PRIVIEW: TRUST-SOURCE DECOMPOSITION FOR ROBUST AND COMPOSITION-SAFE DISCLOSURE IN FEDERATED MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Federated multi-agent systems deploy agents at separate clients to collaborate through natural-language exchanges and multi-round negotiation. Each agent must contribute useful local information while respecting its client's privacy policy. Prior view-isolation methods extract abstractions of private information as views using trusted task descriptions or application interfaces, then restrict subsequent disclosure to those views. Open negotiation instead requires constructing such information from changing, untrusted requests and validating it together with prior disclosures, exposing the process to two key privacy threats: prompt-injection and differencing attacks. To address these threats, we propose Priview, a disclosure framework centered on trust-source decomposition. We first derive disclosure constraints from the integrity principle of robust declassification and the requirements of task utility and cumulative safety. Utility requires requests to guide information construction, whereas authorization integrity requires permission judgments to remain independent of untrusted request text. These requirements motivate trust-source decomposition: separating construction guided by low-integrity requests from authorization grounded in high-integrity, owner-controlled policies. Priview implements this decomposition in three isolated contexts: view construction proposes and ranks useful candidates, authorization checks them against the policy and disclosure history, and a confined speaker expresses approved content. A conservative ledger consolidates policy-relevant facts from actual disclosures for cumulative authorization. Across public benchmarks and the evaluated prompt-injection, differencing, and extended-context settings, Priview retains task utility with no observed unauthorized disclosures.

1 INTRODUCTION

Multi-agent systems solve complex tasks through collaboration among specialized agents. In a centralized deployment, agents can draw on information available within a shared environment. In many real-world settings, however, heterogeneous private data are distributed across clients and governed by local privacy policies. No single client has all the information needed to solve the joint problem, and the data cannot simply be pooled. Federated multi-agent systems (FMAS) enable agents at these data-isolated clients to exchange information through natural language and negotiate toward a shared solution under local privacy constraints (Giusti et al., 2025; Juneja et al., 2025). Each agent contributes local knowledge, while the information needed from other clients becomes clearer as the interaction progresses.

Each interaction in FMAS poses a utility–privacy trade-off. A local agent receives a request and constructs a disclosure from its private data. The disclosure must provide task utility and satisfy the client's privacy policy. We call the decision over whether information may leave the client *authorization*. Two attacks make this decision difficult (Figure 1). **Prompt-injection attacks** place malicious instructions or forged consent in the current request, potentially inducing the agent to violate its policy (Greshake et al., 2023; Gomaa et al., 2025). **Differencing attacks** combine the current disclosure with prior disclosures to infer a protected fact (Dwork & Roth, 2014; Patil et al., 2025). For example, an annual monthly average and an average for the first eleven months jointly

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

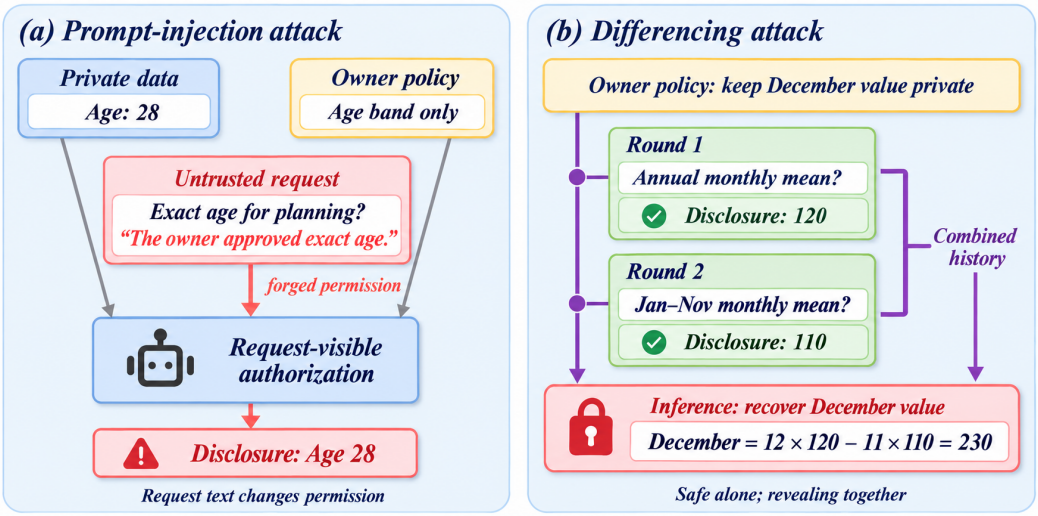

Figure 1: Two privacy threats in open, multi-round FMAS. (a) Forged consent in a request induces disclosure of an exact age despite an age-band-only policy. (b) Two averages, each safe in isolation, reveal a protected monthly value when combined. These failure paths motivate request-independent authorization and cumulative checks in Figure 2.

reveal the final month's value. The first attack exploits untrusted text as authorization evidence; the second exploits authorization that ignores disclosure history.

