30
τ^2 banking, $b=1$	97	2	τ^2	banking	8	schema, Opus	Tab. 31
τ^2 prefetch, $b=1$	211	2	τ^2	ret./bank.	4	schema, Opus	Tab. 33
HotpotQA distractor	15	1	ReAct	corpus	2	Opus 4.7	§D, Tab. 7

Table 6: Experiment matrix. Rows give each corpus size, replicate or trial count, planner, provider, policy set, adversary, and result location. *Policies* counts those run; the airline and banking tables show 6 and 5 of their 8. *Repl.* counts repeated runs of the same configuration; no seed is passed to a model API, so they are replicates rather than seeded runs.

	generated corpus, 150 tasks		τ^2-bench retail		HotpotQA
Policy	recovery	Δ vs Naive (Holm p)	spec.	ghost-only	recovery
Naive	0.104	(reference)	0.741	0.298	1.00
No-Spec	0.080	-0.024 (1.000)	0.000	0.000	—
SIA	0.124	$+0.020$ (0.141)	0.053	0.053	—
Rewrite	**0.029**	$\mathbf{-0.076}$ (0.015)	0.605	0.259	**0.13**
Shadow	**0.000**[§]	$\mathbf{-0.104}$ (2.1×10^{-5})	—	—	—
Shadow + reissue	—	—	0.022	**0.004**	—

Table 7: Headline privacy results for the five main-text policies. Generated corpus: 150 tasks $\times$ 3 replicates, Opus 4.7, chance 0.0067; the parenthesized p-values use task-level paired signs and Holm correction over eleven tests. τ^2: 114 tasks $\times$ 2 trials at one guess per step, chance 0.042, constant guess 0.211. HotpotQA: 15 questions, Opus 4.7, chance 0.067. [§]On the 150-task corpus every committed call is a promoted speculative candidate, which plain Shadow's substitute also serves, so it makes no provider call at all (0.00 per task) and its 0.000 is suppression by construction; on the static-plan corpus, whose committed plan is separate, it dispatches that plan (1.03 calls per task, Table 8); Shadow + reissue is the deployable variant and is the row to read next to Naive. Nine further policies: Appendix C.

point of Proposition 2, that an action protects the channel it transforms, and the table is here to show it, not to average over it. What does hold everywhere is that Shadow and Shadow + reissue sit at the floor and the post-hoc policies sit at Naive.

C.4 STATIC-PLAN BENCHMARK

Table 8 reports the static-plan result behind *Mechanism* in §5, Q1.

C.5 30-TASK REAL-PLANNER

Table 9 reports the real-LLM-planner numbers behind *Mechanism* in §5, Q1, per replicate and pooled.

C.6 150-TASK PAIRED-FRONTIER REPLAY

Table 10 reports the full numbers behind Figure 5 (*Paired-frontier replay* in §5, Q2).

Is replay open-loop here too? No. Each task on this corpus is one `propose_speculative` call followed by one `commit_branch` call, so there is no later frontier for a rewritten observation to change. Speculative results do reach the commit decision, and a policy can therefore change *which*

Policy	Adv. recovery	Δ vs No-Spec	Wall (ms)	Calls
Naive	0.967	0.522	614	4.03
Late-Scrub	0.967	0.522	615	4.03
SIA	0.967	0.522	616	4.03
ACL	0.967	0.522	616	4.03
Drop (dest)	0.956	0.511	583	3.73
Rewrite	0.533	0.089	583	3.73
Shadow	0.422	−0.022	615	1.03
Drop	0.444	0.000	197	1.03
No-Spec	0.444	0.000	197	1.03

Table 8: Static-plan benchmark on 30 tasks and three replicates. Recovery is closed-set recovery over 30 labels; Δ vs No-Spec is recovery minus the No-Spec floor; calls are mean provider-visible projections. Plain Shadow's 1.03 calls per task are committed calls; its protected speculative calls have no provider projection.

Policy	s1	s2	s3	Mean [95% CI]
Naive	0.467	0.533	0.467	0.489 [0.322, 0.656]
SIA	0.467	0.467	0.533	0.489 [0.322, 0.667]
Drop	0.333	0.300	0.300	0.311 [0.156, 0.478]
Rewrite	0.100	0.000	0.033	0.044 [0.000, 0.100]
Shadow	0.000	0.000	0.000	0.000 [0.000, 0.000]
Gate	0.333	0.300	0.333	0.322 [0.178, 0.478]
No-Spec	0.333	0.367	0.333	0.344 [0.189, 0.500]
Pre-Scrub	0.467	0.467	0.467	0.467 [0.300, 0.633]
Taint	0.300	0.333	0.367	0.333 [0.200, 0.478]
Late-Scrub	0.467	0.500	0.533	0.500 [0.333, 0.667]
ACL	0.500	0.467	0.500	0.489 [0.322, 0.667]

Table 9: 30-task real-planner corpus with 11 policies, three replicates, and simulated tool backends. Per-seed and pooled closed-set recovery; CIs average replicates within task and bootstrap the 30 tasks.

candidates are promoted, but not how many are proposed. On multi-turn τ^2 the same holds because a rewritten speculative call ends its branch (§4), and replay there matches live runs on every task of the 30-task live slice at one guess per step (Appendix D).

Policy	Adv. recovery [95% CI]	Δ vs Naive [95% CI]	Task-sign p	pairs/tasks
Naive	0.104 [0.060, 0.151]	(reference)	n/a	450/150
SIA	0.124 [0.078, 0.176]	+0.020 [+0.007, +0.038]	0.0156	450/150
Drop	0.078 [0.040, 0.118]	−0.027 [−0.062, +0.007]	0.4807	450/150
Rewrite	0.029 [0.007, 0.056]	−0.076 [−0.122, −0.031]	0.0015	450/150
Shadow	0.000 [0.000, 0.000]	−0.104 [−0.151, −0.060]	< 0.0001	450/150
Gate (τ=0.7)	0.080 [0.042, 0.122]	−0.024 [−0.060, +0.009]	0.4807	450/150
No-Spec	0.080 [0.042, 0.122]	−0.024 [−0.058, +0.007]	0.4545	450/150
Pre-Scrub	0.118 [0.073, 0.167]	+0.013 [−0.016, +0.044]	0.4545	450/150
Taint	0.078 [0.042, 0.116]	−0.027 [−0.064, +0.009]	0.3593	450/150
Taint-F	0.071 [0.038, 0.109]	−0.033 [−0.069, +0.000]	0.1892	450/150
Late-Scrub	0.104 [0.062, 0.151]	+0.000 [−0.011, +0.011]	1.0000	450/150
ACL	0.104 [0.062, 0.151]	+0.000 [−0.011, +0.011]	1.0000	450/150

Table 10: 150-task paired-frontier replay behind Figure 5. The adapters on this corpus label every argument INTENT-REVEALING, so Rewrite, Shadow and Drop act on every speculative call. Each policy replays the same Naive-sampled frontier per task and replicate. Replicates are averaged within task; CIs resample tasks and sign tests operate on the 150 task-level mean differences.

C.7 150-TASK PAIRED REPLAY: CALLS PER POLICY

Table 11 complements Table 10 on the same 450 replayed traces per policy with chance-normalized recovery and the provider calls each policy makes.

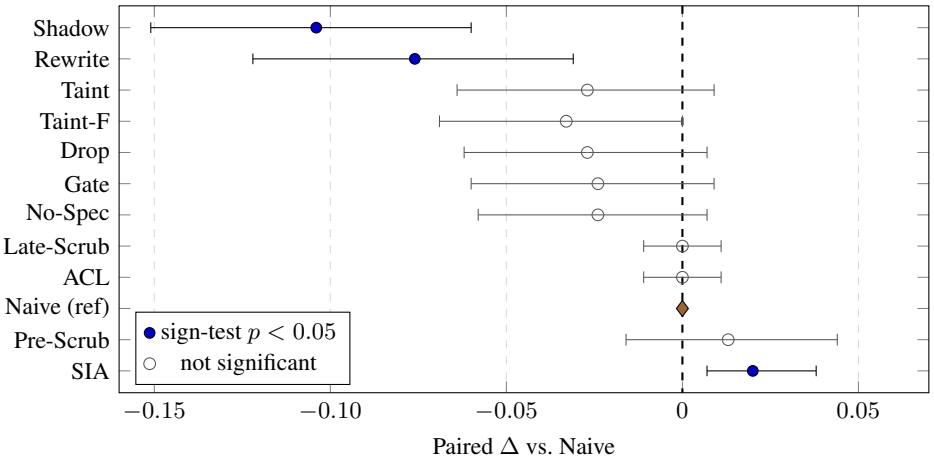

Figure 5: Paired-replay leakage by policy on 150 tasks and three replicates (450 rows per policy). Points show Δ in adversary recovery relative to Naive; error bars use a task-clustered bootstrap and negative values mean less leakage. Filled markers have *uncorrected* task-level sign-test $p<0.05$; after Holm correction over the eleven tests only Rewrite and Shadow remain significant (Table 10). The diamond marks Naive.

Policy	Adv. recovery	Norm. recovery	Provider calls/task
Naive	0.104	0.098	7.95
SIA	0.124	0.119	7.95
Pre-Scrub	0.118	0.112	7.95
Late-Scrub	0.104	0.098	7.95
ACL	0.104	0.098	7.95
Taint	0.078	0.072	2.76
Taint-F	0.071	0.065	2.67
Gate (τ=0.7)	0.080	0.074	1.00
Drop	0.078	0.072	0.96
No-Spec	0.080	0.074	0.96
Rewrite	0.029	0.022	6.72
Shadow	0.000	−0.007	0.00

Table 11: The 150-task paired replay of Table 10 (n=450 per policy): recovery, chance-normalized recovery $(\text{recovery} - 1/N)/(1 - 1/N)$ for N=150, and mean provider calls per task. CIs and paired tests are in Table 10.

