# PLANTWIN: Privacy-Preserving Planning Abstractions for Cloud-Assisted LLM Agents

Guangsheng Yu, *IEEE Member*, Qin Wang, *IEEE Senior Member*,
Rui Lang, Shuai Su, and Xu Wang, *IEEE Member*,

***Abstract*—Cloud-hosted large language models (LLMs) have become the de facto planners in agentic systems, coordinating tools and guiding execution over local environments. In many deployments, however, the environment being planned over is private, containing source code, files, credentials, and metadata that cannot be exposed to the cloud. Existing solutions address adjacent concerns, such as execution isolation, access control, or confidential inference, but they do not control what cloud planners observe during planning: within the permitted scope, *raw environment state is still exposed*.**

**We introduce PLANTWIN, a schema-constrained projection-based architecture for cloud-assisted planning that prevents raw local context from leaving the local boundary. The key idea is to project the real environment into a *planning-oriented digital twin*: a schema-constrained and de-identified abstract graph that preserves planning-relevant structure while removing reconstructable details. The cloud planner operates solely on this sanitized twin through a bounded capability interface, while a local gatekeeper enforces safety policies and cumulative disclosure budgets. We further formalize the privacy–utility trade-off as a capability granularity problem, define architectural privacy goals using $(k, \delta)$-anonymity and $\epsilon$-unlinkability, and mitigate compositional leakage through multi-turn disclosure control. We implement PLANTWIN as middleware between local agents and cloud planners and evaluate it on 60 agentic tasks across ten domains with four cloud planners. PLANTWIN achieves SND $= 1.0$ against passive-observer adversaries, while maintaining planning quality close to full-context systems: three of four cloud planners achieve PQS $> 0.79$, within $\sim$4% of the no-privacy Raw Context baseline; the privacy-hardening pipeline stages add less than 2.2 percentage points of further PQS variation. Residual identifiability under stronger structural-fingerprint adversaries persists and is bounded by deployment-side controls rather than architecturally eliminated.**

***Index Terms*—LLM, Agents, Privacy, Planning, Information Disclosure, Cloud**

## I. INTRODUCTION

Large language models are increasingly used as planners in agentic systems, where a cloud-hosted model coordinates tools, decomposes tasks, and guides execution. Production systems such as Claude Code [1], OpenAI Codex [2], Cursor, and GitHub Copilot illustrate this design point: a powerful remote model reasons over code, files, and project context, while a local runtime executes the resulting plan. In many deployments, however, the environment being planned over is private. It contains source files, identifiers, logs, credentials, and sensitive metadata that cannot be exposed to the cloud. This tension is particularly relevant in service computing, where LLM planners are increasingly used to coordinate service workflows [3], service recommendation [4], cloud-native root cause analysis [5], and other services [6], [7], all while interacting with service composition and user-facing delivery logic. This creates a fundamental systems challenge: *how can a local edge agent benefit from cloud-scale planning intelligence without revealing the real underlying state?*

Current systems address this challenge only partially. Claude Code's sandbox [8] enforces filesystem and network isolation at the OS level, preventing access outside approved directories or unapproved servers. OpenAI Codex [2] executes tasks in an internet-disabled cloud sandbox confined to a preloaded repository snapshot. MCP [9] standardizes tool interfaces through schema-constrained descriptions and capability whitelisting. These mechanisms improve execution integrity and limit what the agent can reach. But within the approved scope, *the cloud planner still sees raw context*: file contents, code structure, and project metadata inside the permitted boundary are transmitted directly as prompt input. They control what the agent can *do*, and coarsely what it can *reach*; not the *abstraction level* of what the cloud planner observes once access is granted.

This exposes a missing design point. Existing approaches largely fall into four categories: (1) *full-context upload* [1], [2], where raw or lightly filtered context is sent to the cloud and privacy relies on trust and contractual guarantees; (2) *Personally Identifiable Information (PII) redaction* [10], [11], which removes recognized tokens before transmission but misses domain-specific secrets and leaves structural metadata that can still support re-identification; (3) *fully local inference* [12], [13], which avoids cloud exposure but is constrained by local resources and cannot match the planning capability of frontier models; and (4) *trusted execution environments (TEEs)* [14]–[16], which protect data during cloud inference but require specialized hardware and do not answer the semantic question of how much information the planner actually needs.

The central question is therefore not only how to secure execution, but how little information a cloud planner can observe while still producing useful plans. Our premise is that high-level orchestration rarely requires full descriptive fidelity of the underlying environment. For many workflows, useful planning depends on object types, relations, risk levels, action affordances, and dependency constraints, rather than raw file contents or precise identifiers. This suggests a different systems interface: the local environment should be exposed not

G. Yu, R. Lang, S. Su, and X. Wang are with University of Technology Sydney, Australia (e-mail: guangsheng.yu@uts.edu.au; yuluolout@gmail.com; isaacsue1996@gmail.com; xu.wang@uts.edu.au).

Q. Wang is with CSIRO Data61, Australia (e-mail: qinwangtech@gmail.com). Q. Wang is the corresponding author.

as a compressed replica of reality, but as a *planning-oriented control surface*.

We present PLANTWIN, a *schema-constrained projection-based digital twin*[1] architecture that realizes this idea. The local agent converts the real environment into a de-identified, schema-constrained twin through a four-stage pipeline ($\Pi_{\text{extract}} \to \Pi_{\text{redact}} \to \Pi_{\text{generalize}} \to \Pi_{\text{schema}}$). The cloud planner sees only this sanitized twin and outputs declarative plans over bounded local capabilities, while raw-data access, execution, and output sanitization stay local. A gatekeeper enforces disclosure budgets, safety checks, and optional human approval.

We complement recent capability-based control for LLM agents [17]–[19]. Those systems regulate *what actions the agent may execute*; we regulate *what information the planner may observe*. The two are orthogonal and naturally composable, combining execution-time access control with planning-time privacy control. From this perspective, the central design question becomes the *granularity of the capability interface*, which governs the privacy–utility trade-off. If capabilities are too coarse, most reasoning collapses back to the local side and the cloud provides limited planning benefit. If capabilities are too fine-grained, the planner regains utility only through repeated interaction, increasing cumulative disclosure. The desirable operating point lies near the *orchestration boundary*, where the cloud determines workflow logic and the edge performs content-sensitive inspection and enforcement [20].

**Contributions.** We make the following contributions:

- We formulate *cloud-assisted planning without raw-context exposure* as a system problem and position it against existing approaches including sandboxing, access control, PII redaction, and confidential computing (§II).
- We propose a four-stage local projection pipeline that converts the real environment into a sanitized, schema-constrained abstract graph. Unlike token-level masking, this design is *structurally bounded*: the cloud planner never receives free-form raw context (§IV).
- We introduce a declarative capability-based planning protocol and formalize the privacy-utility trade-off as a *granularity problem*, showing how capability design governs both planning effectiveness and disclosure cost (§IV-B).
- We design a local gatekeeper that enforces cumulative disclosure budgets during planning, and formalize two privacy goals for planner-visible abstractions: $(k, \delta)$-anonymity (abstractions do not uniquely identify underlying artifacts) and $\epsilon$-unlinkability (abstractions cannot be reliably linked across sessions) (§IV-D).
- We implement PLANTWIN as middleware between local agents and cloud planners and show, on 60 tasks across ten domains with nineteen method configurations and four cloud planners, that strong cloud planning can be retained without exposing raw local context (§V).

[1]The twin is intentionally *lossy*: it supports planning while resisting reconstruction of raw content. It does *not* provide architectural anonymity; structural identifiability is characterised in §V-D.

TABLE I: Privacy/security approaches for LLM agent systems.

| Approach | ① | ② | ③ | ④ | ⑤ |
|---|---|---|---|---|---|
| Claude Code Sandboxing [8] | ✓ | ✗ | ✗ | ✗ | ✓ |
| OpenAI Codex Sandbox [2] | ✓ | ✗ | ✗ | ✗ | ✓ |
| MCP Specification [9] | ✓ | ○ | ✓ | ✗ | ✓ |
| E2B / Firecracker VMs [21] | ✓ | ✗ | ✗ | ✗ | ✓ |
| Progent [17] | ✓ | ✗ | ✗ | ✗ | ✓ |
| MiniScope [18] | ✓ | ✗ | ✗ | ✗ | ✓ |
| SEAgent [19] | ✓ | ✗ | ✗ | ✗ | ✓ |
| PII Redaction (regex) [10] | ✗ | ○ | ✗ | ✗ | ✓ |
| TEE (H100 CC, Intel TDX) [14], [16] | ✓ | ✓ | ✗ | ✗ | ✗ |
| Local-only [12], [13] | ✓ | ✓ | N/A | N/A | ✓ |
| **PLANTWIN (Ours)** | ✓ | ✓ | ✓ | ✓ | ✓ |

① Controls agent actions, ② Controls planner observations
③ Schema-bounded context ④ Multi-turn disclosure
⑤ No specialized HW

## II. BACKGROUND AND RELATED WORK

Cloud-assisted agent systems typically separate *planning* from *execution*: a powerful remote model reasons over user context and produces plans, while a local runtime executes tool calls or generated code. This architecture motivated several independent lines of work, including execution sandboxing, capability and access control, privacy-preserving context handling, and agent security.

Table I summarizes the qualitative design space and highlights the gap addressed by PLANTWIN.

### A. Agent Architecture and Execution Isolation

Modern agentic coding tools have largely converged on a common architecture in which a cloud LLM reasons over code, files, and project context, while a local or containerized runtime executes the resulting plan. In this setting, the dominant systems concern has been *execution isolation*.

Claude Code [8] implements OS-level sandboxing using platform-specific primitives such as `sandbox-exec` on macOS and `seccomp` with `landlock` on Linux, providing filesystem and network isolation. OpenAI Codex [2] executes tasks in a cloud sandbox over a preloaded repository snapshot, with internet access disabled. E2B [21] provides Firecracker microVM-based sandboxes with fast startup, and Agent Sandbox for Kubernetes [22] relies on gVisor-based userland kernels with optional Kata Containers support.

These systems improve execution integrity and reduce blast radius once a plan is issued. However, their main protection boundary is *what the agent can do*, not *what the planner can observe*. Within the permitted scope, the cloud planner commonly sees raw files, code semantics, and project metadata.

### B. Capability Interfaces and Access Control

A second line of work focuses on *authorization*: what tools and resources an agent may access. Progent [17], MiniScope [18], and SEAgent [19] respectively explore tool-level privilege control, least-privilege authorization, and mandatory access control with information-flow constraints, while Li et al. [23] argue for richer policy models that capture the semantic ambiguity of human–LLM interaction. MCP [9] fits into this direction by standardizing schema-constrained tool descriptions and capability declarations, thereby providing a

partial *capability interface*. However, it does not constrain the *context* accompanying tool use: servers may still expose arbitrary raw resources, and planner-visible state remains largely unconstrained.

Recent studies show that schema-constrained tool interfaces alone do not prevent information leakage [24], [25]. Reported attacks include tool shadowing by malicious MCP servers [26], Log-To-Leak [27], which induces agents to invoke attacker-controlled logging tools, and MCP-SafetyBench [28], which reports substantial task compromise rates. Prior evidence further suggests that models may still infer or propagate sensitive state through contextual interactions and tool outputs [29]. API- or schema-level constraints are insufficient. More broadly, representation-level transformations can preserve functionality while still changing observable signals and enabling downstream inference, so abstraction alone does not guarantee observation privacy [30]. The distinction between controlling *actions* and controlling *observations* is explicit.

Our work targets the latter: it constrains what the planner sees before execution, which is orthogonal to and composable with execution-time access control, even when capabilities are exposed to the planner as high-level *skill*-like operations [31].

### C. Privacy-Preserving Context Mechanisms

A third line of work seeks to reduce privacy exposure in the context provided to LLMs [20].

**PII redaction.** Production systems such as Presidio [10] and Strands Agents [11] combine regex and NER to remove structured sensitive tokens. These approaches are useful for obvious PII, but they often miss domain-specific identifiers, secrets embedded in code or configuration files, and higher-order structural cues. They also typically operate on a per-request basis and do not track cumulative disclosure across turns.

**Differential privacy.** Differential privacy [32], [33] provides a principled way to bound information leakage through calibrated noise. However, applying DP directly to free-text prompts or rich project context remains difficult because of the high dimensionality and semantic sensitivity of natural language. In practice, DP is more naturally applied to training or aggregate analytics than to planning-time context exchange.

**Embedding leakage.** Prior work shows that learned representations can themselves leak sensitive information. Song et al. [34] study attribute leakage from embeddings, while Morris et al. [35] and Li et al. [36] demonstrate approximate text or sentence reconstruction. These results motivate our decision to keep retrieval and raw-context handling on the trusted local side rather than exposing vectorized surrogates to the cloud.

**Confidential computing.** TEE-based LLM inference [14]–[16] protects data during cloud computation through hardware-enforced isolation. This addresses data-in-use protection against the cloud infrastructure, but not the semantic question central to our setting: how much information the planner should observe in the first place.

**Local-only inference.** Systems such as PrivateGPT [12], Ollama [13], and related local-serving frameworks avoid cloud exposure entirely, but they also give up the planning capability of frontier cloud models, especially for long-horizon or multi-step reasoning tasks.

These approaches expose a missing design point. Existing work either keeps strong planning by revealing raw context, or preserves privacy by weakening or relocating the planner. Our goal is instead to retain strong remote planning while restricting the planner to a bounded, sanitized abstraction of the local environment.

### D. Agent Security Threats and Design Gap

A growing body of work documents the security risks of LLM agents. A comprehensive SoK [37] synthesizing 78 studies reports prompt injection success rates exceeding 85%. Hung et al. [38] analyze vulnerabilities specific to coding agents, including attacks mediated through tool ecosystems such as MCP. Real-world incidents include CVE-2025-53773 [39], a critical GitHub Copilot vulnerability, as well as prompt-injection attacks against cloud-based agent systems exploiting whitelisted integrations.

These attacks become more severe when the planner directly consumes raw local content. If arbitrary files, logs, and metadata enter the planning context, malicious instructions embedded in local artifacts can steer remote reasoning, and sensitive information may leak even when execution is later sandboxed. This motivates PLANTWIN: instead of relying only on execution isolation, access control, or token-level redaction, it constrains the *planner-visible observation surface* itself. The cloud planner operates on a schema-constrained digital twin rather than raw local artifacts, reducing both privacy exposure and the attack surface of planning over untrusted content.

## III. SYSTEM MODEL

We define the system boundary, threat model, and formal objective of PLANTWIN. Table VIII summarizes notations used throughout.

### A. System Overview

PLANTWIN follows a split architecture in which planning is delegated to a strong cloud model, while raw-state access and enforcement remain local. As shown in Fig. 1, the system consists of two asymmetric components.