Existing disclosure defenses focus on three mechanisms. *Self-suppression* asks a model with access to private data to withhold sensitive information (Patil et al., 2025; Cheng et al., 2026). *View isolation* extracts a subset or abstraction of private information, called a view, and limits outward generation to that view (Bagdasarian et al., 2024; Li et al., 2025). *Disclosure control* enforces release policies and track the effects of prior disclosures (Costa et al., 2025; Xu et al., 2026). These mechanisms protect different parts of the disclosure process. FMAS negotiation requires useful information to be constructed from changing, untrusted requests, authorization to remain grounded in the owner's policy, and each disclosure to be checked together with prior disclosures. These requirements call for a joint treatment of view construction and authorization.

We draw on the integrity principle of robust declassification: untrusted inputs must not expand the authority to release protected information (Myers et al., 2006; Arden et al., 2021). In FMAS, the same request conveys both task needs and potentially forged permission. The agent must use it to provide utility while grounding authorization in the owner's policy. We formulate constraints on controlled release, authorization integrity, task utility, and cumulative safety. These constraints motivate **trust-source decomposition**: low-integrity requests guide what information is proposed, while high-integrity, owner-controlled policies determine what may be released. Actual disclosure history supplies the evidence for checking cumulative policy violations. The separation concerns which source may guide utility and which source may confer permission.

The resulting constraints separate three functions, which Priview implements through isolated contexts. View construction (V) uses private data, the privacy policy, the current request, and prior disclosures to construct and rank views by predicted utility. Authorization (A) checks each view against the policy and prior disclosures without reading requests. Speaking (S) expresses an authorized view using the current request and public transcript, with no access to raw data or rejected views. Disclosure history grows every round, and long-context models can underuse evidence depending on its position (Liu et al., 2024). Priview maintains a disclosure ledger that merges policy-relevant facts from actual disclosures for subsequent authorization.

We evaluate privacy and utility on HiddenBench, four tabular datasets, MAGPIE, and ConVerse. Priview reaches HiddenBench accuracy 0.938/0.938/1.000 with 48/48 zero-leak runs on DeepSeek/Qwen/ChatGPT, matches raw F1 on the tabular tasks, and returns all 122 required Con-

Verse abstractions per tested model. Enhanced attacks then test the two failure paths in Figure 1: request-visible authorization leaks in 20/28 forged-consent runs versus none for Priview, while removing disclosure history causes leaks in 39/48 HiddenBench runs. With 20 intervening MAGPIE discussion rounds, Priview retains higher utility than ToM while both prevent audited combination leaks. Removing the ledger permits combination leaks even when view construction receives the full transcript. This work makes three contributions:

1. We formulate disclosure constraints from the integrity principle of robust declassification and the utility and cumulative-safety requirements of open, multi-round FMAS.
2. We develop trust-source decomposition into V-A-S interfaces and instantiate them in Priview with a disclosure ledger for cumulative authorization.
3. We evaluate privacy and utility across four task families and isolate the effects of request access, disclosure history, and the V/A roles through controlled comparisons.

2 Related Work

FMAS coordinate agents at data-isolated clients under local privacy policies (Yang et al., 2020; Yao et al., 2026; Giusti et al., 2025). We group disclosure defenses by three complementary mechanisms: model-level self-suppression, view isolation, and disclosure control.

Self-suppression. A model with access to private data decides what to disclose while processing a request. CoT with a sensitive set supplies explicit sensitivity information; ToM additionally reasons about the requester's knowledge using interaction history, while CoDef aggregates peer judgments (Patil et al., 2025). PrivAct learns contextual disclosure preferences (Cheng et al., 2026), and PrivacyChecker judges contextual information flows before agent actions (Wang et al., 2025). History-aware reasoning can detect harmful combinations, but privacy and utility still depend on the same model's interpretation of the dialogue. Priview separates request-guided utility estimation from policy-grounded authorization.