C.8 Label-noise robustness

Table 12 reports the false-negative label-noise sweep summarized under *Ablations and robustness* in §5. Noise downgrades SENSITIVE/INTENT-REVEALING argument labels and SENSITIVE destination labels to PUBLIC at the specified rate so the monitor lets the call through. Labels are the policy's only input, so this ablation prices a fallible labeler. An independent-error heuristic illustrates the expected trend: if every frontier had k protected calls and each label failed independently with probability η, at least one would fail with probability $1-(1-\eta)^k$. Interpolating between clean-policy recovery $L_P(0)$ and Naive recovery L_N gives $L_P(0)+[1-(1-\eta)^k](L_N - L_P(0))$. This expression is *not* a security bound: frontiers vary in size, a single missed call need not reveal the intent, and the adversary's success need not be monotone in the calls it sees. The measured six noisy cells, rather than this heuristic, establish the degradation reported here; the shipped constant labeler also violates the independent-error assumption.

C.9 Ablations and robustness, in full

This expands the ablations paragraph of §5; the per-corpus tables it draws on are in §C. *Channel isolation*: under Naive, full projection recovers 9/9, arguments and names 9/9, destination and names 5/9, names only 3/9, so arguments dominate; masking every tool name leaves recovery unchanged (0.49). *Destination anonymization*: neutralizing every destination URL moves Naive from 0.113

Policy	fn-05	fn-10	fn-20
Naive	.122 [.08,.17]	.118 [.08,.16]	.116 [.07,.16]
Pre-Scrub	.138 [.09,.19]	.122 [.08,.17]	.122 [.08,.17]
Late-Scrub	.124 [.08,.17]	.113 [.07,.16]	.124 [.08,.17]
ACL	.122 [.08,.17]	.118 [.07,.17]	.124 [.08,.18]
Rewrite	.027 [.01,.05]	.036 [.01,.06]	.058 [.03,.09]
Shadow	.022 [.01,.04]	.042 [.02,.06]	.078 [.05,.11]
Paired Δ vs clean (same policy/task/replicate):			
Rewrite	−0.002	+0.007	+0.029
Shadow	+0.022	+0.042	+0.078
Paired Δ vs noisy Naive:			
Rewrite	−0.096	−0.082	−0.058
Shadow	−0.100	−0.076	−0.038
Realized argument / destination flip rate:			
	.043 / .067	.099 / .116	.196 / .224

Table 12: False-negative label-noise sweep on the 150-task paired-replay corpus, Haiku-4.5 adversary, three replicates (n=450 per cell). Reports accuracy, paired deltas against clean and noisy baselines, and realized argument/destination flip rates. Intervals are task-clustered bootstraps over the 150 tasks, matching the rest of the paper; at fn-20 the Rewrite delta is significant (sign test $p = 0.007$) and the Shadow delta is not ($p = 0.087$).

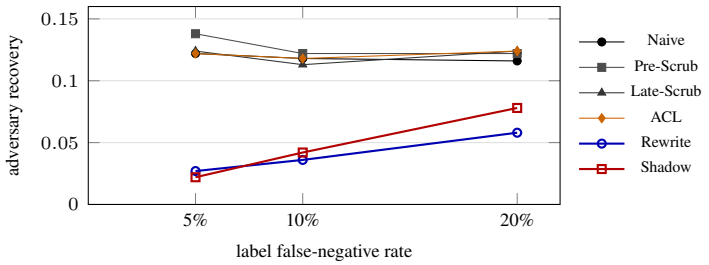

Figure 6: Label noise degrades the two working policies gracefully and leaves the Naive-equivalent cluster flat, because those policies never act on a label in the first place. Even at a 20% false-negative rate Rewrite stays below Naive ($p = 0.007$); Shadow's margin is no longer significant ($p = 0.087$). Same cells as Table 12; 150-task paired replay, Haiku-4.5 adversary, three replicates.

to 0.127 and leaves Rewrite and Shadow unchanged, so topic-bearing URLs are not necessary for leakage in this corpus. *Label noise*: the policy acts on labels, so we flip sensitive arguments and destinations to public at 5/10/20% to simulate labeler false negatives; at 20%, Rewrite (0.06) stays significantly below noisy Naive (task-level sign test $p = 0.007$) and Shadow (0.08) stays below it but not significantly ($p = 0.087$), while the Naive-equivalent cluster stays within a 0.01 spread (Figure 6). *Adversary and metric*: the ranking holds across Opus 4.7, Haiku-4.5, GPT-4o-mini, a 7B open-weight model, keyword and TF-IDF baselines, under top-k, NMI, an open-set adversary and a decoy menu, at a real HTTP boundary (DuckDuckGo, 30 tasks: 0.33 against Rewrite's 0.03), and on AgentDojo. An observer holding labeled traffic history for the same intents reaches 0.65 on the 30-task corpus, while a runtime-log adversary that sees truncated arguments and decision tags reaches 0.24 against the provider adversary's 0.26; absolute recovery therefore depends on the observer's data and view.

C.10 Channel-isolation ablation

Table 13 reports the channel-isolation ablation summarized under *Ablations and robustness* in §5.

The decision channel of §3.2 is measured on the 150-task paired replay: an adversary that sees call counts, tool names, hosts and timing but no arguments recovers no label under any policy (0 of 450 each, against a 0.0067 chance floor). At the coarser level of the three domains, call counts alone reach 0.30–0.45 against a 0.33 chance, the residual that obliviousness would remove.

Adversary view	Acc.	Correct
Full	1.000	9/9
Args + names (no dest)	1.000	9/9
Dest + names (no args)	0.556	5/9
Names only	0.333	3/9

Table 13: Channel-isolation ablation on 9 seed tasks under Naive. Rows mask different observer fields and report adversary accuracy.

On the 30-task real-planner corpus, the complementary tool-name-masking probe leaves pooled accuracy unchanged (0.489 with and without tool names for Naive; 0.467 with and without names for Pre-Scrub; 23 to 27 of 30 per-task predictions match across conditions, per seed and policy).

C.11 Cross-adversary rescoring

Tables 14 and 15 report the cross-model and non-LLM adversary rescoring summarized under *Ablations and robustness* in §5.

Mode	Opus	Haiku	GPT-4o-mini
Closed-set	0.339	0.258	0.219
Open-set ($J \geq 0.5$)	0.025	0.008	n/a
Open-set ($\cos \geq 0.5$)	0.125	0.119	n/a

Table 14: Cross-adversary rescoring across 12 policies and 30 tasks (n=360). Open-set rows report scores under strict lexical and semantic matching rules.

Per-policy on GPT-4o-mini closed-set, Shadow 0.000 and Rewrite 0.033 sit at the floor; Naive, Pre-Scrub and a read-only variant distinct from SIA tie at 0.333 at the top (SIA itself is 0.300); Taint 0.200; the others lie between 0.167 and 0.300. Haiku-4.5 closed-set reproduces the same ranking.

Adversary	Pooled	Training signal
Opus 4.7 (closed)	0.339	label list
Haiku-4.5 (closed)	0.258	label list
Keyword match	0.061	label list
TF-IDF nearest	0.078	label list
Naive Bayes (LOO)	0.650	per-task labels

Table 15: LLM and non-LLM adversaries on the same 12-policy, 30-task trace set (n=360).

C.12 Async-latency sweep

Table 16 reports the single-tool DuckDuckGo async-overlap measurement.

Per-stage decomposition (Naive-async, mean across n=90): t_{propose}=5481 ms, $t_{\text{speculate_dispatch}}$=351 ms, $t_{\text{commit_plan}}$=2501 ms, and $t_{\text{commit_dispatch}}$=0 ms. The recorded $t_{\text{speculate_overlap}}$=2501 ms is the elapsed concurrent phase, $\max(t_{\text{speculate_dispatch}}, t_{\text{commit_plan}})$, not the work hidden by overlap; the latter is separately recorded as $t_{\text{overlap_saving}}$=351 ms, $\min(\cdot,\cdot)$, averaged per run. The 5.4% p50 difference in Table 16 is a separate between-policy measurement and cannot be derived from these stage means.

C.13 Real-provider inference (DDG-30)

The body's real-provider claim (*The action must match the channel*, §5, Q2) re-runs the 30-task corpus against DuckDuckGo Instant Answer with Naive and Rewrite, three replicates, Opus 4.7 closed-set adversary on the resulting projections. This is a different experiment from the async-latency sweep above (which measures wall-clock overlap with a leak side-metric).

Condition	p50	p95	p99	Ghost-only
No-Spec sequential	8.39	10.77	11.56	0.300
Naive-async	7.94	10.37	14.22	0.389
Contract-async (Rewrite)	8.04	10.89	13.25	0.100
Contract-async (Shadow)	8.45	13.01	15.77†	0.000

Table 16: DuckDuckGo async-latency sweep on 30 tasks and three replicates (n=90 per row). Naive-async overlaps speculative dispatch with commit planning; Contract-async adds issue-time policy gating. † Shadow's p99 is 40.20 s with the single 223 s model-API tail included and 15.77 s without it.

Policy	Adv. recovery [95% CI]
Naive	0.333 [0.189, 0.489]
Rewrite	0.033 [0.000, 0.089]

Table 17: DuckDuckGo Instant Answer, 30 tasks × 3 replicates, Opus 4.7 closed-set adversary, chance 0.033; CIs resample tasks. Rewrite sits at the chance floor and the non-overlapping CIs reproduce the synthetic-corpus gap at the real HTTP boundary.

C.14 Runtime/latency (synchronous)

Table 18 reports the synchronous-dispatch wall-clock on the 9-task real-planner study. The Brave-9 real-provider check (Naive 0.67, Rewrite 0.44 ghost-only recovery) is measured on this same 9-task corpus.