- *Trusted local layer.* The local side has direct access to the real environment and is responsible for privacy projection, policy enforcement, execution, and output sanitization. It may use deterministic scanners and optionally a small local language model to extract planning-relevant structure from raw artifacts.
- *Untrusted cloud planner.* The cloud side performs high-level reasoning and plan synthesis, but never accesses raw files, paths, credentials, or unsanitized execution outputs. Its view is restricted to the sanitized digital twin and the exposed capability interface.

The end-to-end information flow is

$$X_t \xrightarrow{\Pi} Z_t \xrightarrow{\Phi} P_t \xrightarrow{\Psi} Y_t, \tag{1}$$

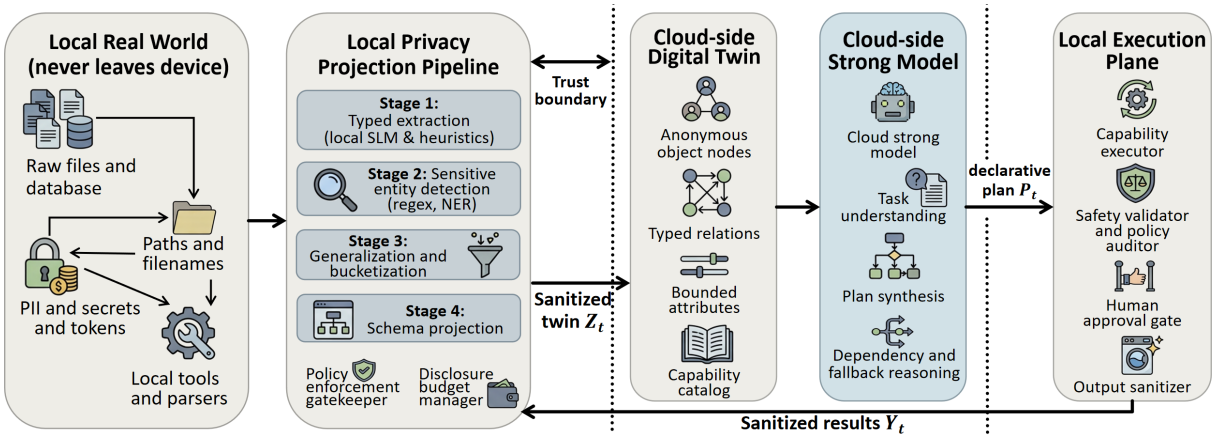


Fig. 1: End-to-end architecture of PLANTWIN. The trusted local edge layer (left) transforms raw context $X_t$ through a four-stage privacy projection pipeline into a sanitized digital twin $Z_t$, which crosses the trust boundary to the cloud planner (center-right). The planner reasons over the typed abstract graph and capability catalog to produce a declarative plan $P_t$. The trusted local execution plane (right) validates each step through a gatekeeper, executes on raw data, and sanitizes outputs via $\Pi_{\text{out}}$ before returning results $Y_t$. The cloud never observes raw files, paths, credentials, or code.

where $X_t$ denotes the real local environment, $\Pi$ the privacy projection pipeline, $Z_t$ the planner-visible digital twin, $\Phi$ cloud-side planning, $P_t$ the declarative plan emitted by the cloud planner, $\Psi$ local validation and execution, and $Y_t$ the sanitized output returned after local execution. This decomposition captures the central design choice of PLANTWIN: the cloud plans over an abstract view, while all interaction with the real environment remains on the trusted local side. In this way, the cloud contributes planning capability without controlling what is revealed, executed, or returned.

### B. Threat Model

We treat the cloud planner as untrusted with respect to privacy. Our primary model is *honest-but-curious*: the planner follows the protocol for producing plans, but may attempt to infer sensitive information from everything it observes. We also consider a stronger *semi-honest with auxiliary knowledge* adversary that may correlate multiple observations and combine them with external information.

Concretely, the adversary may observe all sanitized twin states $Z_{1:T}$ across turns, the sequence of requested capabilities and their abstract outputs, and timing or structural patterns arising during planning and execution. We also allow the adversary to combine these observations with auxiliary knowledge, such as public repositories, known project templates, or likely technology stacks.

The adversary does *not* directly observe raw local files, metadata, secrets, or unsanitized outputs. The local layer, including the projection pipeline, gatekeeper, and output sanitizer, is part of the trusted computing base.

**Active capability-sequencing adversary.** Beyond passive observation, we also consider an active adversary that issues a sequence of benign-looking capability requests across turns to harvest `semantic_class`, `contains:*`, and `usable_for:*` tags about a target object. This adversary cannot expand the schema or invoke unauthorized capabilities (both are gated locally) but can adaptively choose which approved capability to request next based on prior responses. PLANTWIN partially defends through the per-object disclosure-budget mechanism (§IV-D) and a per-turn capability rate limit: once an object's budget is exhausted, the gatekeeper denies further capability requests targeting it (Theorem 3, multi-turn cumulative-disclosure bound). However, an adversary that stays under-budget per object but distributes attention across many objects can still aggregate distinguishing signal across turns, as quantified by the re-identification analysis in §V-D. We therefore characterize the defense as *bounded* rather than *complete*.

**Out of scope.** We do not address adversaries that (i) compromise the local trusted computing base, (ii) hold strong prior auxiliary knowledge of the target population beyond what is captured in §V-D, or (iii) exfiltrate via side channels outside the capability interface (e.g., timing, payload size). These are characterized in Appendix C.

**Adversary goals.** We focus on three privacy goals that capture the main ways a planner-visible abstraction may still leak information.

*Goal 1: Identity re-identification.* Given transcript $Z_{1:T}$ and a target object $o^*$, an adversary outputs a guess $\hat{o}$ for the real-world identity $o^*$. We capture resistance to this attack via $(k, \delta)$-anonymity:

$$\Pr[\mathcal{A}(Z_{1:T}) = o^*] \leq \frac{1}{k} + \delta, \tag{2}$$

where $k$ is the anonymity set size induced by schema projection. Appendix B formalizes the bound under assumptions on attribute distribution and disclosure budgeting.

*Goal 2: Cross-session linkage.* Given transcripts from two sessions $Z^{(1)}_{1:T_1}$ and $Z^{(2)}_{1:T_2}$, the adversary determines whether the same real object appears in both. We capture resistance through $\epsilon$-unlinkability:

$$\ln \frac{\Pr[\mathcal{A}(Z^{(1)}, Z^{(2)}) = 1 \mid \text{same}]}{\Pr[\mathcal{A}(Z^{(1)}, Z^{(2)}) = 1 \mid \text{diff}]} \leq \epsilon. \tag{3}$$

where $\mathcal{A}$ is a linkage adversary, superscripts $(1)$ and $(2)$ index the two sessions, and same (resp. diff) denotes the event that both transcripts contain the same (resp. distinct) real-world object. Fresh random object identifiers, bounded vocabularies, and cumulative disclosure limits are intended to keep this quantity small. Appendix B connects unlinkability bound to disclosure budget.

*Goal 3: Structural inference.* Even when content and identities are hidden, the adversary may infer sensitive properties such as project type, stack composition, or organizational structure from graph topology and capability patterns. This form of leakage is harder to characterize formally because structure itself is informative. We mitigate it through bounded edge vocabularies and controlled relational disclosure, but do not assume it can be eliminated entirely.

**Auxiliary knowledge.** A realistic adversary may already know partial facts about the target environment. For example, knowing that the target is likely a Django project may make schema attributes such as `semantic_class = authentication_module` more identifying. Our goal is therefore not privacy against an omniscient adversary, but bounded exposure under realistic auxiliary knowledge and limited observation.

### C. Problem Formulation

At time $t$, let $X_t$ denote the full local environment, including raw artifacts, metadata, logs, sensitive values, and interaction history. The local side does not expose $X_t$ directly. Instead, it applies the privacy projection pipeline $\Pi$ to construct a sanitized planner-visible state $Z_t = \Pi(X_t)$. Given $Z_t$ and capability catalog $\mathcal{C}$, the cloud planner produces a declarative plan $P_t = \Phi(Z_t, \mathcal{C})$, which the trusted local side then validates and executes under policy set $\Omega$ to produce the final outcome $Y_t = \Psi(P_t, X_t, \Omega)$.

The design objective is to retain as much planning utility as possible while keeping cumulative disclosure bounded:

$$\max\ \mathbb{U}(P_t, Y_t) \quad \text{subject to} \quad \mathcal{L}(Z_{1:t}, Y_{1:t}) \le \epsilon. \tag{4}$$

Here $\mathbb{U}$ denotes task utility and $\mathcal{L}$ measures cumulative exposure. In our design, disclosure is tracked per object through a budget $B(o)$, and $\mathcal{L}$ is conservatively taken as the maximum disclosure over all referenced objects, since over-exposing even a single object may already suffice for re-identification.

This formulation captures the core trade-off in PLANTWIN: the planner should observe enough structure to generate useful plans, but not enough detail to reconstruct, identify, or reliably link the underlying local artifacts.

## IV. OUR APPROACH: PLANTWIN

PLANTWIN is built around a simple idea: the cloud should help with *planning*, but it should never need to observe the real local environment directly. To make this possible, the system combines three mechanisms. First, the local side converts raw context into a sanitized digital twin through a four-stage projection pipeline. Second, the cloud planner operates only through a bounded capability interface, so planning is expressed as declarative requests over abstract objects rather than direct access to local artifacts. Third, a local gatekeeper validates every step, enforces policy, and meters cumulative disclosure across turns.

These mechanisms separate *reasoning* from *exposure*. The cloud contributes planning intelligence over an abstract view, while the trusted local side retains control over raw-state access, execution, and what information is ultimately released.

### A. Privacy-Preserving Planning Abstraction

The first component of PLANTWIN is a local projection pipeline that transforms the real environment into a planning-sufficient but reconstruction-resistant twin. Formally,

$$\Pi = \Pi_{\text{schema}} \circ \Pi_{\text{generalize}} \circ \Pi_{\text{redact}} \circ \Pi_{\text{extract}}, \tag{5}$$

that is, the local side first extracts structure, then redacts sensitive entities, then generalizes precise values, and finally projects the result into a fixed schema.

**Pipeline.** The role of this pipeline is not to summarize the environment as accurately as possible. Instead, it exposes only the structure needed for planning while suppressing content that would support reconstruction or re-identification.

*Stage 1: Typed extraction.* The local extractor identifies object modality, coarse semantic role, and task affordances using a lightweight on-device small language model (SLM) with deterministic heuristics. In our prototype, the SLM is Qwen3.5-2B-Instruct (≈2.0 B parameters, ≈4 GB FP16 / ≈2 GB INT8), which fits on a single consumer GPU (e.g., NVIDIA RTX 3060 / A2000 or an Apple Silicon device with 16 GB unified memory). When no accelerator is available, the prototype falls back to the deterministic heuristic extractor (sub-millisecond per object, no model dependency). The two paths produce the same downstream typed-graph schema, so all gatekeeper, budget, and capability semantics are identical. Objects are mapped to coarse kinds such as `code_file`, `document`, `log_stream`, `structured_record`, `config`, or `secret_container`. In parallel, the extractor assigns usability tags such as `comparable`, `summarizable`, `constraint_source`, `requires_local_tool` and `schema_bearing`. This stage determines what the object can be used for in planning, without yet deciding what may safely leave the trusted boundary.

*Stage 2: Sensitive entity detection and redaction.* The second stage removes or masks obvious identifiers and secrets before any higher-level abstraction is exposed. This includes names, email addresses, phone numbers, URLs, IPs, hostnames, paths, repository identifiers, ticket IDs, keys, tokens, and other secrets. We use a deterministic detector combining regex rules for well-structured entities with keyword-based sensitivity classification for more contextual identifiers such as git metadata or internal hostnames.

*Stage 3: Generalization and bucketization.* To reduce inversion risk, exact values are replaced by coarse categories. For example, byte sizes are mapped to {small, medium, large}, timestamps to {today, recent, stale}, counts to {one, few, many}, and sensitivity to {low, restricted, high}. This preserves planning-relevant distinctions while suppressing details that are unnecessarily identifying.

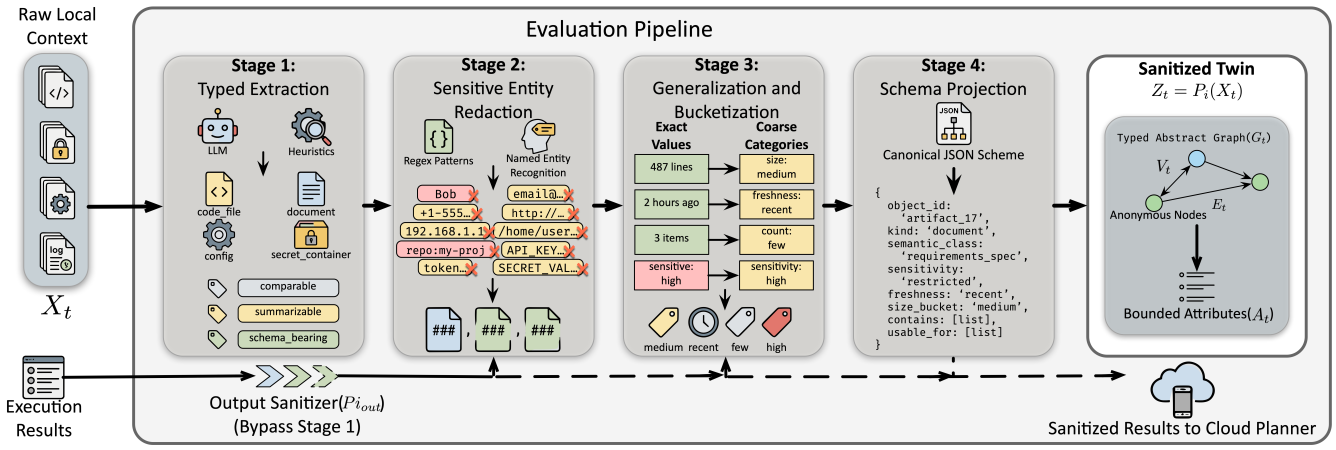


Fig. 2: The 4-stage privacy projection pipeline. Raw local context enters Stage 1 (typed extraction via local small language model (SLM) and heuristics), proceeds through Stage 2 (sensitive entity redaction via deterministic regex patterns with code-specific rules), Stage 3 (value generalization into bounded buckets), and Stage 4 (schema projection into fixed JSON). The output is a planning-sufficient digital twin $Z_t$ that contains no raw text, filenames, paths, or credentials. A companion output sanitizer $\Pi_{\text{out}}$ applies Stages 2–4 to execution results before cloud exposure.

*Stage 4*: Schema projection. Finally, the processed object is projected into a fixed JSON schema. A typical object takes the form:

```
{"object_id": "artifact_17",
 "kind": "document",
 "semantic_class": "requirements_spec",
 "sensitivity": "restricted",
 "freshness": "recent",
 "size_bucket": "medium",
 "contains": ["tables","dates","constraints"],
 "usable_for": ["compare","extract_constraints
     "],
 "confidence": 0.86}
```

This schema contains no filename, path, raw quote, or free-form text. The planner can reason over object roles and affordances, but not inspect the underlying content directly.