Policy	p50	p95	p99	Calls	Block
No-Spec	8392	10437	12040	1.41	1.000
Naive	8583	11237	11964	11.67	0.000
SIA	8287	10750	11650	11.70	0.000
Drop	8220	10067	11126	1.48	0.182
Rewrite	8790	10078	10269	10.19	0.173
Shadow	8542	9804	10458	2.00	0.000
Gate	8182	12098	13198	1.81	0.938

Table 18: Wall-clock latency (ms) on the 9-task real-planner study, three replicates (n=27). Synchronous dispatch; No-Spec is reference. *Block*: fraction refused at issue time. Async-overlap counterpart: Table 16.

C.15 Branch-confidence threshold sweep

Table 19 sweeps the threshold of Gate, the privacy/utility lever named under *Policies* in §5.

C.16 AgentDojo corpus construction

The body cites Table 20 for the external-validity subset. We import 86 AgentDojo tasks. A deterministic keyword rubric labels 25 of them privacy-bearing; a conservative Haiku-4.5 auto-labeler then surfaces 2 additional privacy-bearing prompts the rubric missed (yoga-class scheduling, hotel-spend planning), for a total of 27 privacy-bearing and 59 neutral. Each of the six tabulated policies then produced a trace on 76 to 80 of the 86 under a Haiku-4.5 planner+adversary at one seed (75 to 81 over all twelve policies run). Scoring each policy on its own subset would compare policies on different prompts, so Table 20 reports the 65-task intersection that *every* tabulated policy completed (19 privacy-bearing, 46 neutral). The ordering reproduces under the original 25-task Opus-adversary run.

Attrition bias. The 21 tasks outside the intersection (86→65) are mostly argument-heavy travel and banking prompts, where the redactor (Pre-Scrub) or the taint policies leave the planner unable to fill a required slot; Rewrite and Shadow complete more of them because their argument-transformation surface keeps slot syntax valid. Attrition is therefore not balanced across policies, and intersecting makes the policy comparison use identical prompts but may still select easier tasks. The intersection

τ	Accuracy [95% CI]	Calls/task
0.3	0.120 [0.073, 0.169]	5.44
0.5	0.087 [0.049, 0.129]	1.73
0.7	0.047 [0.020, 0.078]	1.00
0.9	0.058 [0.029, 0.091]	0.96
Naive*	0.104 [0.060, 0.151]	7.95

Table 19: Gate threshold sweep on the 150-task paired-frontier replay, three replicates (n=450 per row); 95% CIs average replicates within task and bootstrap the 150 tasks, while calls/task is the mean provider-visible call count. The four τ rows are one adversary pass and are comparable to each other: raising τ trades disclosure against speculative work, from 5.44 calls per task at τ=0.3 to 0.96 at τ=0.9. *The Naive row is *not* from this pass. Its file is a byte-identical copy of the paired-replay run's Naive, so a delta or sign test against it would pair two different adversary passes, so we report none. Read Gate against Naive in Table 10 instead, where both come from one pass (τ=0.7: 0.080, $\Delta = -0.024$, task-sign $p = 0.48$, not significant after Holm). The gap between that 0.080 and the 0.047 here is the same policy on byte-identical traces (all 450 executed projections match) scored twice: an LLM adversary at its default temperature is not deterministic, which is why every contrast we claim is paired within one pass.

Policy	Adv. recovery	FP rate	Calls
Naive	0.684 [0.474, 0.895]	0.283 [0.152, 0.413]	5.03
SIA	0.684 [0.474, 0.895]	0.283 [0.174, 0.413]	4.91
Pre-Scrub	0.789 [0.579, 0.947]	0.261 [0.152, 0.391]	4.94
Taint	0.105 [0.000, 0.263]	0.022 [0.000, 0.065]	0.32
Rewrite	**0.000** [0.000, 0.000]	0.043 [0.000, 0.109]	5.14
Shadow	**0.000** [0.000, 0.000]	0.000 [0.000, 0.000]	0.03

Table 20: AgentDojo external validity, on the 65 tasks that produced a trace under *every* policy (19 privacy-bearing, 46 neutral), so all six rows are scored on the same prompts. Privacy-bearing inference rate and neutral-prompt false-positive rate under a Haiku-4.5 planner/adversary, with `none` admissible in the closed-set menu; 95% bootstrap CIs over tasks. Our AgentDojo adaptation imports the task prompts and runs them against five read-only adapters, so SIA has nothing to restrict; recovery and false-positive rates equal Naive's, and the small call-count difference reflects independent live planner runs rather than a policy effect. Per-policy trace counts before intersecting were 76–80 of 86.

does not establish performance on those 21 excluded tasks. The qualitative ordering (Shadow $<$ Rewrite $<$ Taint $\ll$ Naive, Pre-Scrub) is preserved under the stricter Opus-adversary configuration on the original 25-task labeled set, which uses a different completion criterion and so is robust to this particular attrition pattern.

C.17 Metric robustness

We rescore the 30-task and 150-task paired-replay traces under four alternative metrics: top-k accuracy (k=1, 3, 5), normalized mutual information (NMI) between true labels and adversary predictions, semantic open-set adversary (sentence-embedding cosine ≥ 0.5), and a decoy-mixed adversary that ranks the true label against ten unrelated decoys (chance ≈ 0.091). Top-k triples top-1 (150-task: 0.07 $\rightarrow$ 0.20) but preserves the ranking (Spearman 0.83). NMI orders Naive (0.64) above Rewrite (0.24) and Shadow (0.00), but with one gold label per task it reduces to the entropy of the predictions, so it measures their spread rather than agreement with the truth (Pre-Scrub scores 0.67); we report it for completeness only. Semantic open-set on the 150-task corpus pools to 0.065, half the 30-task figure (consistent with a larger label space being harder to hit by paraphrase). The decoy-mixed adversary recovers the true label at 0.90–0.93 for the Naive cluster, confirming that the closed-set adversary discriminates via projection content rather than candidate-list recognition. Shadow holds the floor under every metric.

C.18 Field-sensitive IFC baseline (Typed-IFC)

We implement **Typed-IFC**, a field-sensitive information-flow policy in the style of FIDES (Costa et al., 2025): a typed sink table with per-collection accept-sets, field-level taint propagation (lexical

Policy	p50 (s)	speed-up	provider calls/task
No speculation*	4.99	—	4.24
Naive	4.38	+12.1%	12.72
Rewrite	4.44	+11.0%	12.50
Shadow	4.51	+9.5%	0.31
Shadow + reissue	4.60	+7.7%	4.25

Table 21: What the control costs in wall-clock: a separate workload with ≥ 3 parallel speculative calls per task (30 tasks $\times$ 3 replicates), including live DuckDuckGo web search. The table counts *all* provider calls over the 80 (seed, task) pairs from 28 tasks timed under every condition. On those same pairs, DuckDuckGo-only requests total 90 for Shadow + reissue, 85 for No-Spec, and 281 for Naive; the contract is near No-Spec in total call volume, not identical call-for-call. *No-Spec is a sequential reference, not an optimized parallel committed-plan baseline.

labels plus implicit flow), and accept-set compatibility checks. *It is not a reimplementation of FIDES, and the difference matters for reading its numbers.* FIDES attaches static confidentiality and integrity labels to the datastore variables tools read, notes that deployments can reuse the document classifications enterprise suites already expose, and needs only two policies, one for secrecy and one for integrity. Typed-IFC instead labels by topic (medical, legal, financial, employment, security, immigration, relationship, PII, user-query) and carries a per-tool accept-set rather than two global policies, so it is only as good as that taxonomy and is inert wherever the harness declares no source for it to act on. We report it because it is the closest field-sensitive baseline we could run against our frontiers, not as a measurement of FIDES. On the 150-task paired-replay (three replicates, n=450), this policy reaches 0.131 [0.084, 0.182] recovery, above Rewrite (0.029) and Shadow (0.000), and inside Naive's task-clustered interval. The simpler substring-flatten Taint (0.078) and field-level Taint-F (0.071) are no better than it once the eleven-test Holm correction is applied (neither survives against Naive); their stricter blocking rejects topic-aligned flow that FIDES's accept-sets permit. The contract abstraction's REWRITE/SHADOW actions transform projections in ways that source-sink labeling alone does not capture.

C.19 Schema-artifact controls

We test whether topic-bearing destination URLs drive the leak using two complementary controls.

150-task anonymization. We rewrite every destination URL in the replicate-1 150-task paired-replay traces to a neutral schema and rerun the Opus 4.7 closed-set adversary on all 12 policies. Naive recovery shifts from 0.113 to 0.127 (paired CI overlaps zero), Rewrite stays at 0.027, Shadow stays at 0.000. Every paired delta is within ± 0.03 of the original, and the policy ordering is preserved. The replicate-1 0.113 Naive baseline differs from the multi-replicate 0.104 in Tab. 10 because anonymization was rerun on a single seed; what matters is the paired delta.

Enterprise-style corpus. We also generate a fresh 30-task corpus where every tool dispatches to a generic endpoint (e.g., `/api/search`, `/api/retrieve`) regardless of topic; sensitivity is encoded only in argument fields. The six sensitive-topic domains from the 150-task corpus are preserved. Running all 12 policies through this corpus with the Opus 4.7 adversary (three replicates, n=90 per policy) gives Naive 1.00, Pre-Scrub/Late-Scrub/SIA/ACL 1.00, Rewrite 0.089 [0.033, 0.156], Shadow 0.000, and Gate (τ=0.7) 0.956 [0.911, 0.989], level with Drop and No-Spec rather than with Rewrite: on this corpus the frontier carries few low-confidence candidates, so a confidence threshold suppresses almost nothing. Naive recovery does not drop; it rises, partly because the candidate menu is smaller, and the policy ordering is preserved.

Together, the two controls show that topic-bearing destination URLs are not necessary for the observed leak: the argument channel carries the leak on its own, and topic-bearing destinations amplify rather than originate it.