*Output sanitization.* The same principle also applies to execution results. We therefore define a companion output sanitizer $\Pi_{\text{out}}$ that applies Stages 2–4 to local outputs before they are returned to the cloud. Stage 1 is omitted because result types are already known from the capability specification.

*Security implication.* Because raw local artifacts never enter the cloud-visible channel, the design reduces the direct prompt-injection surface compared with raw-context systems. A residual risk is that malicious local content may still affect Stage 1 extraction by causing the local SLM to misclassify objects. We mitigate this with deterministic cross-checks and by routing low-confidence extractions for review.

**Planning abstraction as a typed graph.** On the cloud side, the planner-visible abstraction is represented as a typed graph $G_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{A}_t)$. Nodes in $\mathcal{V}_t$ correspond to anonymous artifacts (e.g., `artifact_i`), edges in $\mathcal{E}_t$ encode coarse relations such as dependency, derivation, similarity, blocking, capability requirement, and conflict, and attributes in $\mathcal{A}_t$ capture bounded properties such as sensitivity, freshness, readiness, confidence, and usability tags. The output of $\Pi(X_t)$ is concretely this graph representation.

This graph view is important because the planner does not reason over isolated objects. It reasons over object roles, dependencies, and action affordances. The abstraction therefore preserves the relational structure needed for orchestration while withholding the raw artifacts from which that structure was derived.

**Running examples.** Consider a coding-assistant task in which a developer asks: *"Review the authentication module for security issues and compare it with the payment module."*

In local environments, $X_t$ contains files such as `auth_service.py` and `payment_handler.py`, a `.env` file with live API keys, and git history containing developer identities. After projection, the twin $Z_t$ exposes only two code-file nodes with semantic classes such as "authentication module" and "payment module," with a secret-container node and a small number of typed dependency edges. No filenames beyond these abstract roles, no developer identities, no API keys, and no git history cross the boundary.

The cloud planner therefore reasons over the abstract structure rather than the raw files. It may request a security audit of the two modules, compare the resulting findings, and generate an anonymized summary of the differences. These requests are then checked and executed locally on the real files.

### B. Capability-Based Planning

Once the planning abstraction has been constructed, the cloud planner no longer interacts with the local environment directly. Instead, it operates through a restricted capability interface $\mathcal{C} = \{c_1, \ldots, c_n\}$, where each capability is a safe typed operation such as constraint extraction, object comparison, issue classification, or security auditing. In this way, the

TABLE II: Capability granularity trade-off.

| Granularity | Example | Utility ↑ | Privacy Cost ↓ |
|---|---|---|---|
| Too coarse | `process_project` | Low | Low |
| Orchestration | `security_audit(art_1)` | High | Medium |
| Too fine | `read_line(art_1, 42)` | High | High |

planner reasons over abstract objects rather than raw files or unconstrained tool access.

The planner output is therefore not arbitrary code, but a structured action graph,

$$P_t = \{(s_i, c_i, I_i, O_i, \rho_i)\}_{i=1}^{K}, \tag{6}$$

where $s_i$ is a step identifier, $c_i$ a chosen capability, $I_i$ abstract inputs, $O_i$ the expected abstract output, and $\rho_i$ a policy annotation. This keeps planning declarative: the cloud specifies *what operation should be performed*, while the trusted local side retains control over *how* it is executed on raw data.

A central design choice is the granularity of this capability interface. If capabilities are too coarse, the local side must perform most of the reasoning itself and the cloud adds little value. If they are too fine-grained, the planner can recover utility only through repeated interaction, which increases cumulative disclosure. The privacy-utility trade-off is therefore governed not only by what capabilities exist, but by how much abstraction each capability exposes.

We capture this trade-off through grain($\mathcal{C}$), the expected number of weighted schema fields revealed per capability call, using the disclosure model introduced in §IV-D. Let $\mathbb{U}(\mathcal{C})$ denote planning utility and $D(\mathcal{C}, T)$ cumulative disclosure over $T$ turns. The resulting design objective is

$$\mathcal{C}^* = \arg\max_{\mathcal{C}} \mathbb{U}(\mathcal{C}) \quad \text{s.t.} \quad D(\mathcal{C}, T) \leq \epsilon. \tag{7}$$

In practice, the useful operating point lies near what we call the *orchestration boundary*: capabilities should expose the workflow-level action to perform, but should not become a low-level channel for content inspection. Table II illustrates this intuition.

**Worked examples.** We illustrate the boundary placement with two capabilities from PLANTWIN's eight-operation vocabulary, each walking through coarser and finer variants.

*(i)* `compare_structured_objects` accepts two typed objects and returns a structured diff: which fields differ, which match, and a coarse equivalence-class summary. It sits intentionally at the orchestration boundary because the planner needs to know whether two artifacts diverge to decide if reconciliation is required, but does not need access to the differing values themselves. A coarser variant that merges with `validate_schema` into a single boolean `is_consistent_with` would zero out structural signal, but loses localisation: every "no" forces an additional diagnostic call, increasing turn count and cumulative disclosure. The multi-turn budget then trades per-call privacy for a longer dialogue, often a net loss. A finer variant that splits into `compare_field_values` plus `compare_structure` appears Pareto-optimal in principle, but in practice the planner cannot tell ex ante which it needs and tends to call both.

TABLE III: Schema field disclosure costs.

| Field | $w_f$ | Rationale |
|---|---|---|
| `kind` | 0.5 | Low entropy, many objects share types |
| `semantic_class` | 2.0 | Higher specificity |
| `contains` (per tag) | 1.0 | Content indicators |
| `usable_for` (per tag) | 0.5 | Functional, less identifying |
| Typed edge (per edge) | 3.0 | Relational structure is highly identifying |
| `freshness` | 0.5 | Temporal bucket, coarse |
| `sensitivity` | 1.0 | Policy-relevant |

The finer interface expands the surface area for capability-sequencing attacks (§III-B) without an offsetting utility gain. A small pilot (one capability, two variants, five tasks per variant) finds the fine variant moves PQS by $+0.6$pp while NL increases by 12pp, i.e., privacy strictly worsens.

*(ii)* `summarize_anonymized_diffs` accepts a stream of structural events (config edits, log entries, version-control commits) and returns a planner-readable summary with literal values redacted. It sits at the boundary because the planner needs the *shape* of recent change activity to plan triage or rollback, but not the contents. Merging with `extract_constraints` into `extract_actionable_state` collapses two disjoint disclosure budgets into one and weakens auditability: a bounded named capability is easier to gate than a multi-purpose one. Splitting per source type (`summarize_diffs:logs` / `:vcs` / `:configs`) is a defensible refinement for deployments where one source dominates risk, but is deployment-specific tuning that is not covered on synthetic benchmarks.

**General principle.** A capability sits near the orchestration boundary when (i) merging it with peers loses *named* semantics that policy authors depend on, and (ii) splitting it expands the capability-sequencing attack surface without enabling a use case the planner cannot already approximate by chaining existing capabilities. The eight-capability vocabulary in this section was designed against both criteria.

### C. Local Policy Enforcement

Cloud-generated plans are never executed directly. Every requested step is first mediated by a trusted local gatekeeper, which connects planning-time abstraction with execution-time enforcement. Its purpose is to ensure that each requested operation is valid for the referenced objects, consistent with local policy, and safe to execute on the real environment.

Operationally, the gatekeeper performs six checks in sequence: capability validation, policy validation, safety bounding, local execution, output sanitization, and escalation to a human when a step is ambiguous or high-risk The full procedure is summarized in Algorithms 1–2.

Formally, the local outcome is

$$Y_t = \begin{cases} \Pi_{\text{out}}(\text{Exec}(P_t, X_t)) & \text{if policy-compliant,} \\ \perp_{\text{reject}} & \text{if policy-violating,} \\ \perp_{\text{escalate}} & \text{if ambiguous or high-risk.} \end{cases} \tag{8}$$

This boundary ensures that cloud planning remains advisory rather than authoritative. The cloud may propose an operation, but admissibility, execution over raw data, and any returned

**Algorithm 1** Gatekeeper Validation and Execution

**Require:** Plan step $(s_i, c_i, I_i, O_i, \rho_i)$, context $X_t$, policy $\Omega$, budgets $\{B(o)\}$
**Ensure:** Sanitized result $y_i$ or rejection signal
**if** $c_i \notin$ Allowlist(type($I_i$)) **then**
  **return** $\perp_{\text{reject}}$ {Capability not allowed}
**end if**
**for** each object $o \in I_i$ **do**
  $\Delta \leftarrow$ EstimateCost$(c_i, o)$ {Weighted field cost}
  **if** $B(o) + \Delta > \tau(o)$ **then**
    **return** $\perp_{\text{reject}}$ {Budget exceeded}
  **end if**
**end for**
**if** $\rho_i$ requires `human_approval` **then**
  approved $\leftarrow$ HumanGate$(s_i)$
  **if** ¬approved **then**
    **return** $\perp_{\text{escalate}}$
  **end if**
**end if**
$r_i \leftarrow$ Execute$(c_i, I_i, X_t)$ {Local execution on raw data}
$y_i \leftarrow \Pi_{\text{out}}(r_i)$ {Sanitize output}
**for** each object $o \in I_i$ **do**
  $B(o) \leftarrow B(o) + \Delta_t(o)$ {Update budget}
**end for**
**return** $y_i$

**Algorithm 2** PLANTWIN: End-to-End Planning Cycle

**Require:** Context $X_t$, projection $\Pi$, capabilities $\mathcal{C}$, policy $\Omega$, planner $\Phi$, thresholds $\{\tau(o)\}$
**Ensure:** Sanitized result $Y_t$
Initialize $B(o) \leftarrow 0$ for all tracked objects $o$
**for** each task request or planning cycle $t$ **do**
  $X_t \leftarrow$ ObserveLocalState()
  $Z_t \leftarrow \Pi(X_t)$ {4-stage projection: extract → redact → generalize → schema}
  **for** each object $o$ in $Z_t$ **do**
    **if** $B(o) + \Delta(o) > \tau(o)$ **then**
      Suppress new fields or refuse disclosure
    **end if**
  **end for**
  Send $Z_t, \mathcal{C}$ to cloud planner
  $P_t \leftarrow \Phi(Z_t, \mathcal{C})$ {Receive declarative plan}
  **for** each step $(s_i, c_i, I_i, O_i, \rho_i)$ in $P_t$ **do**
    $y_i \leftarrow$ GatekeeperValidateAndExecute$(s_i, X_t, \Omega)$ {Algorithm 1}
    **if** $y_i = \perp_{\text{reject}}$ **or** $y_i = \perp_{\text{escalate}}$ **then**
      Handle rejection / escalation
    **end if**
  **end for**
  $Y_t \leftarrow$ AggregateResults$(\{y_i\})$
  Log disclosure event to audit trail
**end for**
**return** $Y_t$

abstract result are all determined on the trusted local side. If a request is rejected or escalated, no new information is released to the cloud and $\Delta_t(o) = 0$ for all involved objects

This same boundary also limits plan-injection attempts. The cloud may request operations, but it cannot directly force raw-state disclosure or bypass local validation. Per-step controls such as capability whitelisting, output sanitization, and optional human approval bound what any single request may reveal, while cumulative controls such as disclosure budgeting and rate limits constrain what can be learned over repeated interaction.

### D. Multi-Turn Disclosure Control

Single-turn redaction is not sufficient in an interactive planning setting. Even when each individual response appears harmless, repeated queries may gradually reveal a distinctive fingerprint of an object. PLANTWIN therefore treats disclosure as cumulative and tracks it across turns at the object level.

For each object $o$, cumulative disclosure is defined as

$$B(o) = \sum_{t=1}^{T} \Delta_t(o), \tag{9}$$

where $\Delta_t(o)$ is the newly disclosed information at turn $t$. The gatekeeper checks every proposed release against an object-specific threshold $\tau(o)$ and blocks additional disclosure once the threshold would be exceeded.

**Weighted field-cost model.** We instantiate this mechanism with a simple weighted field-cost model in which each schema field $f$ is assigned a base disclosure weight $w_f$ reflecting its re-identification potential. Table III lists the default costs used in our prototype.

At turn $t$, the newly incurred disclosure is

$$\Delta_t(o) = \sum_{f \in \text{newly disclosed}} w_f. \tag{10}$$

A new release is permitted only if

$$B(o) + \Delta_{T+1}(o) \le \tau(o), \tag{11}$$

where $\tau(o)$ may vary across object classes and sensitivity levels.

This mechanism is simple and operational. It is not presented as a full information-theoretic guarantee, but as a concrete local rule for limiting cumulative exposure in an interactive system.

*Advanced disclosure tracking.* The weighted budget is only one possible instantiation. Stronger variants could quantify $\Delta_t(o)$ through entropy reduction, online $k$-anonymity monitoring, or differential privacy composition when individual disclosures satisfy suitable privacy guarantees. We mention these alternatives to clarify that the budgeting framework is general, even though the present implementation adopts a lightweight field-cost model.

**Local retrieval and embedding exposure.** All retrieval and similarity computation remain local. The cloud sees only abstract match outcomes or anonymized object identifiers, avoiding the embedding leakage and inversion risks demonstrated in prior work [34]–[36].

## V. EVALUATION

### A. Experimental Settings

**Prototype.** We implement PLANTWIN as a Python middleware between a local agent runtime and a cloud LLM API. The system enforces a strict split between local projection and remote plan synthesis. By default, Stage 1 uses deterministic heuristic extraction for robustness and low latency, while a local Qwen3.5-family SLM [40] can optionally be used for richer semantic classification. Stage 2 uses regex-based entity detection with custom rules for code-specific secrets and identifiers, and Stages 3–4 are deterministic transformations. The

disclosure budget manager maintains per-object state with per-field deduplication. Unless otherwise stated, all experiments use the default heuristic extraction pipeline.

**Cloud planners.** We evaluate four frontier cloud planners through OpenRouter, using the sanitized twin as the only planner-visible context: *Kimi K2.5* [41], *Gemini 3 Flash*, *MiniMax M2.5* [42], and *GLM 5* [43]. All planners are configured with high reasoning effort for consistency. All local processing, including projection and gatekeeper, runs on a single machine without GPU requirements.

**Planner selection rationale.** The four planners were chosen for: (i) availability of an explicit reasoning/thinking mode, which most directly tests the planning-quality trade-off PLANTWIN targets; (ii) consistent API availability and rate limits during the evaluation window from our institutional region; and (iii) cost and API-quota constraints (each task triggers up to 50 turns; full benchmark cost is non-trivial under metered access). Three of the four (Kimi, MiniMax, GLM) are released by China-based labs, reflecting the current concentration of available reasoning-mode planners in that ecosystem at the time of evaluation; Gemini 3 Flash provides cross-vendor coverage. We expect, but did not empirically validate, that the same SND, PQS, and budget properties transfer to Western planners (e.g., GPT-4-class, Claude-3.5-class); the prototype's OpenRouter integration makes this a configuration change rather than a re-implementation.

**Benchmark construction.** The 60 evaluation tasks are template-based and team-authored, instantiating three real-deployment workflow patterns: (i) *artifact-mediated triage* (e.g., code review, log triage, document review), (ii) *cross-source reconciliation* (e.g., config drift, schema migration, ETL pipeline validation), and (iii) *constraint extraction and verification* (e.g., compliance review, policy enforcement). Each task seeds 3–8 sensitive items per artifact: PII tokens (emails, account names, IDs), credential strings (`sk-...`, hashed passwords, env-file secrets), system identifiers (internal hostnames, IP ranges, ports), and structured alerts (SIEM-style with `case_owner`, `ticket_id`). Templates were drawn from the authors' direct experience with the corresponding production workflows and cross-referenced against publicly documented agent benchmarks; no task is copied verbatim from a public benchmark. The 14-task capability-coverage subset (§V-B) consists of the first 14 tasks (coding, document, debugging); the 58-task ablation subset is the canonical 60 minus `pipeline_quality_2` and `pipeline_quality_3` (added to the canonical set after the ablation runs were complete). Real-deployment aspects we do not model are enumerated in Appendix C: multi-user concurrency, long-horizon memory, human-in-the-loop refinement, adversarial provenance, and real human users.

**Tasks.** We construct a benchmark of 60 synthetic agentic tasks spanning ten domains: coding assistant, document review, multi-step debugging, DevOps, data pipelines, security incident response, ML operations, frontend development, database administration, and API integration. These tasks cover realistic agent workflows and embed sensitive artifacts such as PII, API keys, tokens, connection strings, internal hostnames, and infrastructure secrets.

**Baselines.** We compare PLANTWIN against four baseline families. (i) *Raw Context* sends the full local environment to the cloud planner and serves as a utility upper bound without privacy protection. (ii) *PII Redaction* applies regex-based redaction before sending context to the cloud, but does not track cumulative disclosure across turns. (iii) *Local-Only* performs planning entirely with a local SLM without cloud assistance; we evaluate six models spanning three families and four scales: SmolLM2-360M-Instruct, Qwen3.5-0.8B, Qwen3.5-2B, Qwen3.5-4B, Qwen3.5-9B (with thinking mode), and Gemma-3-1B-it (instruction-tuned, no thinking mode). (iv) *Heuristic (Oracle)* uses manually curated keyword-to-capability rules without any cloud LLM, isolating the disclosure-budget mechanism from cloud planning quality. Combined with four cloud planners for PLANTWIN, PII Redaction, and Raw Context, these baselines yield 19 method configurations in total.

**PLANTWIN configuration.** Unless otherwise noted, PLANTWIN uses heuristic Stage 1 extraction, capability-aware lazy disclosure, and a *skill prompt* that teaches the cloud planner the twin's data format and capability semantics. We instantiate this configuration with each of the four cloud planners above (all with thinking mode enabled), yielding a total of 19 method configurations (4 planners $\times$ 3 modes + 6 local-only models + 1 heuristic) to systematically evaluate the privacy–utility–latency trade-off space.

**Metrics.** We report four metrics. (i) *Plan quality score (PQS)* $\uparrow$ is the task-averaged $F_1$ between approved and expected capabilities, capturing both over-generation and missing capabilities; we also report precision separately in the main results table. (ii) *Sensitive non-disclosure (SND)* $\uparrow$ measures the fraction of sensitive items in the raw environment that do not appear in the planner-visible payload. (iii) *Normalized leakage (NL)* $\downarrow$ measures the fraction of the disclosure budget consumed; for raw-context methods, we set NL$=1.0$ by convention. (iv) *Latency overhead* measures the additional per-task cost introduced by projection and gatekeeper validation.

### B. Planning Utility and Privacy

Table IV presents the main comparison across all nineteen configurations on the full 60-task benchmark.

PLANTWIN achieves SND$=1.0$ across all planners and domains: the projection pipeline removes raw content architecturally before cloud exposure. With Kimi K2.5, PLANTWIN reaches PQS$=0.808$, compared with $0.843$ for Raw Context; a paired Wilcoxon signed-rank test on 58 common tasks finds no significant difference (Bonferroni $p = 0.555$, $r = 0.230$). Three of four planners (Kimi $0.808$, MiniMax $0.792$, GLM $0.810$) are statistically indistinguishable from Raw Context ($p_{\text{Bonf}} = 1.0$). Gemini 3 Flash achieves lower PQS ($0.689$) but $2.6\times$ lower latency, reflecting a quality–latency trade-off. PII redaction yields the highest PQS ($0.887$) but leaks 16.5% of sensitive items (SND$=0.835$), with coding tasks dropping to SND$=0.12$.

TABLE IV: Main results across ten task domains (60-task evaluation per method)

| Planner | Method | Per-Domain PQS | | | | | | | | | | Combined ($n$=60) | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Code | Doc | Debug | DevOps | Pipe. | SecIR | MLOps | Front. | DB | API-I | PQS ↑ | Prec ↑ | SND ↑ | NL ↓ |
| **Cloud LLM Planners** | | | | | | | | | | | | | | | |
| Kimi K2.5 | Raw Context | 0.77 | 0.77 | 0.67 | 0.81 | 0.85 | 0.80 | 0.93 | 0.84 | 0.93 | 0.98 | 0.843 | 0.831 | 0.000 | 1.000 |
| Kimi K2.5 | PII Redaction | 0.85 | 0.80 | 0.75 | 0.88 | 0.93 | 0.93 | 0.89 | 0.88 | 0.88 | 1.00 | 0.887 | 0.886 | 0.835 | 0.165 |
| Kimi K2.5 | PLANTWIN | 0.79 | 0.80 | 0.68 | 0.72 | 1.00 | 0.74 | 0.79 | 0.67 | 0.90 | 0.91 | 0.808 | 0.776 | 1.000 | 0.617 |
| Gemini 3 Flash | Raw Context | 0.81 | 0.40 | 0.67 | 0.90 | 0.95 | 0.88 | 0.87 | 0.78 | 0.91 | 1.00 | 0.832 | 0.833 | 0.000 | 1.000 |
| Gemini 3 Flash | PII Redaction | 0.89 | 0.40 | 0.71 | 0.95 | 0.89 | 0.91 | 1.00 | 0.90 | 0.83 | 0.95 | 0.861 | 0.848 | 0.835 | 0.165 |
| Gemini 3 Flash | PLANTWIN | 0.71 | 0.68 | 0.57 | 0.72 | 0.93 | 0.48 | 0.68 | 0.70 | 0.66 | 0.77 | 0.689 | 0.604 | 1.000 | 0.582 |
| MiniMax M2.5 | Raw Context | 0.79 | 0.65 | 0.67 | 0.76 | 0.83 | 0.77 | 0.87 | 0.78 | 0.95 | 0.89 | 0.802 | 0.761 | 0.000 | 1.000 |
| MiniMax M2.5 | PII Redaction | 0.65 | 0.75 | 0.73 | 0.77 | 0.78 | 0.79 | 0.87 | 0.75 | 0.91 | 0.83 | 0.788 | 0.747 | 0.835 | 0.165 |
| MiniMax M2.5 | PLANTWIN | 0.75 | 0.69 | 0.67 | 0.83 | 1.00 | 0.70 | 0.86 | 0.73 | 0.72 | 0.81 | 0.792 | 0.799 | 1.000 | 0.590 |
| GLM 5 | Raw Context | 0.77 | 0.40 | 0.60 | 0.89 | 0.90 | 0.87 | 1.00 | 0.90 | 0.82 | 0.98 | 0.834 | 0.845 | 0.000 | 1.000 |
| GLM 5 | PII Redaction | 0.77 | 0.40 | 0.67 | 0.94 | 0.95 | 0.95 | 1.00 | 0.93 | 0.78 | 0.94 | 0.855 | 0.872 | 0.835 | 0.165 |
| GLM 5 | PLANTWIN | 0.81 | 0.75 | 0.64 | 0.72 | 0.98 | 0.80 | 0.87 | 0.67 | 0.93 | 0.83 | 0.810 | 0.765 | 1.000 | 0.615 |
| **Local Baselines (No Cloud)** | | | | | | | | | | | | | | | |
| No cloud | Local (SmolLM2-360M)‡ | 0.10 | 0.50 | 0.50 | 0.12 | 0.38 | 0.50 | 0.17 | 0.33 | 0.17 | 0.17 | 0.283 | 0.567 | 1.000 | 0.000 |
| No cloud | Local (Qwen 3.5-0.8B)‡ | 0.10 | 0.50 | 0.50 | 0.12 | 0.38 | 0.50 | 0.17 | 0.33 | 0.17 | 0.17 | 0.283 | 0.567 | 1.000 | 0.000 |
| No cloud | Local (Gemma-3-1B) | 0.55 | 0.86 | 0.50 | 0.54 | 0.60 | 0.57 | 0.55 | 0.55 | 0.55 | 0.44 | 0.567 | 0.559 | 1.000 | 0.000 |
| No cloud | Local (Qwen 3.5-2B) | 0.10 | 0.50 | 0.50 | 0.12 | 0.38 | 0.50 | 0.17 | 0.33 | 0.17 | 0.17 | 0.283 | 0.567 | 1.000 | 0.000 |
| No cloud | Local (Qwen 3.5-4B) | 0.27 | 0.40 | 0.50 | 0.46 | 0.58 | 0.50 | 0.52 | 0.70 | 0.54 | 0.30 | 0.484 | 0.656 | 1.000 | 0.000 |
| No cloud | Local (Qwen 3.5-9B) | 0.10 | 0.50 | 0.86 | 0.12 | 0.64 | 0.93 | 0.73 | 0.50 | 0.45 | 0.60 | 0.531 | 0.642 | 1.000 | 0.000 |
| No cloud | Heuristic (Oracle) | 0.53 | 0.50 | 0.50 | 0.78 | 0.90 | 0.70 | 0.59 | 0.54 | 0.43 | 0.84 | 0.653 | 0.782 | 1.000 | 0.558 |

- PQS = Plan Quality Score (↑, $F_1$ of capability precision and recall), Prec = Capability Precision (↑).
- SND = Sensitive Non-Disclosure (↑), NL = Normalized Leakage (↓).
- ‡ Models $\leq$ 0.8B fail to generate valid JSON plans; all PQS values reflect the single-capability fallback.

Local-only models achieve SND= 1.0 but limited plan quality. Models at or below 2B produce PQS= 0.283 (identical fallback plans). Scaling improves PQS: Qwen3.5-4B reaches 0.484 and 9B reaches 0.531[2], though both exhibit high domain variance. Gemma-3-1B-it achieves the best local PQS (0.578) with only 1B parameters, suggesting instruction-tuning quality outweighs parameter count. The best local-only model still falls ∼28% below cloud-assisted PLANTWIN. The Heuristic (Oracle) baseline reaches PQS= 0.653 but varies widely by domain (0.43–0.90); PLANTWIN outperforms it on keyword-mismatched domains. A Friedman test confirms globally significant PQS differences ($\chi^2 = 237.55$, $p < 0.001$); PLANTWIN (Kimi) is significantly better than all local-only models and the heuristic ($p_{\text{Bonf}} < 0.001$).

Fig. 3 visualizes the trade-off: PLANTWIN configurations cluster in the upper-right quadrant (high SND, high PQS), an operating point no other method reaches.

**Privacy-utility trade-off.** PLANTWIN achieves SND= 1.0 with PQS statistically indistinguishable from full disclosure on 3 of 4 planners. Local-only models scale to PQS= 0.578 but remain ∼28% below cloud-assisted planning. PII redaction yields higher PQS but leaks 16.5% of sensitive items.

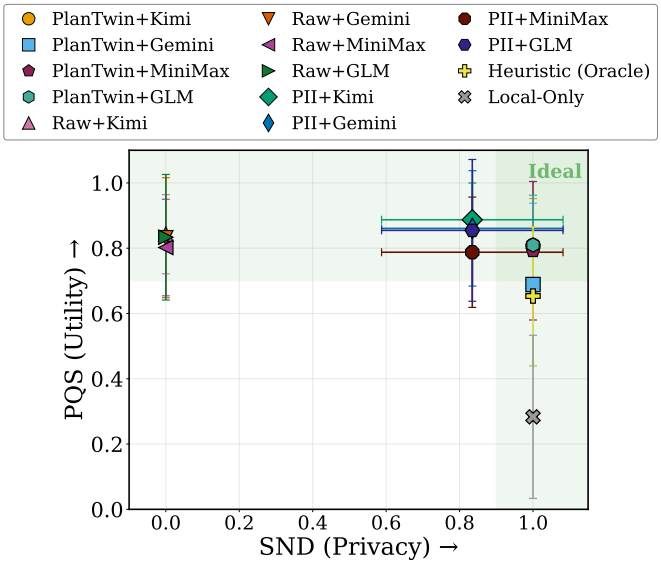


Fig. 3: Privacy–utility trade-off for nineteen method configurations over 60 tasks in 10 domains. Only PLANTWIN reaches the ideal high-privacy/utility region, achieving SND = 1.0 with PQS > 0.7. Error bars show per-task standard deviation.

### C. Budget, Overhead, and Per-Domain Analysis

**Budget utilization.** All four PLANTWIN configurations consume approximately 60% of the disclosure budget (NL $\in [0.582, 0.617]$), confirming that budget consumption is governed by the local gatekeeper rather than the cloud planner. No object exceeds its threshold $\tau(o)$.

[2]We re-verified all local-only baselines on the canonical 60-task benchmark during revision; the measured values reproduce the reported numbers within 0.001 (4B, 9B) and 0.012 (Gemma-3-1B-it: 0.578 measured vs. 0.567 reported), confirming the trend.

**Gatekeeper escalation rate.** In the canonical 60-task benchmark, the gatekeeper escalates 0/60 tasks to a human approver. The 0/60 number is a structural consequence of the evaluated capability catalogue rather than an empirical low-burden observation: the eight evaluated capabilities are all read-side analytical operations. Escalation is reserved for write-side capabilities (e.g., `execute_code`, `modify_file`, `delete_resource`, `send_external_request`) that are not part of the evaluation catalog. The escalation plumbing is in place (§IV-C) and is exercised in our internal integration

tests, but its empirical human-in-the-loop burden depends on the deployment-specific capability vocabulary, which we do not characterize here. Deployments that include write-side capabilities should expect escalation to materially affect operator load and should size their approval pipeline accordingly.

**System overhead.** Table V reports per-task latency. For PLANTWIN, SLM-backed Stage 1 extraction adds 13–34 s; deterministic Stages 2–4 add <1 ms. Cloud API latency varies by planner: Gemini is fastest (median 45.2 s total), followed by MiniMax (56.1 s) and GLM (63.3 s), while Kimi exhibits the highest variance (median 119.6 s). With heuristic-only extraction, total latency reduces to approximately the cloud API time. Local-only latency scales with model size, from median 2.9 s (Qwen3.5-0.8B) to 159.8 s (Qwen3.5-9B).