C.20 Multi-tool async benchmark

The single-tool DuckDuckGo overlap of Table 16 saves 5.4% at p50 because $t_{\text{speculate_dispatch}}$ is dwarfed by the planner-LLM call. To measure the control in a multi-tool regime, we built a 30-task

corpus where each task triggers ≥ 3 parallel speculative tool calls (web-search + document-retrieval + calendar-lookup + CRM-lookup combinations). Each condition has 90 recorded rows (30 tasks $\times$ 3 replicates), of which 82 to 90 contain a usable timing: No-Spec 87, Naive-async 83, Rewrite 90, Shadow 85, Shadow-with-reissue 82, Gate 83, and Taint-F 84. Most missing timings are deterministic validation failures on two malformed employment tasks with an empty `name`; three are failed planner/provider responses. We do not count failures as zero-time runs or compare conditions on different subsets. Instead, `scripts/analyze_async_intersection.py` uses the 80 (seed, task) pairs, spanning 28 tasks, timed under *every* condition. On that intersection (Table 21) the p50 savings against sequential No-Spec are 12.1% (Naive-async), 11.0% (Rewrite), 9.5% (Shadow), 7.7% (Shadow-with-reissue), 13.1% (Gate), and 12.2% (Taint-F). The paired median savings and task-clustered 95% intervals are 11.7% [7.7, 15.7], 12.4% [7.0, 15.6], 11.2% [8.6, 14.0], 8.1% [4.8, 11.2], 13.3% [5.6, 16.6], and 13.3% [9.0, 16.8], respectively. Those marginal intervals overlap. The paired contrast between two policies is the sharper test, and it is the one we apply to every privacy comparison in this paper, so we apply it here too: on the same 80 pairs Naive is faster than Shadow + reissue by a median 6.8% of the sequential baseline [1.2, 10.4], task-level sign test $p = 0.013$ over 28 task clusters. The contract therefore gives up about half of speculation's saving on this workload. **Shadow with re-issue** makes 4.25 provider calls per task against No-Spec's 4.24 on that intersection. Its closed-set leak equals No-Spec's 100% floor because the committed plan alone identifies intent on this corpus. Most importantly, Shadow + reissue overlaps only calls allowed at issue time: each withheld promoted read executes after commitment, contributing no hidden tool latency.

C.21 AGENTDOJO METRIC FIX

We score AgentDojo with `none` admissible in the closed-set menu. Without it the adversary must pick a topical label on every neutral prompt, which yields a 1.000 false-positive rate for the let-the-arguments-through baselines (Naive, SIA, Pre-Scrub) and even for Rewrite, whose arguments are a generic wordbag. With `none` admissible, Rewrite's FP rate is 0.058 [0.000, 0.135] and the let-through cluster is 0.29–0.33 when each policy is scored on *all of its own completed traces*. Table 20 instead restricts every policy to the *same 65 completed tasks*; there Rewrite is 0.043 [0.000, 0.109] and the let-through cluster is 0.26–0.28. These are different denominators, not different adversaries. The let-through policies still expose literal arguments, so the adversary often picks a sensitive label even with abstention available. Privacy-bearing inference is approximately stable across the menu fix (within 12 percentage points per policy; the largest move is Late-Scrub's, 0.846 to 0.731). Table 20 reports the common-task numbers as the headline. ACL and Late-Scrub were run as well and fall in the let-through cluster, as expected without argument transformation: ACL recovers 0.870 with a 0.291 FP rate, Late-Scrub 0.731 and 0.286, on their own traces; they are outside the six-policy intersection because intersecting all eight would leave 62 tasks.

C.22 LABELER VALIDATION

Theorem 1 assumes (A2) label correctness. To stress this assumption we validated the prototype's rule-based labeler against an Opus 4.7 gold annotator on 394 events sampled across the 30-task sensitive corpus, the 150-task pooled corpus, and the 30-task enterprise corpus.

Channel	Accuracy	Macro-F1
Destination privacy class	0.726	0.585
Argument privacy class	0.310	0.095
Effect class	0.810	0.447

Table 22: Rule-based labeler vs Opus 4.7 gold annotator on n=394 events. Macro-F1 is below 0.6 on every channel; the exact label-correctness assumption (A2) does not hold for this labeler.

Three concrete failure modes drive the result. First, the per-adapter contract emits a single constant argument label (INTENT-REVEALING) regardless of argument content; the labeler never inspects the args. Gold labels span public/personal/sensitive/intent-revealing roughly uniformly, so the constant emission scores at chance. Second, the destination labeler never emits PSEUDONYMOUS; enterprise endpoints like `/api/users/{sha1}` are flagged PUBLIC, and the labeler over-labels routine document-retrieval calls as PERSONAL. Third, the effect-class labeler emits EXTERNAL-READ for

calls that gold considers SEMI-STATEFUL (provider-side logging counts as state). The local classifier evaluated below improves the protect decision, but its accuracy remains schema-dependent.

C.23 LABELER QUALITY AND THE COST OF GETTING IT WRONG

The shipped rule-based labeler emits one constant argument class per adapter, which is why Table 22 scores it near chance on that channel. A local classifier removes that defect without leaving the trusted boundary: an all-MiniLM-L6 embedding of the rendered call plus 25 structural features, fed to a class-balanced logistic regression and evaluated with task-grouped five-fold cross-validation over the same 394 gold events, reaches macro-F1 0.601 ± 0.015 on four classes and $F_1 = 0.84$ on the binary protect decision that contracts actually act on (recall 0.816, precision 0.873). As a reference for annotation noise, relabeling every event with the same annotator, prompt, and batching agrees with the original gold at macro-F1 0.739 and $\kappa = 0.68$, and the probe's macro-F1 is 81% of that repeat-agreement reference. It transfers poorly to the anonymized-endpoint enterprise corpus (0.337), and neither end-to-end fine-tuning (0.585) nor a zero-shot 7B model (0.447) beats it, which is consistent with annotation noise contributing to the gap. Dropping the trained probe into τ^2 without retraining protects only 0.8% of calls and agrees with the schema labeler on 31% of the 390 distinct calls, so a labeler has to be written for, or trained on, the schema it guards. Batched over a whole frontier on CPU the probe costs 28.1 ms at four threads (p99 49.3 ms) and 55.8 ms at one.

Labeler	Protect recall	ms/frontier	Served	Excess vs. No-Spec [95% CI]
none (Naive)	—	—	1.000	$+0.324$ [$+0.253$, $+0.409$]*
Qwen2.5-7B, call only (GPU)	0.657	45	0.683	$+0.098$ [$+0.031$, $+0.167$]
Qwen2.5-7B, + user msg (GPU)	0.829	45	0.507	$+0.044$ [-0.024, $+0.120$]
MiniLM + logistic probe (CPU)	0.816	28–56	0.502	$+0.022$ [-0.056, $+0.102$]
gold annotator (reference)	0.889	—	0.491	$+0.004$ [-0.073, $+0.084$]

Table 23: Assumption A2 priced in offline end-to-end policy replay. Rows are selective Rewrite with re-issue on the 150-task corpus (450 rows; 150 independent tasks), so promoted calls keep or reissue real arguments; this is not an independent task-success grade. Leakage is scored by the strongest local observer, trained on the same policy's traffic from other replicates. *Promoted served* is the share of promoted calls still served from speculation. CIs resample task clusters. After task-level sign tests and Holm correction, only the no-labeler Naive row differs from No-Spec; the four practical/reference labeler rows do not. This full-corpus replay is a mechanism ablation, not a held-out test of the trained probe: 70 tasks contributed labels to training. Table 24 reports the disjoint-task check.

Policy	Recovery [95% CI]	Promoted served	Δ vs. Naive [95% CI]
Naive	0.733 [0.638, 0.821]	1.000	—
No-Spec	0.338 [0.242, 0.438]	0.000	-0.396 [-0.513, -0.283]
MiniLM Rewrite+reissue	0.450 [0.358, 0.546]	0.555	-0.283 [-0.379, -0.183]

Table 24: Post-hoc task-disjoint check of the trained local labeler: 80 extension tasks (240 frozen replays) that supplied no training annotations. Recovery is from the stronger of the matched and rewrite-aware local observers, trained on other replicates; intervals resample tasks. Relative to Naive, the probe reduces recovery ($p = 3.4 \times 10^{-7}$, exploratory task-level paired sign test), while its point estimate remains above No-Spec. Promoted served counts reused calls, not independently graded task success.

Exploratory operating point. Choosing the protect threshold on the 70 training tasks alone, by a rule fixed in advance (training recovery within 0.05 of No-Spec's, most reuse), selects the default 0.5 used in Table 24. A stricter rule we adopted only after seeing that result (training recovery no higher than No-Spec's) selects 0.2. On the 80 held-out tasks it gives 0.396 [0.300, 0.492] recovery against No-Spec's 0.338, a difference that is not significant ($\Delta = +0.058$ [-0.029, $+0.150$], $p = 0.57$) but is not a demonstrated equivalence, while still serving 29.5% of promoted calls from speculation.

C.24 SPECULATIVE-FRONTIER STATISTICS

Table 26 reports aggregated frontier characteristics on the 30-task real-planner corpus and the 150-task paired-replay corpus (three replicates each, pooled). Speculative frontiers are small (median 8 candidates) and retrieval-heavy; abandoned candidates outnumber promoted ones 607 to 141 (4.3×)

Workload	Intent carried by	Contract measured	Served share
τ^2-bench retail	write-tool identity	Shadow + reissue	1.00 (0.92 of 0.92)
τ^2-bench banking	read argument	Shadow + reissue	0.64 (38 of 59 calls)
Generated corpora, 150 tasks	read argument	selective Rewrite + reissue	0.50 labeler, 0.49 reference

Table 25: Where withholding is free and where it is paid for. *Served share* is promoted calls answered from speculation, relative to Naive; the banking share uses the exact 38/59 count, not the rounded 0.20/0.30 calls per task. On retail it is 1.00 because protected calls are writes or lookups that speculation never served anyway; on the read-argument workloads it falls because protecting a promoted read means withholding it. The multi-tool workload shows the same exchange in wall-clock, where Shadow + reissue keeps a 7.7% saving against 12.1% for unprotected speculation, consistent in direction rather than in magnitude. The three rows report the contract actually run on each corpus: Shadow + reissue was not run on the 150-task corpus, and neither it nor a served-share measurement exists for HotpotQA, whose 15 questions place it on the same read-argument side of the boundary by construction (its speculator abandons 4 of 123 calls).