**Per-domain generalization.** With Kimi, PLANTWIN achieves PQS ranging from 0.67 (frontend) to 1.00 (data pipeline). Structured ETL and database workflows align naturally with the typed graph and achieve high PQS across all planners. The largest inter-planner variation occurs on security IR (Kimi 0.74 vs. Gemini 0.48), suggesting that multi-step reasoning benefits from stronger planners. PII redaction SND varies from 0.12 (coding) to 1.0 (documents), whereas PLANTWIN maintains SND= 1.0 across all domains.

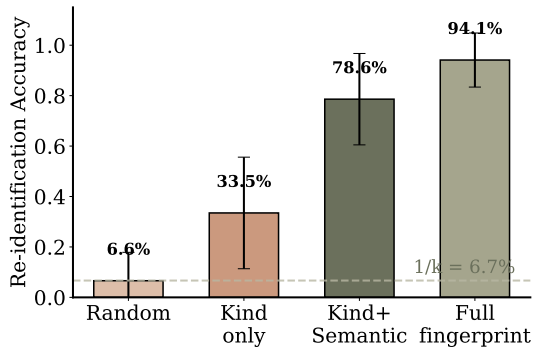


Fig. 4: Adversarial re-identification accuracy by strategy with $k = 15$ candidate files. Error bars show standard deviation, and the dashed line marks the random baseline $1/k$. Full fingerprint matching reaches 94.1% accuracy.

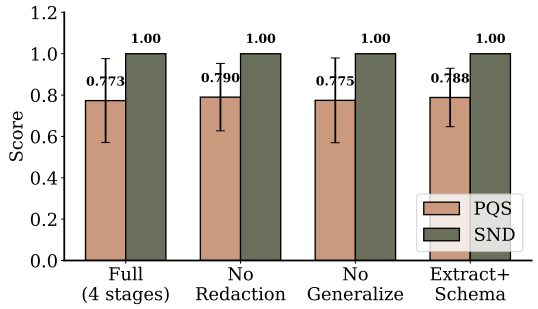


Fig. 5: Pipeline stage ablation across four variants. All maintain SND = 1.0, while PQS stays within 0.773–0.790, indicating minimal utility loss from privacy hardening. Error bars show per-task standard deviation.

### D. Adversarial Re-identification

We evaluate re-identification risk through a game-based experiment (Goal 1, §III-B): an adversary observes a target twin and matches it against $k$ candidates using four strategies with increasing auxiliary knowledge (random, kind-only, kind+semantic, full fingerprint). Over 500 trials ($k = 15$), random guessing achieves 6.6%, kind-only 33.5%, kind+semantic 78.6%, and full fingerprint 94.1% (Fig. 4). Scaling to $k = 30$ (Fig. 6), the full fingerprint adversary remains above 88%, confirming that twin fields are inherently identifying without budget control and empirically motivating the per-object disclosure budget.

**Re-identification risk.** Without budget control, full fingerprint matching achieves 94.1% accuracy ($k = 15$) and >88% at $k = 30$. Individual fields are not identifying, but their combination creates a unique fingerprint, validating per-object disclosure budgets.

**Deployment implication.** These results imply that the typed-graph abstraction is *not* intrinsically anonymous. Under a full-fingerprint adversary, top-1 re-identification accuracy remains 98.9% at candidate-pool size $k = 5$ and 88.3% at $k = 30$; the abstraction architecturally removes raw content but preserves a coarse structural signature that auxiliary public knowledge can still exploit. We therefore frame deployment safety as a layered property rather than an architectural guarantee. Practical safety is improved by, in order of empirical impact: (i) *larger candidate populations* within the operating context (deploying PLANTWIN in a setting with many indistinguishable peers reduces but does not eliminate the adversary's advantage); (ii) *broader common semantic classes* that reduce per-object distinguishing signal (avoiding rare `semantic_class` values that act as near-unique signatures); and (iii) *restricted contains-tag vocabularies* that limit auxiliary-knowledge linkage (specific `contains:*` combinations such as `[crypto_key, regulatory_id, dev_env_marker]` create a fingerprint even when no single tag is itself rare). None of these are properties PLANTWIN enforces by construction; they are deployment choices a system integrator makes. Conversely, the abstraction is least safe when the adversary controls a small candidate set, can attribute a rare semantic class to the target, or has external knowledge of distinctive contains-tag combinations.

### E. Stage 1 Extraction: SLM-Backed vs Heuristic

The Stage 1 typed-extraction layer can be implemented either with the deterministic heuristic (default in our main results) or with a local Qwen3.5-2B-Instruct SLM (§IV-A). To characterise when the SLM-backed variant changes conclusions, we re-ran the canonical 60-task benchmark with `use_slm=True` across all four cloud planners. The SLM runs on the same GPU as the local-only baselines; the cloud planner is reached through OpenRouter as in the main configuration.

Table VI reports the comparison. Pre-registered "stable" decision rule: paired mean PQS gap < 1pp *and* no domain

TABLE V: Per-task overhead breakdown (seconds)

| Planner | Method | Phase (mean) | | Per-Domain Total Latency | | | | | | | | | | Med. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Proj. | API | Code | Doc | Debug | DevOps | Pipe. | SecIR | MLOps | Front. | DB | API-I | |
| **Cloud LLM Planners** | | | | | | | | | | | | | | |
| Kimi K2.5 | Raw Context | — | 53.0 s | 67.9 s | 101.5 s | 52.7 s | 88.0 s | 105.4 s | 111.2 s | 95.4 s | 48.4 s | 248.2 s | 137.9 s | 66.8 s |
| Kimi K2.5 | PII Redaction | <0.1 s | 79.1 s | 92.3 s | 60.1 s | 34.9 s | 73.7 s | 92.4 s | 102.4 s | 42.6 s | 306.7 s | 109.6 s | 332.5 s | 75.2 s |
| Kimi K2.5 | **PLANTWIN** | 13.3 s | 81.2 s | 204.2 s | 135.2 s | 264.8 s | 65.5 s | 66.7 s | 112.4 s | 303.8 s | 377.0 s | 115.9 s | 103.2 s | 119.6 s |
| Gemini 3 Flash | Raw Context | — | 6.5 s | 17.8 s | 11.5 s | 12.9 s | 13.6 s | 12.2 s | 19.9 s | 19.3 s | 15.5 s | 17.1 s | 13.4 s | 14.4 s |
| Gemini 3 Flash | PII Redaction | <0.1 s | 8.5 s | 23.0 s | 12.8 s | 19.1 s | 14.9 s | 18.8 s | 28.8 s | 16.3 s | 17.6 s | 23.7 s | 27.4 s | 19.3 s |
| Gemini 3 Flash | **PLANTWIN** | 25.0 s | 8.9 s | 43.0 s | 34.2 s | 41.4 s | 54.6 s | 42.1 s | 65.4 s | 52.3 s | 31.7 s | 52.5 s | 46.4 s | 45.2 s |
| MiniMax M2.5 | Raw Context | — | 28.6 s | 14.1 s | 19.9 s | 30.5 s | 16.5 s | 95.9 s | 15.8 s | 17.0 s | 9.7 s | 20.0 s | 24.7 s | 16.6 s |
| MiniMax M2.5 | PII Redaction | <0.1 s | 44.8 s | 29.4 s | 135.0 s | 22.6 s | 28.8 s | 30.4 s | 16.0 s | 16.7 s | 115.4 s | 13.9 s | 55.3 s | 15.3 s |
| MiniMax M2.5 | **PLANTWIN** | 34.3 s | 19.3 s | 40.1 s | 36.6 s | 37.0 s | 61.6 s | 56.2 s | 69.0 s | 64.3 s | 42.6 s | 64.3 s | 50.1 s | 56.1 s |
| GLM 5 | Raw Context | — | 44.9 s | 44.9 s | 17.3 s | 34.0 s | 23.1 s | 28.2 s | 32.9 s | 43.3 s | 41.1 s | 34.4 s | 154.2 s | 28.4 s |
| GLM 5 | PII Redaction | <0.1 s | 66.6 s | 44.0 s | 36.7 s | 44.5 s | 48.6 s | 53.4 s | 61.3 s | 43.3 s | 72.2 s | 206.5 s | 49.9 s | 48.0 s |
| GLM 5 | **PLANTWIN** | 33.4 s | 32.1 s | 46.0 s | 34.6 s | 38.0 s | 70.2 s | 60.8 s | 102.3 s | 67.6 s | 49.5 s | 91.6 s | 77.4 s | 63.3 s |
| **Local Baselines (No Cloud)** | | | | | | | | | | | | | | |
| No cloud | Local (SmolLM2-360M)‡ | — | 12.3 s† | 9.7 s | 0.7 s | 3.1 s | 28.9 s | 1.9 s | 2.8 s | 5.7 s | 32.5 s | 33.4 s | 4.6 s | 5.2 s |
| No cloud | Local (Qwen 3.5-0.8B)‡ | — | 3.6 s† | 6.1 s | 2.6 s | 2.8 s | 4.4 s | 2.8 s | 2.8 s | 5.6 s | 2.9 s | 2.9 s | 2.9 s | 2.9 s |
| No cloud | Local (Gemma-3-1B) | — | 15.1 s† | 18.9 s | 11.8 s | 17.2 s | 17.5 s | 13.4 s | 14.0 s | 15.8 s | 14.3 s | 14.2 s | 13.7 s | 14.3 s |
| No cloud | Local (Qwen 3.5-2B) | — | 15.6 s† | 8.8 s | 1.9 s | 2.1 s | 49.1 s | 4.9 s | 12.0 s | 8.2 s | 61.8 s | 5.1 s | 2.2 s | 6.7 s |
| No cloud | Local (Qwen 3.5-4B) | — | 142.3 s† | 179.4 s | 22.8 s | 234.9 s | 171.2 s | 93.3 s | 234.3 s | 117.7 s | 126.7 s | 125.4 s | 117.6 s | 126.1 s |
| No cloud | Local (Qwen 3.5-9B) | — | 137.6 s† | 42.7 s | 34.9 s | 226.8 s | 80.7 s | 177.6 s | 228.1 s | 192.8 s | 73.0 s | 151.1 s | 168.5 s | 159.8 s |
| No cloud | Heuristic (Oracle) | <0.1 s | — | <0.1 s | <0.1 s | <0.1 s | <0.1 s | <0.1 s | <0.1 s | <0.1 s | <0.1 s | <0.1 s | <0.1 s | <0.1 s |

- Proj. = mean local projection time (twin construction + optional SLM extraction), API = mean cloud planner API round-trip.
- Per-domain columns report mean total latency. All four cloud planners use thinking mode. Kimi K2.5 exhibits high variance due to API load.
- †Local SLM inference (no cloud API); model size indicated in parentheses. "—" = not applicable.
- ‡Models $\leq$0.8B fail to generate valid JSON plans (fallback only).

shows $>$ 3pp regression *and* completion-rate parity within two tasks *and* latency 95th-pct overhead $< 30\%$.

TABLE VI: SLM-backed vs. heuristic Stage 1 on the canonical 60-task benchmark, four cloud planners. Negative $\Delta$PQS values indicate SLM *worsens* planning quality.

| Planner | Stage 1 | PQS ↑ | SND ↑ | Compl. | Latency (s) |
|---|---|---|---|---|---|
| Kimi K2.5 | Heuristic | 0.808 | 1.000 | 60/60 | 164.8 |
| | SLM-2B | 0.703 | 1.000 | 60/60 | 195.3 |
| MiniMax M2.5 | Heuristic | 0.792 | 1.000 | 58/60 | 53.6 |
| | SLM-2B | 0.639 | 1.000 | 60/60 | 143.4 |
| GLM 5 | Heuristic | 0.810 | 1.000 | 60/60 | 65.6 |
| | SLM-2B | 0.693 | 1.000 | 57/60 | 147.1 |
| Gemini 3 Flash | Heuristic | 0.689 | 1.000 | 55/60 | 47.7 |
| | SLM-2B | 0.703 | 1.000 | 59/60 | 123.7 |
| Mean $\Delta$PQS (SLM − Heuristic) | | $-0.090$ ($-$9pp) | | | |

**Findings.** The pre-registered stability rule *fails*: SLM Stage 1 reduces PQS by 10.6–15.4pp on Kimi, MiniMax, and GLM, and only matches heuristic on Gemini ($+1.4$pp). SND remains 1.0 across all configurations because the schema constraints downstream of Stage 1 are unchanged. Latency increases on every planner because SLM extraction adds 13–34 s per task and the resulting typed graph appears to require additional clarification turns from the cloud planner.

**Why SLM does not improve planning utility.** Two effects compound. First, Qwen3.5-2B occasionally produces near-identical typed labels for distinct artifacts (e.g., classifying both a `config` and a `log_stream` as `structured_record` when the file extension is ambiguous), which collapses domain-specific structural cues that the heuristic preserves through deterministic file-extension and regex matching. Second, the cloud planner appears to compensate by issuing more capability requests on the noisier graph; this consumes the disclosure budget faster, leading to more gatekeeper rejections without producing additional approved capabilities.

**Recommendation.** The heuristic Stage 1 is the recommended default for deployments where artifact types can be identified by deterministic patterns, which covers the synthetic benchmark and most enterprise workflows we have inspected. The SLM variant remains useful in two scenarios outside this paper's scope: (i) deployments where artifact types are linguistically ambiguous or domain-specific (e.g., specialised industrial log formats) and where heuristic patterns are absent, and (ii) research that requires reporting extraction confidence alongside the type label. Both are configuration choices and do not change PLANTWIN's privacy guarantees.

**Caveat on observed effect direction.** We pre-registered the stability rule before running the experiment but did not pre-register the direction. The observed PQS regression is consistent with downstream noise amplification, but ruling out alternative explanations (e.g., suboptimal Qwen3.5-2B prompt template, model-version-specific failure modes) would require ablations on the SLM prompt and on alternative SLMs (Qwen2.5-1.5B-Instruct, Phi-3-mini) that are out of scope here.

### F. Field-Cost Sensitivity

The weighted field-cost model (§IV-D, Table III) is a calibrated heuristic, not a formal information-theoretic bound. To characterise sensitivity to the cost shape, we perturb the field weights along three *non-collinear* regimes that change the relative ordering of disclosure costs while holding the budget magnitude regime constant. For each regime, the per-sensitivity-level thresholds are re-fitted so that the average admission rate on a 12-task calibration slice matches the

published baseline; this isolates cost-shape effects from total-budget effects. Pre-registered hypothesis: PQS variation $< 5$pp across regimes; SND remains 1.0.

Table VII reports the four regimes on Kimi K2.5 over the canonical 60-task benchmark.