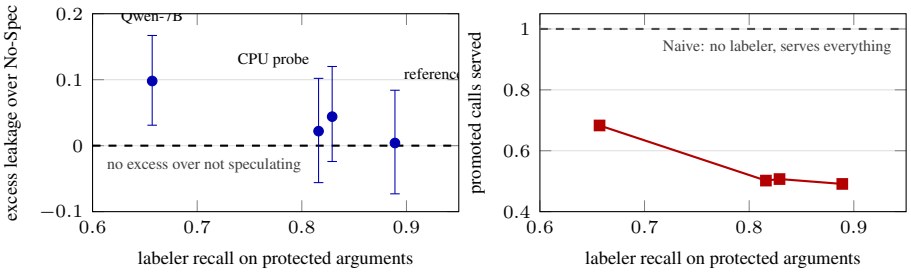

Figure 7: Assumption A2, priced in offline replay. **Left:** as a local labeler protects more of what should be protected, the point estimate of excess leakage over the no-speculation floor falls toward zero; none of the four labeler rows differs from No-Spec after task-level Holm correction. **Right:** the cost is speculative reuse, since a labeler that protects more calls serves fewer of them from speculation. Naive runs no labeler and appears only as the dashed reference at 1.0. Rows and task-clustered intervals are those of Table 23.

on the 30-task corpus and 3,146 to 432 (7.3×) on the 150-task corpus, supporting the framing that the planner emits speculation as cheap probing rather than confident pre-commit dispatch.

Metric	30-task	150-task
Tasks (pooled across replicates)	90	450
Mean frontier size	8.31	7.95
Median frontier size	8	8
95th-percentile frontier size	14	12.5
Frontier-size range	2–16	1–19
Mean ghost-call rate	82.1%	88.6%
Mean promotion rate	17.9%	11.4%
Mean branch confidence	0.39	0.35
Top tool (share on frontier)	doc-retr. 44%	doc-retr. 39%

Table 26: Speculative-frontier statistics across corpora, pooled over three replicates.

D τ^2-BENCH AND HOTPOTQA INTEGRATION

Harness. The unmodified τ^2 `LLMAgent` runs the retail domain. Before each agent turn the released Speculative Actions speculator proposes b next tool calls; each proposal passes the contract dispatcher and, if dispatched, executes against a deep copy of the environment, so a speculative write never touches the real database. When the agent issues its own call, a matching speculative branch is promoted: a promoted read that executed speculatively and was allowed is served from that result, and every other committed call is dispatched at commit through the same dispatcher, where a policy may refuse or alter it. Branch 0 is byte-identical across widths and later branches are sampled at temperature 0.7 or above, with a key-only branch tag so the request cache cannot serve one branch's

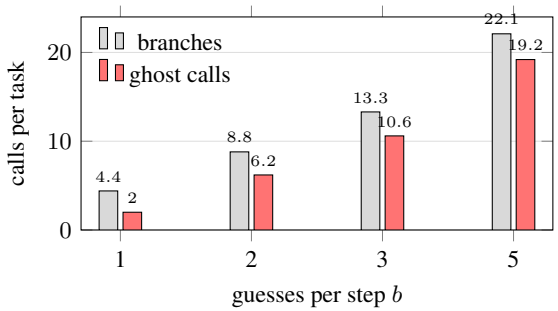

Figure 8: Absolute frontier size on τ^2-bench retail, per task, at each speculation width: branches the speculator proposes and the abandoned part of them. In total the captures hold 457 ghost calls at b=1 and 4,387 at b=5 across 228 traces. This is the counts panel of Figure 2, shown here at a readable size.

response to another. The grader is τ^2's own; a strict-replay mismatch (which a policy that refuses a committed call can cause) is recorded and re-evaluated without strict replay.

Sequentiality. In the 228 captured traces at one guess per step, 813 of 1,139 tool-calling assistant turns (71.4%) issue a single call, and 93 of them in all, 0.41 per task, are answered entirely from speculation. A committed runtime can parallelize independent calls within the remaining multi-call turns after the agent emits them, but it cannot parallelize a one-call turn with a later turn that consumes its result. Pre-commit speculation is the mechanism that overlaps those sequential calls with model planning; served-call count therefore measures available overlap, while realized latency still depends on the planner and tool times.

How the corpora were built. The benchmark, the simulated users, the grader, and the speculator are all third-party, so the τ^2 tasks were not written by us and cannot have been shaped to favor a contract. Where we do supply tasks, two choices keep them honest: the 30-task static-plan corpus was hand-written with fixed committed plans *before* any policy existed, so holding the plan constant isolates the ghost-call contribution rather than the planner's choices, and the 150-task corpus is template-generated and deduplicated, one gold label per task. Every policy then replays a byte-identical frontier, so no policy is scored on a different task distribution. The one choice that could have flattered us, topic-bearing destination URLs, is controlled for by a corpus with anonymous endpoints (Appendix C.19).

Intents and labels. The adversary maps each retail API to one intent by schema and never infers it from a model: the seven read tools and the local calculation map to no intent, and `cancel_pending_order`, `exchange_delivered_order_items`, `return_delivered_order_items`, `modify_pending_order_{items,address,payment}`, `modify_user_address`, and `transfer_to_human_agents` map to one each. A task's label is the set of intents in its gold actions, which gives 24 labels over the 114 tasks. The contract labeler is local: effect classes come from the benchmark's read/write metadata and argument sensitivity from the retail schema (names, addresses, payment methods, item identifiers). A no-tool control scores 0.000 task success on 10 tasks, so these tasks do need tools.

Replay validity. Running fourteen policies live at four widths is not affordable, so we capture Naive live and replay the others over the captured frontiers. Table 28 checks that against live runs of every policy on the same 30 tasks. Replay matches live recovery and task success for policies that only allow or suppress speculative calls, including the primary comparison; Drop's ghost-call count differs slightly. It is exact for Rewrite with its branch-ending rule as well, because a rewritten speculative call then ends its branch: a branch matches Naive's up to its first rewritten call and stops, so the live run and the replay agree on every task. Without the rule it is not (row d of Table 28). Policies that refuse or alter the agent's own committed calls change its trajectory: over the full capture, access control refuses 1,780 committed calls and issue-time redaction alters 1,383 and refuses 297, and

Policy	replay, 114 tasks × 2 trials spec.	ghost-only	served	live, 30 tasks × 1 trial spec.	ghost-only	success	commit
Naive	0.741	0.298	0.92	0.800	0.300	0.600	7.00
No-Spec	0.000	0.000	0.00	0.000	0.000	0.600	7.87
Shadow + reissue	0.022	0.004	0.92	0.000	0.000	0.600	7.00
Drop	0.022	0.004	0.92	0.000	0.000	0.600	6.93
Gate	0.000	0.000	0.00	0.000	0.000	0.600	7.87
SIA	0.053	0.053	0.92	0.033	0.033	0.600	7.00
Late-Scrub	0.741	0.298	0.92	0.800	0.300	0.600	7.00
Rewrite	0.605	0.259	0.88	0.700	0.300	0.600	7.00
Pre-Scrub†	—	—	—	0.400	0.133	0.200	5.53
ACL†	—	—	—	0.000	0.000	0.000	0.00

Table 27: τ^2 retail at one speculative guess per step. The replay and live columns are different task sets (114 × 2 against 30 × 1), so their values differ even where replay is exact for the policy. Task success in the replay columns is the captured run's (0.575) and is not the policy's own; †these two refuse or alter the agent's committed calls, so only their live rows are valid (Table 28). Taint, Taint-F and Typed-IFC behave exactly as Naive here because this harness declares no taint source for them to act on, so we omit them rather than report an inert row; this is a limit of our harness, not a result about information-flow control. Rewrite + reissue matches Rewrite on every column.

every such row is flagged and reported live instead. A policy that changes the agent's own calls can change the frontier it is evaluated on, which is why those rows are reported live.

Policy (30 tasks)	spec. live/replay (agree)	ghost-only live/replay (agree)	ghosts/task	success (agree)
Naive, No-Spec, Shadow + reissue, SIA, Late-Scrub, Drop, Gate, Rewrite[c]	identical (1.00)	identical (1.00)	same except Drop[a]	1.00
Pre-Scrub[b]	0.40 / 0.77 (0.57)	0.13 / 0.27 (0.80)	1.63 / 1.13	0.53
ACL[b]	0.00 / 0.00 (1.00)	0.00 / 0.00 (1.00)	0.00 / 0.00	0.40
Rewrite, default[d]	0.80 / 0.80 (1.00)	0.53 / 0.30 (0.77)	4.50 / 1.97	1.00

Table 28: Replay against live on the same 30 tasks and trial. Every *agree* figure, including the last column, is the fraction of tasks on which live and replay give the same outcome, not a task-success rate. [a]Drop observes 0.13 against 0.10 ghosts per task with identical recovery. [b]Refuses or alters committed calls, so its replayed task success is invalid. [c]Both with and without re-issue, with a rewritten speculative call ending its branch (§4). [d]Without that rule, live: the speculator keeps building on rewritten lookups, which replay over Naive's frontier cannot reproduce; Naive on the same tasks has 1.97 ghosts per task and 0.30 ghost-only recovery.

b	Naive: Opus 4.7 / schema speculative	ghost-only	S+R: Opus / schema ghost-only	per-task agreement
1	0.693 / 0.741	0.294 / 0.298	0.018 / 0.004	0.91–0.97
3	0.702 / 0.746	0.706 / 0.741	0.044 / 0.022	0.89–0.96
5	0.715 / 0.750	0.711 / 0.750	0.075 / 0.022	0.89–0.90

Table 29: The paper's Opus 4.7 adversary re-scoring the τ^2 traces at every width (114 tasks × 2 trials, chance 0.042). It reads observed arguments and destinations rather than tool identity, and reproduces both the ghost-only projection and its saturation with width. Naive − Shadow + reissue on the ghost-only projection under this adversary is $+0.276$, $+0.662$, and $+0.636$ at $b = 1, 3, 5$ (Holm $p \leq 6.5 \times 10^{-14}$); No-Spec recovers 0.000 everywhere. The contract's residual grows with width under this adversary (0.018 to 0.075) while the schema classifier holds it at 0.022, because the LLM also reads the benign lookups the contract allows.