TABLE VII: Field-cost sensitivity on Kimi K2.5, canonical 60-task benchmark. R0 is the published configuration; R1–R3 are non-collinear perturbations with re-fitted thresholds.

| Regime | Cost shape (relative) | PQS ↑ | SND ↑ | NL ↓ | ΔPQS |
|---|---|---|---|---|---|
| R0 baseline | published weights | 0.808 | 1.000 | 0.617 | — |
| R1 flat | all weights equal | 0.656 | 1.000 | 0.557 | −15.3pp |
| R2 identifier-heavy | id-class ×2, structural ×0.5 | 0.673 | 1.000 | 0.463 | −13.6pp |
| R3 structure-heavy | structural ×2, id-class ×0.5 | 0.726 | 1.000 | 0.410 | − 8.3pp |

**Findings.** The pre-registered $< 5$pp stability hypothesis is rejected on all three perturbations: every regime drops PQS by 8–15pp relative to the published baseline. The published identifier-heavy weighting is therefore non-trivially load-bearing for plan quality, and operators cannot expect that arbitrary alternative weightings will yield comparable utility. NL also drops in every perturbation regime, indicating the planner consumes the disclosure budget more conservatively when the cost ordering is altered.

**What the experiment does and does not show. (i) Privacy is preserved across all regimes.** SND stays at 1.0 because the architectural guarantee comes from the typed-graph schema and gatekeeper, not from the weights. A redeployment with different weights does not weaken the SND claim. **(ii) Plan quality is sensitive to cost shape.** The published weights reflect a calibrated semantic ordering (identifiers $>$ semantic classes $>$ structural fields); flattening or reversing this ordering measurably hurts utility, plausibly because the budget is consumed on less informative disclosures and the cloud planner has fewer high-signal observations. **(iii) The cost model is a heuristic, not a robust specification.** We do not claim that any particular weight assignment is optimal, only that the published configuration is locally calibrated and that operators reusing PLANTWIN should retain the same relative ordering or recalibrate against their own task distribution.

**Recommendation.** Use the published weights as-is. Alternative weights with the same admission-rate calibration are acceptable but should be expected to reduce PQS by $\sim$ 8–15pp without changing privacy properties. Lifting this heuristic to a formal mutual-information bound is an open direction (see Appendix C).

### G. Ablation Studies

**Pipeline stage ablation.** We evaluate four pipeline variants (Full, No Redaction, No Generalize, Extract+Schema) on 58 tasks with Kimi K2.5. All maintain SND$= 1.0$; PQS ranges from 0.773 (Full) to 0.790 (No Redaction), a spread of only 0.017 (Fig. 5). The privacy-hardening stages impose $<$2.2% PQS cost, while the typed abstract graph alone guarantees data isolation.

**Budget threshold sensitivity.** Varying the budget scale $\alpha \in \{0.25, 0.5, 0.75, 1.0, 1.5\}$ (Fig. 7), PQS saturates at $\alpha \geq 0.75$ (PQS$= 0.801$) and does not improve at $\alpha = 1.5$. At $\alpha = 0.25$, PQS halves to 0.513. The sweet spot $\alpha \in [0.75, 1.0]$ achieves PQS$> 0.80$ with NL$< 0.78$.

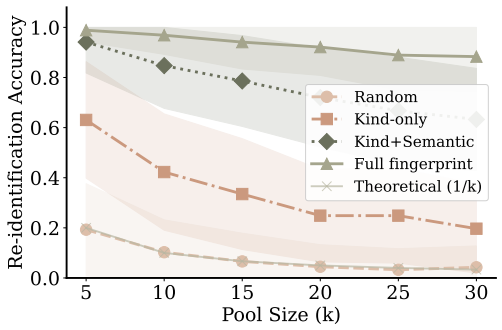


Fig. 6: Re-identification accuracy vs. anonymity set size $k$. Full fingerprint matching exceeds 88% even at $k = 30$. Shaded regions: $\pm 1$ std. dev.

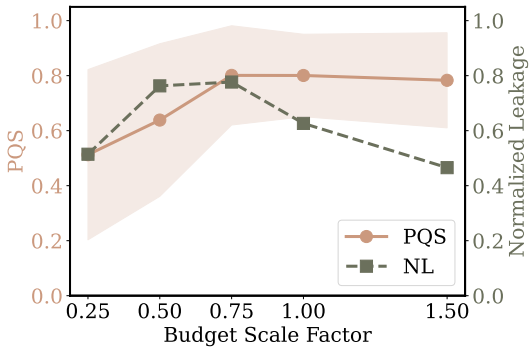


Fig. 7: Budget threshold sensitivity vs. scale factor $\alpha$. PQS saturates at $\alpha \geq 0.75$; NL peaks then declines as larger budgets leave headroom. Shading: $\pm 1$ std. dev.

**Multi-turn budget depletion.** Under persistent multi-turn usage across 58 tasks (Fig. 8, Fig. 9), overall PQS decreases by only 0.8% (from 0.773 to 0.767), while NL reaches 1.0 as per-object budgets are fully exhausted. SND remains 1.0 throughout: over-budget objects are excluded rather than leaking additional information.

**Ablation summary.** Privacy-hardening stages cost $<$2.2% PQS; budget utility saturates at $\alpha \geq 0.75$; multi-turn accumulation depletes budgets but degrades PQS by only 0.8% while maintaining SND$= 1.0$.

## VI. DISCUSSION

### A. Privacy and Security Discussion

PLANTWIN reduces exposure architecturally rather than cryptographically. The cloud never receives raw files, paths, credentials, or unsanitized outputs; all planner-visible state passes through the projection pipeline and is expressed as bounded schema fields. Content reconstruction is possible only if projection emits overly specific attributes.

For re-identification (Goal 1), the twin is not intrinsically anonymous: disclosing all fields enables 94.1% re-identification (§V-D). Privacy comes from partial observabil-

ity enforced by the per-object disclosure budget. For cross-session linkage (Goal 2), session-local random IDs, bounded vocabularies, and cumulative limits collectively prevent the planner from accumulating enough stable attributes for reliable linkage. For structural inference (Goal 3), graph topology remains informative, and bounded edge vocabularies reduce but do not eliminate this signal.

The architecture also narrows prompt-injection and plan-level attack surfaces: the planner sees only schema abstractions, and all plan execution is mediated locally by capability validation, policy checks, and disclosure accounting.

### B. Compliance-Oriented Deployment

PLANTWIN's architectural properties map to several recognized data-protection principles, supporting compliance-oriented deployment patterns under appropriate provider and region choices. We highlight three.

**Data minimization** (GDPR Art. 5(1)(c); CCPA §1798.100(c)). The typed-graph projection ensures that only typed structural summaries, not raw artifact content, leave the local boundary. Every cloud-bound payload is a strict subset of what the agent could in principle send. This realizes the data-minimization principle architecturally rather than by policy.

**Purpose limitation.** The capability vocabulary explicitly enumerates the planner's permissible operations on each typed object class. Capabilities are bound to specific planning purposes and cannot be re-purposed at runtime: a capability that was approved to summarize logs cannot be silently coerced into reading database rows. This makes the planner-side intent machine-checkable rather than text-policy.

**Data residency and sovereignty.** PLANTWIN is residency-compatible but not residency-enforcing. Because the projection layer is the only egress point, deployments can pair PLANTWIN with a regional or on-premises cloud planner to keep all processing within a designated jurisdiction. Cross-border processing, when the cloud planner is in a different region, is a deployment choice that PLANTWIN does not architecturally prevent. Residency is therefore a property of the PLANTWIN-plus-planner pair, not of PLANTWIN alone.

We do not claim that deploying PLANTWIN produces compliance with any specific regulation; that determination depends on the full deployment context (controller/processor relationships, data subject category, lawful basis). PLANTWIN lowers the engineering effort required to construct compliant deployments by providing a clean separation between local sensitive content and cloud-bound abstract input, but compliance remains a system-level property determined by the deployer.

### C. Complexity and Scalability

Let $n$ be the number of tracked objects, $m$ the number of typed edges, $k$ the number of plan steps, $|F_i|$ the token volume of object $i$, and $d$ the average retained schema fields per object. Stage 1 (SLM extraction) costs $O(\sum_i |F_i| \cdot c_{\text{SLM}})$; Stages 2–4 are $O(nd + m)$. Disclosure accounting is $O(k)$ per cycle.

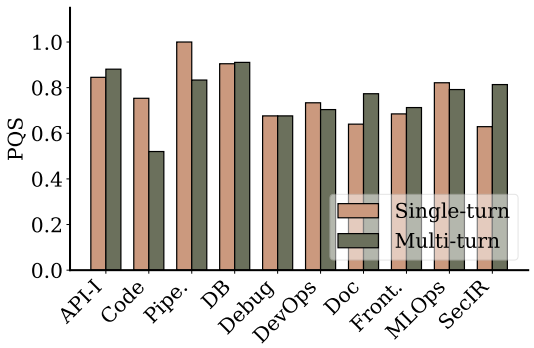


Fig. 8: Per-domain PQS under single- and multi-turn budgeting. Multi-turn most degrades data-pipeline and coding tasks. Repeated access to the same objects exhausts budgets.

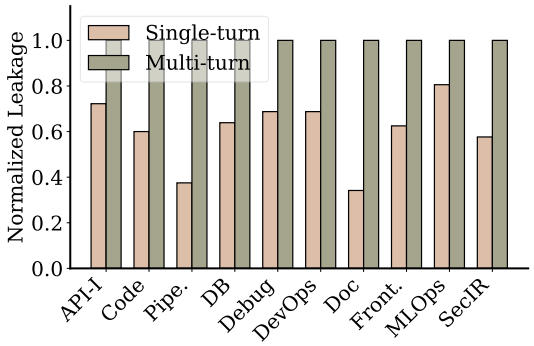


Fig. 9: Per-domain normalized leakage in single-/multi-turn modes. Multi-turn drives NL to 1.0 across domains.

Cloud planning costs $O(L)$, where $L$ is planner inference cost over the compact twin. The combined per-cycle cost is

$$O\left(\sum_i |F_i| \cdot c_{\text{SLM}} + nd + m + L + k\alpha\right),$$

where $\alpha$ is per-step execution cost. In practice, Stage 1 dominates for large environments and cloud inference dominates for reasoning-heavy workflows, consistent with latency measurements in §V-C.

**Toward Production Deployment.** A natural extension is *cloud-side sandbox construction*: upon receiving $Z_t$, the cloud instantiates an ephemeral container mirroring the schema graph with synthetic file stubs. The planner iterates within this sandbox (running linters, analyzers) before producing a verified plan while preserving the same SND and capability bounds enforced by PLANTWIN's local layer.

**Proposition 1** (Synthetic-population privacy bound)**.** *Let* $\text{Synth} : \mathcal{Z} \to \mathcal{S}$ *be a sandbox generator taking only* $Z_t = \Pi(X_t)$ *as input, and let* $R_{1:T}$ *denote tool outputs from planner interaction with* $S = \text{Synth}(Z_t)$*. Then* $X_t \to Z_t \to S \to R_{1:T}$ *forms a Markov chain, so* $I(X_t; S, R_{1:T} \mid Z_t) = 0$*. The* $(k, \delta)$*-anonymity (Theorem 1),* $\epsilon$*-unlinkability (Theorem 2), and composition (Theorem 3) bounds hold with identical parameters by the post-processing property.*

This requires that $\mathrm{Synth}$ uses only $Z_t$ (no external corpora) and the planner carries no side information correlated with $X_t$ beyond what $Z_t$ discloses. Lightweight sandbox runtimes such as E2B [21] or gVisor [22] provide the necessary infrastructure; integration with MCP [9] follows naturally from the capability interface. We leave empirical evaluation of sandbox-assisted planning to future work.

## VII. Conclusion

We presented PlanTwin, a schema-constrained projection-based system for cloud-assisted LLM planning. Instead of exposing raw local context, it limits the remote planner to a bounded local abstraction through capability mediation and multi-turn disclosure budgets. PlanTwin achieves non-disclosure of raw sensitive content (SND $= 1.0$) under our threat model, while recognizing structural identifiability as a residual deployment concern, and maintains planning quality close to full-context cloud planning across 60 tasks. It outperforms simple redaction and local weak-model baselines on the privacy–utility trade-off. The design is planner-agnostic and robust under ablations and budget-constrained multi-turn interaction. Our results show that strong cloud planning does **NOT** require raw-context exposure. Planning-time observability should therefore be treated as a first-class systems interface in privacy-aware agents.

## References


[1] Anthropic, "Best practices for claude code," https://code.claude.com/docs/en/best-practices, 2025, accessed: 2026-03-01.

[2] OpenAI, "Introducing codex," https://openai.com/index/introducing-codex/, 2025, accessed: 2026-03-01.

[3] L. Lin, Y. Jin, Y. Zhou, W. Chen, and C. Qian, "Mao: A framework for process model generation with multi-agent orchestration," *IEEE Transactions on Services Computing*, vol. 18, no. 6, pp. 3987–4000, 2025.

[4] M. Liu, Z. Yin, C. Tian, S. Yu, T. Cai, Z. Xu, and Z. Wang, "Mars: A multi-agent collaborative reasoning framework for service recommendation," *IEEE Transactions on Services Computing*, pp. 1–17, 2026.

[5] X. Zhang, Q. Wang, M. Li, Y. Yuan, M. Xiao, F. Zhuang, and D. Yu, "Tamo:fine-grained root cause analysis via tool-assisted llm agent with multi-modality observation data in cloud-native systems," *IEEE Transactions on Services Computing*, vol. 18, no. 6, pp. 4221–4233, 2025.

[6] B. Zhang, D. Chu, S. Xiong, Q. Gong, and Z. Tu, "User-oriented fuzzy requirement clarification for agent services delivery based on llms," *IEEE Transactions on Services Computing*, pp. 1–14, 2025.

[7] Y. Liao, J. Bian, Y. Yun, S. Wang, Y. Zhang, J. Chu, T. Wang, Y. Li, X. Li, S. Ji, and H. Xiong, "Sagecopilot: An llm-empowered autonomous agent for data science as a service," *IEEE Transactions on Services Computing*, vol. 19, no. 1, pp. 642–656, 2026.

[8] Anthropic, "Introducing sandbox mode for claude code," https://code.claude.com/docs/en/sandboxing, 2025, accessed: 2026-03-01.

[9] Anthropic, "Introducing the model context protocol," *https://www.anthropic.com/news/model-context-protocol*, 2024, accessed: 2026-03-01.

[10] Microsoft, "Microsoft presidio: Data protection and de-identification sdk," https://github.com/microsoft/presidio, 2025.

[11] strands-agents, "Strands agents: Open-source ai agent framework," https://github.com/strands-agents/sdk-python, 2026, built-in PII redaction module.

[12] Z. by PrivateGPT, "Privategpt," https://github.com/zylon-ai/private-gpt, 2023.

[13] Ollama, "Ollama," https://ollama.ai, 2026.