D.1 Second domain: airline

The same harness, speculator and models run τ^2 airline, 50 tasks × 2 trials at one and two guesses per step, with Naive captured live and the other policies replayed over its frontiers as on retail. Replay follows the mechanism verified live there (Table 28): an allow or suppress decision leaves the frontier unchanged, and a rewritten call ends its branch; the same holds for banking below. The schema adversary maps the read tools to no intent and `book_reservation`, `cancel_reservation`,

`send_certificate` and the three `update_reservation_*` tools to one each, which gives 11 labels. 23 of the 50 tasks need no write, so a constant "no write" guess is right on 46% of tasks and accuracy over all of them mostly measures that guess. We therefore score as AgentDojo does: recovery on the 27 tasks that need a write (10 labels, chance 0.100, constant-guess baseline 0.296), and on the other 23 the rate at which the adversary claims a write intent the task never had. The contract labeler follows the retail rule: passenger, payment and itinerary fields are sensitive, reservation and user identifiers pseudonymous. Naive's ghost calls recover the intent of about half the write-bearing tasks and, on the neutral ones, claim a write the user never asked for on 85–91% of runs, the airline form of the misattribution in §5. Shadow + reissue, SIA and No-Spec take recovery to 0.000 at both widths (task-level sign test against Naive, $p \leq 7.6 \times 10^{-6}$), and Rewrite stays close to Naive because, as on retail, the arguments it rewrites do not carry the intent (Table 30).

	b=1				b=2			
Policy	ghosts	ghost-only	spec.	false claim	ghosts	ghost-only	spec.	false claim
Naive	3.33	0.481	0.593	0.848	8.53	0.556	0.574	0.913
Rewrite	2.54	0.370	0.500	0.804	6.47	0.463	0.481	0.891
SIA	1.10	0.000	0.000	0.000	3.10	0.000	0.000	0.000
Shadow + reissue	0.72	0.000	0.000	0.043	2.04	0.000	0.000	0.043
Drop	0.72	0.000	0.000	0.043	2.04	0.000	0.000	0.043
No-Spec	0.00	0.000	0.000	0.000	0.00	0.000	0.000	0.000

Table 30: τ^2 airline, 50 tasks $\times$ 2 trials (Naive live, the other policies replayed over its frontiers as on retail; replay is checked against live runs on retail only). Recovery is on the 27 tasks that need a write (chance 0.100, constant-guess baseline 0.296); Naive's ghost-only recovery is 0.481 [0.333, 0.630] at b=1 and 0.556 [0.407, 0.704] at b=2. *False claim*: the rate at which the adversary names a write intent on the 23 tasks that have none. *Ghosts*: provider-observed abandoned calls per task. Task success is 0.550 in every row.

D.2 Third domain: knowledge-base banking

τ^2 banking (Shi et al., 2026) is the domain in which intent travels in reads. The agent answers a banking customer by searching a 698-document knowledge base and reaches most of its write tools only after unlocking them by name, so what the customer wants is spelled out in the text of its queries before any write is chosen. The same harness, speculator and models run it, 97 tasks $\times$ 2 trials at one guess per step with the benchmark's local BM25 search; Naive is captured live and the other policies replayed over its frontiers. The schema adversary maps each hidden write tool to one intent by the type the benchmark gives it, and reads, the identity-verification log and tool unlocking to none, which gives 37 labels; the customer's own tool calls are not the agent's and do not enter the label. Because the intent sits in query text, a tool-identity classifier understates the channel, and we report our Opus 4.7 adversary, which reads what the provider received (Table 31). Naive's ghost calls alone recover the customer's intent on 0.335 [0.263, 0.407] of tasks, 12$\times$ chance and above the 0.258 constant guess, and on 0.285 [0.201, 0.368] of the 72 tasks that need a write, twice that subset's 0.139 constant guess. Shadow + reissue takes it to 0.036, and to 0.007 on the write-bearing tasks (task-level sign test against Naive, $p = 1.3 \times 10^{-12}$ over all tasks and 9.3×10^{-10} over write-bearing ones), while serving 0.20 of Naive's 0.30 promoted calls per task: the reads it withholds are the 15 knowledge-base searches and 6 name lookups that carry the intent. A promoted call to a state-changing tool, such as unlocking a hidden tool, re-executes at commit rather than being served from speculation. SIA is where this domain separates the policies. Refusing to speculate writes closed the channel on retail and airline, but here SIA leaves 0.247: its ghost calls are the knowledge-base queries it still allows, which the tool-identity classifier cannot read (0.000 on write-bearing tasks) and the Opus adversary can, both on the tasks that need a write (0.069 against Shadow + reissue's 0.007 under the miss-scored recovery metric, $p = 0.004$; the frequency-calibrated observer of Appendix D.3 detects no improvement there) and on the others, where the queries reveal that the customer wants information rather than a change ($p = 3.7 \times 10^{-9}$ over all tasks). A read-only speculation policy is therefore not a privacy policy, which is Proposition 2 on a third-party benchmark, and the contract closes the channel SIA leaves open. Task success is identical under every policy because replay keeps Naive's trajectory; at 0.05 the domain is hard for Haiku-4.5, and no recovery number depends on it.

Policy	ghosts	ghost-only all tasks	ghost-only write tasks	speculative all tasks	speculative write tasks	served
Naive	4.71	0.335	0.285	0.361	0.319	0.30
SIA	3.40	0.247	0.069	0.253	0.083	0.30
Shadow + reissue	0.44	**0.036**	**0.007**	0.052	0.021	0.20
Drop	0.44	0.036	0.007	0.052	0.021	0.20
No-Spec	0.00	0.000	0.000	0.000	0.000	0.00

Table 31: τ^2 banking, 97 tasks $\times$ 2 trials at one guess per step (Naive live, the other policies replayed over its frontiers as on retail, where replay is checked against live runs), scored by the Opus 4.7 adversary over 37 labels: chance 0.027, constant guess 0.258 over all tasks and 0.139 over the 72 that need a write. Naive $-$ Shadow + reissue on the ghost-only projection is $+0.299$ [$+0.222$, $+0.376$] (task-level sign test $p = 1.3 \times 10^{-12}$); SIA exceeds Shadow + reissue ($p = 3.7 \times 10^{-9}$). *Ghosts*: provider-observed abandoned calls per task. *Served*: promoted calls answered from speculation per task.

D.3 Frequency-calibrated observer

The recovery metric scores an empty trace as a miss, which isolates the ghost-only projection but is not what an observer that knows task frequencies would do. We therefore also score every trace with a probabilistic observer: $P(y) = \pi(y)$ for an empty trace and $P(y \mid \hat{y}) = \lambda\, \mathbf{1}[y = \hat{y}] + (1 - \lambda)\, \pi(y)$ otherwise, where $\hat{y}$ is the Opus 4.7 adversary's answer on the ghost-only projection, π is the training tasks' label frequency (add-0.5 over the full label menu), and λ maximizes the training likelihood. Five folds are grouped by task, so both trials of a task are held out together, and nothing is chosen on the test folds. Table 32 reports, on held-out tasks, the prior's and the observer's accuracy and log loss and the improvement in bits, with task-bootstrap CIs and a paired sign-flip permutation test on the mean. A per-task sign test suits accuracy rather than log loss, where the observer loses a little on most tasks and gains much on a few; on accuracy it confirms Naive (40 tasks up against 16 on retail, $p = 0.002$; 31 against 12 on banking, $p = 0.005$) and does not detect SIA (5 against 1 on banking, $p = 0.22$). These are log-loss improvements for this observer, not a bound on the information the traces contain. The single-λ observer agrees with a two-parameter variant (separate weights for *no write* and write answers) and with the schema classifier as adversary on Naive, which gains 0.36–0.46 bits on retail and 0.36–0.67 on banking, and on Shadow + reissue, which gains no more than 0.001. SIA's banking gain holds with two weights (0.140 bits, $p = 0.029$) but not for the schema classifier (-0.001), which sees tool identity only. It splits into 0.002 bits on the 72 write-bearing tasks ($p = 0.98$) and 0.48 bits on the 25 no-write ones ($p = 5 \times 10^{-5}$): its knowledge-base queries mainly reveal that the customer seeks information rather than a change. The write-task recovery contrast of Table 31 (0.069 against 0.007) is therefore a property of the miss-scored metric rather than an improvement this observer detects, and we treat SIA's 0.125-bit result as exploratory.

Domain	Policy	Accuracy	Log loss (bits)	Improvement [95% CI], p
Retail	Naive	0.211 $\to$ 0.329	4.13 $\to$ 3.77	$+0.365$ [$+0.188$, $+0.545$], 1.0×10^{-4}
	SIA	0.211 $\to$ 0.211	4.13 $\to$ 4.13	$+0.006$ [-0.009, $+0.022$], 0.51
	Shadow + reissue	0.211 $\to$ 0.211	4.13 $\to$ 4.14	-0.009 [-0.015, -0.004], 8.5×10^{-4}
	No-Spec	0.211 $\to$ 0.211	4.13 $\to$ 4.13	0
Banking	Naive	0.258 $\to$ 0.376	4.76 $\to$ 4.10	$+0.668$ [$+0.388$, $+0.976$], 5×10^{-5}
	SIA	0.258 $\to$ 0.278	4.76 $\to$ 4.64	$+0.125$ [$+0.013$, $+0.245$], 0.035
	Shadow + reissue	0.258 $\to$ 0.258	4.76 $\to$ 4.76	-0.000 [-0.001, $+0.001$], 0.42
	No-Spec	0.258 $\to$ 0.258	4.76 $\to$ 4.76	0

Table 32: Frequency-calibrated, task-held-out observer on the ghost-only projection (Opus 4.7 answers; retail 114 and banking 97 tasks, two trials each). Accuracy and log loss read prior $\to$ observer. *Prior*: the training tasks' label frequencies; accuracy is its mode and the observer's argmax. *Improvement*: prior minus observer log loss, a task mean in bits; p is a paired sign-flip permutation test. The retail Shadow + reissue row is slightly negative because the fitted weight is not exactly zero in every fold.

HotpotQA. This is an illustrative case study, not a benchmark-scale evaluation: 15 questions for recovery and 5 for the exact-match check, drawn as the first questions of the distractor-setting validation split under a fixed seed with no filtering on content or difficulty. At n=15 a single question

moves recovery by 6.7 points, so we report it with that caveat attached and draw no ordering from it beyond the direction of the effect. The agent answers distractor-setting questions by searching a local corpus, and the speculator guesses its next search. That guess is usually right: at one guess per step only 4 of the 123 speculative calls issued across 15 questions are abandoned, and 3 of those reach the provider as ghost calls, so the ghost-only projection barely exists and the Rewrite result on this benchmark (Opus 4.7 recovery 1.00 to 0.13) describes the speculative projection. Blanket Rewrite also answers promoted searches with the generic query, which cuts exact match from 0.60 to 0.20; with re-issue on promotion the provider still saw only generic queries on every speculative call, real queries left only after commitment, and exact match matched Naive on each of 5 questions. At three guesses per step the picture changes and 14 observed ghost calls per question appear, which is the same width effect τ^2 shows.

Cost. The width sweep, the live slice, and the adversary re-scoring cost $97 in total with Haiku-4.5 for the harness and Opus 4.7 for the adversary, of which the five-guess capture is the largest single item; branch k's requests do not depend on the width, so the two- and three-guess captures replayed from the request cache at no cost. The airline captures cost $20 more, and the banking captures, with the pilot, gates, and a re-capture after a harness fix, $132, plus $7 of Opus re-scoring. Replay itself is free.

D.4 A SECOND SPECULATOR

To test whether the ghost-only projection belongs to one speculator, we add a second with a different mechanism: at each step it sends the conversation to GPT-4o-mini once and dispatches up to three predicted next calls, where the Speculative Actions speculator guesses the user's next message and runs the agent on it. Both run at one guess per step on the same retail and banking tasks, two trials each, with Naive captured live and the other policies replayed over each speculator's own capture; agent, simulated users, grader and adversaries are unchanged. Two criteria were fixed before the runs: Naive above No-Spec, and Shadow + reissue below Naive, each at sign-test $p < 0.01$ over all tasks.

			ghost-only, all tasks			write tasks		Naive > S+R
Domain	Speculator	ghosts/task	Naive	SIA	S+R	Naive	S+R	p (all tasks)
Retail	Speculative Actions	2.0	0.294	0.066	0.018	0.285	0.014	2.4×10^{-14}
	prefetch	4.9	0.404	0.057	0.044	0.416	0.023	1.4×10^{-10}
Banking	Speculative Actions	4.7	0.335	0.247	0.036	0.285	0.007	1.3×10^{-12}
	prefetch	10.1	0.237	0.216	0.201	0.174	0.007	0.49

Table 33: Two speculators on the same τ^2 tasks at one guess per step (retail 114, banking 97 tasks, two trials), Opus 4.7 adversary on the ghost-only projection. No-Spec recovers 0.000 in every cell, and Naive exceeds it at $p \leq 9.3 \times 10^{-10}$ throughout. *Write tasks*: the 107 retail and 72 banking tasks that need a write (a different set from the 107 retail tasks whose agent commits one in App. D.5; they overlap on 105); on them Shadow + reissue removes the leak for both speculators ($p \leq 1.5 \times 10^{-4}$).

Retail replicates, and the second speculator leaks more because it abandons more calls. Banking meets the first criterion but not the second over all tasks, and the reason is the metric rather than a disclosure. This speculator does issue knowledge-base searches, 44 of them across the corpus, every one abandoned and provider-visible under Naive and SIA and none of them visible under Shadow + reissue; what it does not do is put intent in them, and the labeler marks none of the 44 intent-bearing. What Shadow + reissue lets through are account lookups keyed by the pseudonymous user ID (card accounts and transactions, profile, current time), which the labeler passes as benign and which appear in 136 of 144 write-task traces and 46 of 50 no-write ones. Seeing only such lookups, the adversary answers "no write" on 78% of write-task traces and 76% of no-write ones: the lookups do not separate the two. Because an empty trace scores as a miss and a non-empty one does not, that default answer counts as recovery on the 25 no-write tasks (0.76 against Naive's 0.42), whereas under the original speculator most Shadow + reissue traces are empty (6 of 50 no-write traces carry a ghost call). The frequency-calibrated observer of Appendix D.3, which knows that "no write" is the most common banking label, gains nothing from these traces (-0.004 bits [-0.015, $+0.009$], $p = 0.54$) and 0.203 bits from Naive's ($p = 0.033$). Table 34 gives the paired Naive minus Shadow + reissue contrast for both speculators: on this metric, which we adopted after seeing the failure, the contrast is positive in every cell ($p \leq 0.03$). The same traces bound the SIA result: with

this speculator SIA and Shadow + reissue coincide (0.216 and 0.201 recovery; $+0.002$ and -0.004 bits), because this speculator's knowledge-base queries carry no intent, so SIA's read-only rule has nothing intent-bearing to let through. SIA's residual in Table 31 belongs to a speculator whose queries carry the intent.

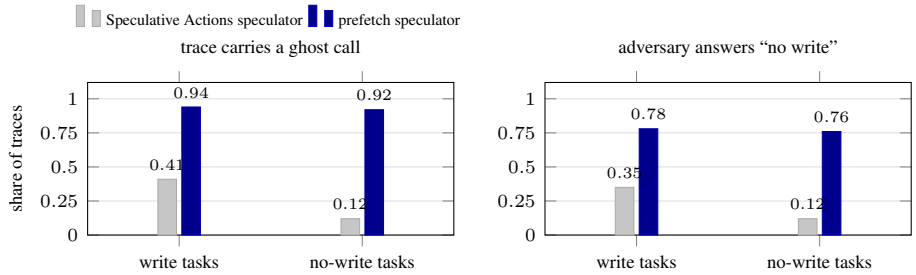

Figure 9: Why the all-task banking criterion fails for the prefetch speculator: Shadow + reissue traces on τ^2 banking (97 tasks, two trials; Opus 4.7 on the ghost-only projection). **Left:** with the prefetch speculator nearly every trace carries a benign account lookup, on write and no-write tasks alike. **Right:** seeing only such lookups, the adversary answers "no write" at the same rate on both, so the answer carries no information; it scores as recovery on the no-write tasks (0.76) because a non-empty trace is not scored as a miss. With the original speculator most no-write traces are empty (6 of 50 carry a ghost call).

Domain	Speculator	Naive gain	S+R gain	Naive − S+R [95% CI], p
Retail	Speculative Actions	$+0.365$	-0.009	$+0.374$ [$+0.196$, $+0.557$], 1×10^{-4}
	prefetch	$+0.947$	-0.005	$+0.951$ [$+0.592$, $+1.321$], 5×10^{-5}
Banking	Speculative Actions	$+0.668$	-0.000	$+0.669$ [$+0.389$, $+0.976$], 5×10^{-5}
	prefetch	$+0.203$	-0.004	$+0.207$ [$+0.032$, $+0.412$], 0.03

Table 34: Both speculators under the frequency-calibrated, task-held-out observer of Appendix D.3 (Opus 4.7 answers on the ghost-only projection; log-loss improvement over the label prior, bits per task). The last column is the paired per-task contrast with a task bootstrap and a sign-flip permutation test. Adopted after the all-task banking criterion of Table 33 failed, so it is a post hoc analysis.

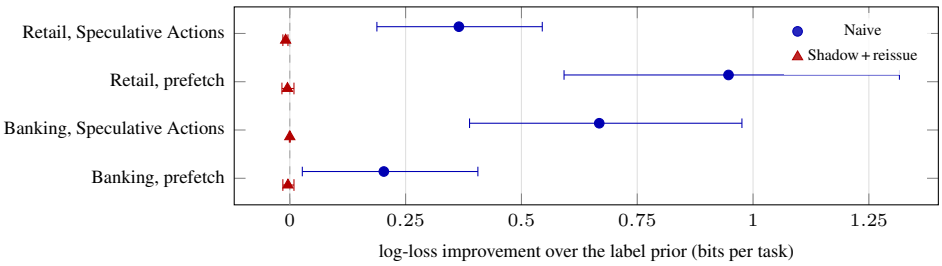

Figure 10: The frequency-calibrated observer's gain from each policy's ghost-only traces, with 95% task-bootstrap intervals, for both speculators (values as in Table 34). Every Shadow + reissue interval contains zero or lies below it; every Naive interval lies above zero.

D.5 Issued versus committed-only inference

§3.2's security goal compares $O_i(S)$ with $O_i(C)$. On τ^2, C is the committed trace of the non-speculative agent, which replay makes identical to each policy's committed calls, so $O(C)$ is No-Spec's trace of the same run. We place both on one timeline. A trace lists its committed calls in order and one speculation window per assistant reply after the greeting; window k is issued before user message $k{+}2$, and each of the 550 promoted branches at one guess is committed at turn $k{+}2$ or later (441 at $k{+}2$). The observer is the schema classifier behind our τ^2 numbers, which reads the write intent each provider-visible call implies; when it has seen no intent it falls back to the most common label, so the committed-only side gets full credit for knowing task frequencies. We score it before the agent commits its first intent-bearing write (over the whole run when it never commits one), at one to three user turns before that write, and at the end of the run.