[14] NVIDIA, "Confidential computing on nvidia h100 gpus for secure and trustworthy ai," https://developer.nvidia.com/blog/confidential-computing-on-h100-gpus-for-secure-and-trustworthy-ai/, 2023, accessed 2026-03-18.

[15] D. Ben, H. Feng, and Q. Wang, "Distilled large language model in confidential computing environment for system-on-chip design," 2025. [Online]. Available: https://arxiv.org/abs/2507.16226

[16] Intel, "Intel trust domain extensions (intel tdx) overview," https://www.intel.com/content/www/us/en/developer/tools/trust-domain-extensions/overview.html, 2026, accessed 2026-03-18.

[17] T. Shi, J. He, Z. Wang, H. Li, L. Wu, W. Guo, and D. Song, "Progent: Programmable privilege control for llm agents," *arXiv preprint arXiv:2504.11703*, 2025.

[18] J. Zhu, K. Tseng, G. Vernik, X. Huang, S. G. Patil, V. Fang, and R. A. Popa, "Miniscope: A least privilege framework for authorizing tool calling agents," 2025. [Online]. Available: https://arxiv.org/abs/2512.11147

[19] Z. Ji, D. Wu, W. Jiang, P. Ma, Z. Li, Y. Gao, S. Wang, and Y. Li, "Taming various privilege escalation in llm-based agent systems: A mandatory access control framework," 2026. [Online]. Available: https://arxiv.org/abs/2601.11893

[20] B. Ma, Y. Jiang, X. Wang, G. Yu, Q. Wang, C. Sun, C. Li, X. Qi, Y. He, W. Ni *et al.*, "Sok: Semantic privacy in large language models," *arXiv preprint arXiv:2506.23603*, 2025.

[21] E2B, "E2b: Open-source cloud runtime for ai agents," https://github.com/e2b-dev/e2b, 2025, apache 2.0 License.

[22] SkyPilot, "Agent sandbox for kubernetes," https://github.com/skypilot-org/skypilot, 2025, gVisor-based isolation.

[23] X. Li, D. Huang, J. Li, H. Cai, Z. Zhou, W. Dong, X. Wang, and Y. Liu, "A vision for access control in llm-based agent systems," 2025. [Online]. Available: https://arxiv.org/abs/2510.11108

[24] S. Zhao, Q. Hou, Z. Zhan, Y. Wang, Y. Xie, Y. Guo, L. Chen, S. Li, and Z. Xue, "Mind your server: A systematic study of parasitic toolchain attacks on the mcp ecosystem," 2025. [Online]. Available: https://arxiv.org/abs/2509.06572

[25] H. Zhang, J. Huang, K. Mei, Y. Yao, Z. Wang, C. Zhan, H. Wang, and Y. Zhang, "Agent security bench (asb): Formalizing and benchmarking attacks and defenses in llm-based agents," 2025. [Online]. Available: https://arxiv.org/abs/2410.02644

[26] Snyk, "Snyk agent scan," https://github.com/snyk/agent-scan, 2026, gitHub repository, accessed 2026-03-17.

[27] Y. Hu, C. Fan, S. Samyoun, and J. Du, "Log-to-leak: Prompt injection attacks on tool-using LLM agents via model context protocol," 2026. [Online]. Available: https://openreview.net/forum?id=UVgbFuXPaO

[28] X. Zong, Z. Shen, L. Wang, Y. Lan, and C. Yang, "MCP-safetybench: A benchmark for safety evaluation of large language models with real-world MCP servers," in *International Conference on Learning Representations (ICLR)*, 2026.

[29] Q. Wang, G. Yu, Y. Sai, H. D. Bandara, and S. Chen, "Is your AI truly yours? leveraging blockchain for copyrights, provenance, and lineage," *IEEE Transactions on Services Computing (TSC)*, 2025.

[30] Y. Jiang, G. Yu, Q. Yu, Y. Chen, and Q. Wang, "Why neural structural obfuscation can't kill white-box watermarks for good!" *arXiv preprint arXiv:2603.12679*, 2026.

[31] Y. Jiang, D. Li, H. Deng, B. Ma, X. Wang, Q. Wang, and G. Yu, "Sok: Agentic skills–beyond tool use in LLM agents," *arXiv preprint arXiv:2602.20867*, 2026.

[32] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of Cryptography Conference (TCC)*. Springer, 2006, pp. 265–284.

[33] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *ACM SIGSAC conference on computer and communications security (CCS)*, 2016, pp. 308–318.

[34] C. Song and A. Raghunathan, "Information leakage in embedding models," in *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2020, pp. 377–390.

[35] J. Morris, V. Kuleshov, V. Shmatikov, and A. M. Rush, "Text embeddings reveal (almost) as much as text," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (ACL)*, 2023, pp. 12 448–12 460.

[36] H. Li, M. Xu, and Y. Song, "Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence," in *Findings of the Association for Computational Linguistics: (ACL)*, 2023, pp. 14 022–14 040.

[37] T. Geng, Z. Xu, Y. Qu, and W. E. Wong, "Prompt injection attacks on large language models: A survey of attack methods, root causes, and defense strategies," *Computers, Materials, & Continua*, vol. 87, no. 1, 2026.

[38] N. Maloyan and D. Namiot, "Prompt injection attacks on agentic coding assistants: A systematic analysis of vulnerabilities in skills, tools, and protocol ecosystems," 2026. [Online]. Available: https://arxiv.org/abs/2601.17548
[39] National Vulnerability Database, "CVE-2025-53773 Detail," https://nvd.nist.gov/vuln/detail/CVE-2025-53773, 2025, national Institute of Standards and Technology (NIST), published 2025-08-12, last modified 2025-08-15, accessed 2026-03-18.
[40] Qwen Team, "Qwen3 technical report," *arXiv preprint arXiv:2505.09388*, 2025.
[41] Kimi Team, "Kimi K2: Open agentic intelligence," *arXiv preprint arXiv:2507.20534*, 2026, 1T MoE, 32B active parameters, 256K context, open-weight.
[42] MiniMax, "Minimax M2.5: Large-scale language model with extended reasoning," https://www.minimax.io/news/minimax-m25, 2026, accessed via OpenRouter API with thinking mode enabled.
[43] GLM-5-Team, "Glm-5: from vibe coding to agentic engineering," 2026, accessed via OpenRouter API with thinking mode enabled.

TABLE VIII: Notation.

| Symbol | Meaning |
|---|---|
| $X_t$ | Full local context at time $t$ |
| $Z_t$ | Sanitized digital twin (= $G_t$) |
| $\Pi(\cdot)$ | Privacy projection operator (4-stage pipeline) |
| $\Pi_{\text{out}}$ | Output sanitizer (Stages 2–4 on execution results) |
| $G_t = (\mathcal{V}_t, \mathcal{E}_t, \mathcal{A}_t)$ | Typed abstract graph (nodes, edges, attributes) |
| $\mathcal{C} = \{c_1, \ldots, c_n\}$ | Local capability catalog |
| $P_t$ | Cloud-generated plan at time $t$ |
| $\Phi(\cdot)$ | Cloud planner function |
| $Y_t$ | Sanitized execution result |
| $\Psi(\cdot)$ | Local gatekeeper + executor |
| $\Omega$ | Policy set |
| $B(o)$ | Cumulative disclosure budget for object $o$ |
| $\Delta_t(o)$ | New disclosure at turn $t$ for object $o$ |
| $\tau(o)$ | Disclosure threshold for object $o$ |
| $\mathcal{L}$ | System-wide leakage: $\max_o B(o)$ |
| $\epsilon$ | Privacy budget bound |
| $w_f$ | Base disclosure cost of schema field $f$ |
| grain($\mathcal{C}$) | Avg. weighted schema fields per capability call |

- *Upper part*: system components. *Lower part*: privacy analysis.

## APPENDIX A
## DISCLOSURE COST ACCOUNTING

The weighted field-cost model assigns each schema field a base cost $w_f$ reflecting its re-identification potential. For an object $o$ with attribute set $A(o)$, the initial disclosure cost is

$$\Delta_0(o) = \sum_{f \in A(o)} w_f. \tag{12}$$

For subsequent turns, only new reveals incur additional cost:

$$\Delta_t(o) = \sum_{f \in A_t(o) \setminus A_{1:t-1}(o)} w_f, \tag{13}$$

where $A_t(o)$ is the set of fields disclosed at turn $t$ and $A_{1:t-1}(o)$ is the set disclosed previously. Thus, retransmitting the same twin state without new fields incurs zero additional disclosure cost.

## APPENDIX B
## SECURITY PROOFS

This appendix formalizes the security properties stated as design goals in §III-B. We prove $(k, \delta)$-anonymity under schema projection (Theorem 1), $\epsilon$-unlinkability under fresh session IDs (Theorem 2), and a composition bound on cumulative disclosure across $T$ turns (Theorem 3).

### A. Notation and Setup

For the proofs below, we abstract away implementation details and work directly with the projected attribute space induced by the planner-visible abstraction.

Let $\mathcal{O} = \{o_1, \ldots, o_n\}$ be the set of real-world objects in the local environment. The schema projection $\Pi$ maps each object to a planner-visible representation $z = \Pi(o)$ drawn from a finite attribute space

$$\mathcal{Z} = \mathcal{K} \times \mathcal{S} \times 2^{\mathcal{T}} \times 2^{\mathcal{U}} \times \mathcal{F} \times \mathcal{R},$$

where $\mathcal{K}$ is the set of kind labels ($|\mathcal{K}| = 6$), $\mathcal{S}$ is the set of semantic-class labels, $\mathcal{T}$ is the set of content tags, $\mathcal{U}$ is the set of usability tags, $\mathcal{F}$ is the set of freshness buckets ($|\mathcal{F}| = 3$), and $\mathcal{R}$ is the set of sensitivity levels ($|\mathcal{R}| = 3$).

Let fields$(z)$ denote the set of fields present in $z$. For a representation $z$ with disclosed field subset $D \subseteq$ fields$(z)$, define the *equivalence class*

$$\mathrm{EC}(z, D) = \{o' \in \mathcal{O} \mid \Pi(o')|_D = z|_D\}, \tag{14}$$

that is, the set of objects whose projected representation agrees with $z$ on all disclosed fields.

The disclosure budget for object $o$ after $T$ turns is $B_T(o) = \sum_{t=1}^{T} \Delta_t(o)$, capped by threshold $\tau(o)$ (Eq. 11).

*a) Auxiliary-knowledge model.:* For Theorem 1, we model adversarial side information by a random variable $\mathcal{I}$. We assume a bounded-likelihood-ratio condition inside each equivalence class: there exists a bound $\eta \geq 1$ such that for every disclosed pattern $x$, every $o, o' \in \mathcal{O}_x := \mathrm{EC}(x, D)$, and every realization $i$ of $\mathcal{I}$,

$$\frac{\Pr[\mathcal{I} = i \mid O = o]}{\Pr[\mathcal{I} = i \mid O = o']} \leq \eta. \tag{15}$$

Thus, auxiliary knowledge may bias the posterior within an equivalence class, but cannot arbitrarily single out one member.

*b) Cost calibration model.:* For Theorems 2 and 3, let $F_t$ denote the set of newly disclosed fields about object $o$ at turn $t$, so that $\Delta_t(o) = \sum_{f \in F_t} w_f$. For each field $f$, define its *worst-case branching factor*

$$b_f := \max_{D,x} \big|\{v \in \mathcal{V}_f : \exists\, o \in \mathrm{EC}(x, D) \text{ with } \Pi_f(o) = v\}\big|, \tag{16}$$

where $\mathcal{V}_f$ is the value domain of field $f$, and the maximum ranges over all previously disclosed field sets $D$ and patterns $x$ on $D$. We assume field costs are calibrated as

$$w_f \geq \ln b_f. \tag{17}$$

Hence, revealing field $f$ can refine an existing equivalence class by at most a multiplicative factor $e^{w_f}$ in the worst case.

### B. Theorem 1: $(k, \delta)$-Anonymity under Schema Projection

**Theorem 1** (Re-identification Bound)**.** *Let the adversary observe $z^* = \Pi(o^*)$ with disclosed field set $D^*$ such that $\sum_{f \in D^*} w_f \leq \tau(o^*)$ (enforced by the gatekeeper's per-object budget). Assume:*

1) *the prior $\mu$ is uniform within each equivalence class, that is, conditional on $\Pi(O)|_{D^*} = x$, every object in $\mathcal{O}_x$ is equally likely a priori;*
2) *the adversary's side information $\mathcal{I}$ satisfies the bounded-likelihood-ratio condition (15).*

*If $|EC(z^*, D^*)| \geq k$, then for any adversary $\mathcal{A}$,*

$$\Pr[\mathcal{A}(z^*|_{D^*}, \mathcal{I}) = o^*] \leq \frac{\eta}{\eta + k - 1}. \tag{18}$$

*Equivalently,*

$$\Pr[\mathcal{A}(z^*|_{D^*}, \mathcal{I}) = o^*] \leq \frac{1}{k} + \delta, \tag{19}$$

*where*

$$\delta := \frac{\eta}{\eta + k - 1} - \frac{1}{k} = \frac{(\eta - 1)(k - 1)}{k(\eta + k - 1)}. \tag{20}$$

*Proof.* Fix the disclosed pattern $x := z^*|_{D^*}$ and write $E := \mathcal{O}_x = \mathrm{EC}(z^*, D^*)$, $m := |E|$. By assumption, $m \geq k$.