Domain	b	tasks	before the first committed write: issued	committed	advantage [95% CI]	up/down, p	end of run: issued / committed
Retail	1	114	0.675	0.211	$+0.465$ [$+0.368$, $+0.557$]	63/4, 1.1×10^{-14}	0.711 / 0.763
Retail	2	114	0.680	0.211	$+0.469$ [$+0.373$, $+0.561$]	63/4, 1.1×10^{-14}	0.702 / 0.763
Retail	3	114	0.697	0.211	$+0.487$ [$+0.390$, $+0.579$]	66/4, 1.7×10^{-15}	0.706 / 0.763
Retail	5	114	0.693	0.211	$+0.482$ [$+0.386$, $+0.575$]	65/4, 3.1×10^{-15}	0.711 / 0.763
Airline	1	27	0.593	0.296	$+0.296$ [$+0.093$, $+0.500$]	13/5, 0.10	0.630 / 0.593
Airline	2	27	0.537	0.296	$+0.241$ [-0.019, $+0.481$]	12/5, 0.14	0.593 / 0.593

Table 35: Inference advantage of Naive's issued trace over the committed-only counterfactual (schema classifier with a constant-guess fallback; airline on its write-bearing tasks, as in Table 30). Advantages and CIs are task means of per-task differences, computed before rounding, with a task bootstrap and an exact sign test. SIA and Shadow + reissue equal the committed-only column in every row, before the write and at the end.

SIA and Shadow + reissue match the committed-only trace in every row and at every point of the timeline: on these domains the intent rides on the write tool, which SIA never speculates and the contract withholds, so §3.2's goal holds on this measure. Naive's advantage comes mostly from the turn before the write (Table 35): on the 107 retail tasks whose agent commits a write, recovery from the issued trace at one guess is 0.229, 0.355 and 0.701 two turns before, one turn before and just before it, against 0.215 for the committed-only trace. The advantage also covers the runs whose agent never commits a write (24 of 228 on retail, 9 of 54 on airline), where the issued trace identifies the intent on 0.375 of retail runs against 0.208: intents the committed plan never discloses. By the end of the run the committed write names the intent, and on retail Naive's issued trace scores below the committed-only one (0.711 against 0.763; 4 tasks up, 13 down, $p = 0.049$) because its ghost writes add misattributed intents to the set the classifier matches. Airline's 27 write tasks are too few to resolve the before-write advantage (neither row is significant) and show no end-of-run reversal: 0.630 issued against 0.593 committed-only at one guess, and 0.593 against 0.593 at two.

An adversary that reads the committed lookups. The schema classifier maps every retail read tool to no intent, so neither projection earns anything from the committed reads and the committed-only trace scores the constant guess. Those reads are not empty. The 228 No-Spec runs issue 1,780 committed calls, most of them order, product and user lookups carrying identifiers, names and zip codes. We therefore rescore the same two projections at the same cutoff with an adversary that reads arguments: a leave-one-task-out naive Bayes over tool names and argument text, which needs no model calls (`scripts/analyze_read_aware_advantage.py`). It raises the committed-only baseline from 0.211 to 0.368. Naive's advantage halves to $+0.211$ [$+0.123$, $+0.303$], still far from zero (32 tasks up, 7 down, $p = 7.0 \times 10^{-5}$). The contract's result does not move: SIA gains $+0.022$ ($p = 0.55$) and Shadow + reissue -0.004 ($p = 1$), so §3.2's goal holds against this observer as well. Under the schema classifier those two rows are zero mechanically, because it scores write-tool identity alone. Here they are measured against an observer that reads the lookups and still finds nothing. That observer is fit to the same deployment's traffic, so it bounds what a local observer extracts rather than a cold reader. We therefore repeat the measurement with the adversary the paper uses elsewhere: Opus 4.7, reading tool names and arguments, on the $b = 1$ traces of Table 35 at the same cutoff and over the same 24-label menu (`scripts/analyze_opus_issued_vs_committed.py`). It recovers 0.211 from the committed-only trace, the constant guess, and 0.693 from Naive's issued trace against the schema classifier's 0.675: an advantage of $+0.482$ [$+0.377$, $+0.588$], 71 tasks up and 11 down, $p = 6.8 \times 10^{-12}$. SIA gains -0.009 ($p = 0.63$) and Shadow + reissue -0.026 ($p = 0.21$). The stronger adversary extracts nothing from the committed lookups that the schema classifier missed, so the naive Bayes's higher baseline reflects lexical regularities in one deployment's traffic rather than intent a cold observer could read. The advantage the schema classifier measures is not an artifact of its blindness to reads.

E Related-Work Positioning

Table 36 positions speculative-issue-time privacy against the adjacent literatures discussed in §2. The combination this paper studies is absent from the work compared in Table 36.

Work	Spec. branch	Issue-time	Transforms proj.	Agent traces
Speculative-agent runtimes[a]	✓	—	—	✓
SIA (Hooper et al., 2026)	✓	partial	—	✓
Committed-call IFC[b]	—	partial	—	✓
Committed-leakage benchmarks[c]	—	—	—	✓
Query privacy and PIR[d]	—	—	partial	—
Side channels and fingerprinting[e]	partial[f]	—	—	—
This work	✓	✓	✓	✓

Table 36: Positioning across speculative-agent and agent-privacy work. *Spec. branch*: involves abandoned speculative branches. [f]Transient-execution and speculative-decoding attacks do exploit abandoned speculative paths, but at the microarchitectural and token level; no work in that line treats an agent's abandoned *tool calls* as the leaking object. *Issue-time*: decides policy before external dispatch (partial: SIA's read-only restriction is a coarse issue-time control with no transformation). *Transforms proj.*: changes the observer-visible projection rather than allow/block alone (partial: query-privacy mechanisms transform a query before it is issued). *Agent traces*: evaluation runs on LLM-agent tool-call traces. [a] (Sui et al., 2026; Ye et al., 2026; Guan et al., 2026; Hua et al., 2025; Nichols et al., 2025; Xia et al., 2026; Feng et al., 2026); [b] (Costa et al., 2025; Zhong et al., 2025; Cai et al., 2026; Mou et al., 2026; Doshi et al., 2026); [c] (Huang et al., 2026; El Yagoubi et al., 2026; Debenedetti et al., 2024; Ruan et al., 2024; Zhan et al., 2024; Yao et al., 2025); [d] (Chor et al., 1998; Howe & Nissenbaum, 2008; Zeng et al., 2024; Nissenbaum, 2004); [e] (Zhang et al., 2025; Wei et al., 2024; Kocher et al., 2019).

F REPRODUCIBILITY

Models and deployment dates. Primary models are Claude Opus 4.7 (planner on the generated corpora and primary adversary, deployed early 2026), Claude Haiku-4.5 (agent, labeler, speculator and user simulator on τ^2 and HotpotQA, planner and adversary on AgentDojo and the async workload, late 2025), and GPT-4o-mini (second speculator and cross-family adversary, mid-2024 release line). Replicate counts are per-corpus (Table 6); each trace is scored by one adversary pass, and every contrast is paired within that pass. 95% bootstrap CIs resample tasks, clustering repeated observations of the same task.

Decoding settings. Planner and adversary decoding use the default temperature and top-p for each model; we vary temperature only for policies that call for stochastic substitution (Rewrite paraphrase generation and Shadow internal-substitute generation), and we flag the variation where it occurs.

Adversary and planner prompts. The closed-set adversary receives the observer projection and the per-corpus candidate label set and chooses exactly one label. The open-set adversary returns a short free-form label scored against the ground truth by lexical Jaccard at threshold 0.5. The planner runs a candidate-generation phase that emits the speculative frontier under a branch-confidence estimate, then a commitment phase; both flow through the same contract-monitor dispatcher.

Task generation and label taxonomy. The 30-task and 150-task corpora are generated by sampling sensitive-topic templates, expanding each with an LLM, and deduplicating by lexical Jaccard. Each task carries a ground-truth label from a per-corpus list covering health, legal, employment, financial, security, and relationship topics. The AgentDojo subset uses its native categories.

Bootstrap and sign test. Bootstrap CIs use a fixed RNG seed throughout: 10,000 resamples at seed 20260522 for the generated-corpus aggregators, and 2,000 at seed 0 for the τ^2, AgentDojo and wall-clock family. Where tasks have replicates, the bootstrap resamples task clusters and retains all replicates in a selected cluster. Paired replay first computes the within-task mean of `correct(policy)−correct(Naive)`; the two-sided sign test then operates on the 150 task-level means, excluding ties. Holm correction controls family-wise error over the eleven policy comparisons.

Frozen frontier file schema. Each frozen frontier file is a tabular log keyed by (task identifier, replicate) whose payload records the planner's speculation candidates and the committed plan emitted

at the end of planning. The same row is replayed against every policy so adversary deltas reflect dispatcher choice rather than planner stochasticity.

Latencies. Static experiments use simulated per-tool latencies (web 200, retrieval 100, calendar 50, CRM 80, email 120 ms); Brave runs measure HTTP-boundary timing.

Artifact manifest. The anonymized supplementary artifact contains the prototype runtime (Python), every policy implementation referenced in the body, the rule-based labeler and the local probe, the task-generation scripts and the generated corpora, the closed-set and open-set adversary prompts, the multi-tool benchmark driver, the Typed-IFC baseline (Appendix C.18), the enterprise corpus and its endpoint adapters (Appendix C.19), the frozen frontiers and per-policy traces behind every table, and the recorded model responses, so that every adversary score can be recomputed without API calls. A reproduction guide maps each table and figure to the command that regenerates it, a script regenerates the per-corpus diagnostics, and the vendored third-party benchmarks are restored at their pinned commits by an included script.