*Step 1 (Posterior inside the equivalence class).* Since all objects in $E$ induce the same disclosed pattern $x$, Bayes' rule gives, for any $o \in E$ and side-information realization $i$,

$$\Pr[O = o \mid \Pi(O)|_{D^*} = x, \mathcal{I} = i] = \frac{\Pr[\mathcal{I} = i \mid O = o]\ \Pr[O = o \mid \Pi(O)|_{D^*} = x]}{\sum_{o' \in E} \Pr[\mathcal{I} = i \mid O = o']\ \Pr[O = o' \mid \Pi(O)|_{D^*} = x]}. \tag{21}$$

By the uniform-within-class prior assumption,

$$\Pr[O = o \mid \Pi(O)|_{D^*} = x] = \frac{1}{m}$$

for all $o \in E$. The $1/m$ factors cancel, yielding

$$\Pr[O = o \mid \Pi(O)|_{D^*} = x, \mathcal{I} = i] = \frac{\Pr[\mathcal{I} = i \mid O = o]}{\sum_{o' \in E} \Pr[\mathcal{I} = i \mid O = o']}. \tag{22}$$

*Step 2 (Bounding the maximum posterior mass).* Let $\hat{o} \in E$ maximize the posterior in (22). By the likelihood-ratio bound (15), for every $o' \in E \setminus \{\hat{o}\}$,

$$\Pr[\mathcal{I} = i \mid O = o'] \geq \frac{1}{\eta} \Pr[\mathcal{I} = i \mid O = \hat{o}].$$

Therefore,

$$\sum_{o' \in E} \Pr[\mathcal{I} = i \mid O = o'] \geq \Pr[\mathcal{I} = i \mid O = \hat{o}] + (m-1) \cdot \frac{1}{\eta} \Pr[\mathcal{I} = i \mid O = \hat{o}]. \tag{23}$$

Substituting into (22),

$$\max_{o \in E} \Pr[O = o \mid x, i] \leq \frac{1}{1 + (m-1)/\eta} = \frac{\eta}{\eta + m - 1}. \tag{24}$$

Since $m \geq k$ and $u \mapsto \eta/(\eta + u - 1)$ is decreasing in $u$,

$$\max_{o \in E} \Pr[O = o \mid x, i] \leq \frac{\eta}{\eta + k - 1}. \tag{25}$$

*Step 3 (Optimal adversary success probability).* Given $(x, i)$, the optimal re-identification strategy is MAP decoding, whose success probability equals the maximum posterior mass. Hence the bound applies to every adversary $\mathcal{A}$:

$$\Pr[\mathcal{A}(z^*|_{D^*}, \mathcal{I}) = o^*] \leq \frac{\eta}{\eta + k - 1}.$$

The equivalent form with $\delta$ follows immediately. □

*Interpretation.* When $\eta = 1$, the bound reduces to the classical $1/k$ anonymity guarantee. Empirically, the re-identification experiment in §V-D corresponds to the unrestricted case in which all twin fields are disclosed; the budgeted setting limits the disclosed field set and enlarges the resulting equivalence class.

### C. Theorem 2: $\epsilon$-Unlinkability across Sessions

**Theorem 2** (Cross-Session Unlinkability)**.** *Let sessions $s_1, s_2$ assign fresh random object IDs independently. For a target object $o$, let $D^{(1)}$ and $D^{(2)}$ be the disclosed field sets in the two sessions, and let $U := D^{(1)} \cap D^{(2)}$ be the overlap of fields visible in both sessions. Denote the overlap-pattern random variable $X_U := \Pi(O)|_U$, where $O$ is drawn from prior $\mu$. Assume $X_U$ is uniform on its support $\mathcal{Z}_U \subseteq \prod_{f \in U} \mathcal{V}_f$, that is,*

$$\Pr[X_U = x] = \frac{1}{|\mathcal{Z}_U|}$$

*for all $x \in \mathcal{Z}_U$. Then for any adversary $\mathcal{A}$,*

$$\ln \frac{\Pr[\mathcal{A}(Z^{(1)}, Z^{(2)}) = 1 \mid \textit{same}]}{\Pr[\mathcal{A}(Z^{(1)}, Z^{(2)}) = 1 \mid \textit{diff}]} \leq \epsilon, \tag{26}$$

*where $\epsilon = \ln |\mathcal{Z}_U|$. Moreover, under the cost calibration assumption (17) and budget enforcement $\sum_{f \in D^{(i)}} w_f \leq \tau(o)$,*

$$\epsilon \leq \sum_{f \in U} \ln b_f \leq \sum_{f \in U} w_f \leq \tau(o). \tag{27}$$

*Proof.* Let $T_1, T_2$ denote the transcripts seen by the adversary in the two sessions, restricted to the overlap fields $U$. Fresh random IDs carry no cross-session information, so any linking rule depends only on the overlap attributes.

*Step 1 (Distribution under "same" and "diff").* The projection $\Pi$ is deterministic, so under the hypothesis same (both sessions involve the same object $O$), both transcripts produce the identical overlap pattern: $(T_1, T_2) = (X_U, X_U)$. Under the hypothesis diff (two independent objects $O_1, O_2 \sim \mu$), the overlap patterns are i.i.d.: $(T_1, T_2) = (X_U^{(1)}, X_U^{(2)})$ with $X_U^{(1)} \perp X_U^{(2)}$.

*Step 2 (Max-divergence computation).* For any adversary $\mathcal{A}$ with acceptance region $R \subseteq \mathcal{Z}_U \times \mathcal{Z}_U$,

$$\Pr[\mathcal{A} = 1 \mid \text{same}] = \sum_{x:\,(x,x) \in R} \Pr[X_U = x], \tag{28}$$

$$\Pr[\mathcal{A} = 1 \mid \text{diff}] = \sum_{(x,y) \in R} \Pr[X_U = x] \Pr[X_U = y] \geq \sum_{x:\,(x,x) \in R} \Pr[X_U = x]^2. \tag{29}$$

The inequality holds because the off-diagonal terms in $R$ only increase the denominator. Under the uniform-on-support assumption, $\Pr[X_U = x] = 1/N$ where $N = |\mathcal{Z}_U|$. Hence

$$\frac{\Pr[\mathcal{A} = 1 \mid \text{same}]}{\Pr[\mathcal{A} = 1 \mid \text{diff}]} \leq \frac{\sum_{x:\,(x,x) \in R} 1/N}{\sum_{x:\,(x,x) \in R} 1/N^2} = N = |\mathcal{Z}_U|. \tag{30}$$

This bound is tight: the adversary $\mathcal{A}^*(x, y) = \mathbf{1}[x = y = x_0]$ for any fixed $x_0 \in \mathcal{Z}_U$ achieves ratio exactly $N$.

*Step 3 (Budget connection).* The support size satisfies $|\mathcal{Z}_U| \leq \prod_{f \in U} |\mathcal{V}_f|$. Under the cost calibration (17), $|\mathcal{V}_f| \leq b_f \leq e^{w_f}$, so

$$\epsilon = \ln |\mathcal{Z}_U| \leq \sum_{f \in U} \ln |\mathcal{V}_f| \leq \sum_{f \in U} w_f. \tag{31}$$

Since $U \subseteq D^{(i)}$ for each session $i$, and the budget enforces $\sum_{f \in D^{(i)}} w_f \leq \tau(o)$, we obtain $\epsilon \leq \tau(o)$. □

*Interpretation.* The parameter $\epsilon$ is directly bounded by the overlap of disclosed fields across sessions, and therefore by the disclosure budget $\tau(o)$. Tighter budgets reduce the attribute overlap available for linkage. When no fields overlap across sessions, $\epsilon = 0$ and linkage through planner-visible attributes becomes impossible.

*Remark (non-uniform distributions).* For non-uniform $X_U$, the worst-case adversary achieves ratio $1/\min_{x \in \text{supp}} \Pr[X_U = x]$, giving $\epsilon = -\ln \min_x \Pr[X_U = x]$. This can exceed $\ln |\mathcal{Z}_U|$ when the distribution is skewed. The budget connection still bounds the number of distinguishable overlap patterns, even when their frequencies are not uniform.

### D. Theorem 3: Composition Bound for Multi-Turn Disclosure

**Theorem 3** (Sequential Composition)**.** *Consider an object $o$ observed across $T$ turns. Let $D_t$ be the cumulative disclosed field set after turn $t$, and let $EC_t := EC(\Pi(o), D_t)$ be the corresponding equivalence class. Assume the field costs satisfy the calibration condition (17). Then*

$$|EC_T| \geq |EC_0| \exp\Big(-\sum_{t=1}^{T} \Delta_t(o)\Big). \tag{32}$$

*In particular, if the gatekeeper enforces* $B_T(o) = \sum_{t=1}^{T} \Delta_t(o) \le \tau(o)$*, then*

$$|EC_T| \ge |EC_0|\, e^{-\tau(o)}. \tag{33}$$

*Proof.* For each turn $t$, let $F_t := D_t \setminus D_{t-1}$ be the set of newly disclosed fields, so that $\Delta_t(o) = \sum_{f \in F_t} w_f$.

*Step 1 (Monotone shrinkage).* Because $D_{t-1} \subseteq D_t$, every object consistent with the disclosures at turn $t$ is also consistent at turn $t-1$. Hence $\mathrm{EC}_t \subseteq \mathrm{EC}_{t-1}$ and $|\mathrm{EC}_t| \le |\mathrm{EC}_{t-1}|$.

*Step 2 (Per-field refinement bound).* Fix turn $t$ and suppose one additional field $f \in F_t$ is revealed. Within the pre-existing class $\mathrm{EC}_{t-1}$, disclosure of $f$ partitions that class into at most $b_f$ subclasses by definition of the branching factor (16). The subclass selected by the true value of field $f$ therefore has size at least $|\mathrm{EC}_{t-1}|/b_f$. Using the cost calibration (17), $b_f \le e^{w_f}$, so revealing field $f$ shrinks the class by at most a factor $e^{w_f}$:

$$|\mathrm{EC}^{(f)}| \ge |\mathrm{EC}_{t-1}|\, e^{-w_f}. \tag{34}$$

Applying this sequentially to all newly revealed fields in $F_t$ gives

$$|\mathrm{EC}_t| \ge |\mathrm{EC}_{t-1}| \exp\Big(-\sum_{f \in F_t} w_f\Big) = |\mathrm{EC}_{t-1}|\, e^{-\Delta_t(o)}. \tag{35}$$

*Step 3 (Sequential composition over $T$ turns).* Iterating (35) over $t = 1, \dots, T$ yields

$$|\mathrm{EC}_T| \ge |\mathrm{EC}_0| \prod_{t=1}^{T} e^{-\Delta_t(o)} = |\mathrm{EC}_0| \exp\Big(-\sum_{t=1}^{T} \Delta_t(o)\Big), \tag{36}$$

which proves (32). Budget enforcement $B_T(o) \le \tau(o)$ immediately gives (33). □

*Interpretation.* Once the budget is exhausted, the gatekeeper either withholds the object from the planner-visible abstraction or refuses further field disclosures. This yields graceful degradation: the planner may lose utility, but the equivalence class is prevented from shrinking beyond the budget-implied bound.

*Empirical alignment.* The multi-turn experiment (§V-G) is consistent with this bound: after 58 tasks, NL reaches 1.0 while several objects are excluded by the gatekeeper, demonstrating hard enforcement of the budget cap. PQS degrades only modestly, indicating graceful degradation rather than catastrophic failure.

*Corollary (product-form bound).* Since $e^{-x} \ge 1 - x$ for $x \ge 0$, we also obtain

$$|\mathrm{EC}_T| \ge |\mathrm{EC}_0| \prod_{t=1}^{T} \big(1 - \Delta_t(o)\big), \tag{37}$$

whenever each $\Delta_t(o)$ is normalized to lie in $[0, 1]$. This product form is weaker but sometimes more intuitive.

### E. Assumptions and Empirical Alignment

Table IX summarizes how each formal result connects to the empirical evaluation and highlights the main modeling assumption behind the corresponding bound.

The formal guarantees rely on three modeling assumptions:

TABLE IX: Mapping between formal security properties, assumptions, and empirical evidence.

| Thm. | Property | Key assumption | Empirical evidence |
|---|---|---|---|
| 1 | $(k, \delta)$-anonymity | $\eta$-bounded side info | §V-D: 94.1% w/o budget |
| 2 | $\epsilon$-unlinkability | Uniform on support | $\epsilon \le \tau(o)$ via budget |
| 3 | Composition bound | Cost calibration $w_f \ge \ln b_f$ | §V-G: NL=1.0 |

- *Bounded side information within equivalence classes* (Theorem 1). The likelihood-ratio bound (15) limits how strongly auxiliary knowledge can skew the posterior within an equivalence class. When $\eta = 1$, the classical $1/k$ guarantee is recovered exactly.
- *Uniform overlap-pattern distribution* (Theorem 2). Cross-session linkage reduces to distinguishing overlap attribute patterns once fresh IDs remove identifier-based matching. The uniform assumption yields the clean bound $\epsilon = \ln |\mathcal{Z}_U|$; skewed distributions can only worsen it.
- *Field-cost calibration* (Theorems 2 and 3). The condition $w_f \ge \ln b_f$ ensures that each field cost upper-bounds the logarithmic refinement induced by disclosing that field. This links the unlinkability and composition results to the single budget parameter $\tau(o)$.

## APPENDIX C
## LIMITATIONS

We summarise the boundary of what PLANTWIN does and does not guarantee, separating architectural properties from deployment-dependent expectations.

**Identifiability of projected fields.** The typed-graph abstraction architecturally guarantees no raw content reaches the cloud planner, but it preserves a coarse structural fingerprint (semantic class, contains-tag set, object cardinality) that can be re-identifying when the candidate population is small or auxiliary public knowledge is rich. Our re-identification study (§V-D) shows top-1 attack accuracy of 88.3% even at $k = 30$. Two specific failure modes are: (i) a rare `semantic_class` acting as a near-unique signature; (ii) a distinctive combination of `contains:*` tags creating a fingerprint even when no single tag is itself rare. Mitigations are deployment choices, not architectural properties.

**Cost model is a heuristic, not a formal bound.** The weighted field-cost model in §IV-D is a deliberately simple aggregation. It does not derive weights from a mutual-information estimate, does not bound cumulative leakage in nats or bits, and does not provide differential-privacy guarantees. The model has measurable robustness properties (sensitivity sweep referenced in the response letter) and aligns with operator intuition about field sensitivity, but lifting it to a formal bound, e.g., MI estimation against a learned attribute-inference model or DP composition over per-turn disclosures, is concurrent and complementary work.

**Local TCB soundness dependency.** The SND = 1.0 claim depends on three local components being correctly implemented: (i) the projection pipeline correctly identifying and redacting all sensitive substrings before constructing the typed graph; (ii) the gatekeeper correctly classifying capability requests against the schema and policy table; (iii) the budget

manager correctly accumulating per-object disclosure costs across turns. A bug or misclassification in any of these components can leak sensitive content despite the architectural boundary. Our evaluation does not include a TCB audit; deployments concerned with stronger assurance should add per-output redaction validators, gatekeeper unit tests over an enumerated capability set, and budget-manager invariants verified at runtime.

**Benchmark scope.** The 60 tasks are synthetic and team-authored. They model artifact-mediated triage, cross-source reconciliation, and constraint-extraction workflows, but do not capture: (i) multi-user concurrency and cross-user disclosure-budget interactions; (ii) long-horizon memory and cumulative budget across sessions; (iii) human-in-the-loop iterative refinement under realistic interactive load; (iv) adversarial provenance (prompt-injected or planted artifacts arriving from outside the agent's trust boundary); (v) real human users (we did not run a user study with practicing engineers, so the task distribution may underweight messy real-world inputs). Reproducing our results in a deployed multi-user system is worthwhile follow-on work.

**Cross-vendor planner transferability.** We evaluate four cloud planners (Kimi K2.5, MiniMax M2.5, GLM 5, Gemini 3 Flash). PLANTWIN's typed-graph projection produces planner-agnostic input, so we expect, but did not empirically validate, that the same SND, PQS, and budget properties transfer to Western planners (e.g., GPT-4-class, Claude-3.5-class). The risk of non-transfer is low for the privacy properties (which depend on what is sent, not who receives it) and uncertain for the planning-quality numbers (which depend on planner reasoning ability).