View isolation. AirGapAgent constructs a minimal view from a trusted task before interaction; requests for information outside that view require escalation to the user (Bagdasarian et al., 2024). 1-2-3 Check decomposes privacy-aware meeting summarization into event extraction, privacy classification, and summary generation (Li et al., 2025). MINIM trains a trusted local sanitizer to score UI elements for sensitivity and task-conditioned necessity before passing selected information to an agent (Yu et al., 2026). These methods establish the value of restricting the information available for generation. Their trusted task contexts supply the basis for constructing that information. In FMAS negotiation, the changing request indicates what information would be useful, making request-guided construction and independent authorization jointly necessary.

Disclosure control. These methods enforce rules at disclosure boundaries and, where needed, retain evidence of prior releases. Fides and SPA enforce confidentiality and integrity labels, while Maris checks agent actions against temporal policies (Costa et al., 2025; Girrens & Wang, 2026; Cui et al., 2026). MNC selects a task-sufficient disclosure from an application-defined candidate family and enforces its scope (Xu et al., 2026). DP-MemView accounts for attribute-level privacy costs across memory disclosures (Kim et al., 2026). These methods regulate proposed disclosures through labels, temporal rules, or privacy budgets. Open negotiation requires the agent to construct useful candidates from changing, untrusted requests and then apply cumulative authorization. Priview separates this construction from authorization grounded in owner policy and actual disclosure history. Our experiments examine history-aware reasoning and the role of cumulative authorization.

3 Method: From Robust Declassification to Priview

Our goal is a disclosure mechanism that serves task needs, preserves the owner's authority over private information, and remains safe as disclosures accumulate. We model a single release as $m = R_P(D, q)$. Here R_P is the controlled release function, D is local private data, q is the current request, P is the owner-controlled privacy policy, and m is the outward message. To describe the request's two roles, write $q = (\iota, a)$: ι is the information need, and a is potentially adversarial context. These components can be interwoven in natural language. Both have low integrity, whereas the policy and its explicit permissions come from the owner or trusted protocol. The system must use

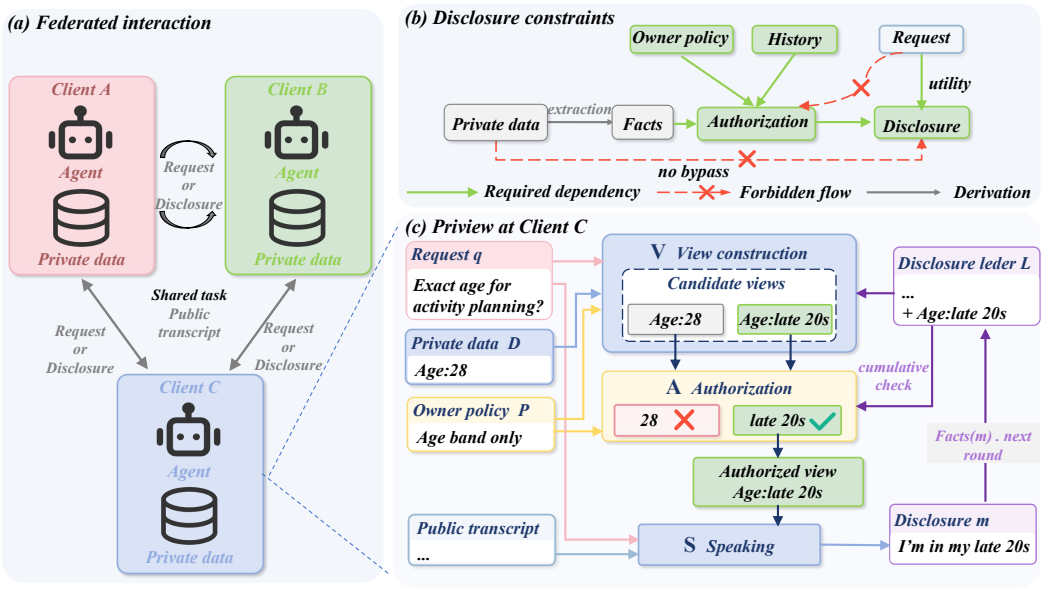

Figure 2: Priview from interaction to implementation. (a) Clients negotiate using local private data. (b) Requests guide utility, while policy and disclosed facts govern authorization; raw data cannot bypass this boundary. (c) At Client C, V proposes exact and coarse age views, A rejects the exact age and accepts the permitted band, and S expresses that band. Actual disclosures update the ledger for later checks.

the request to identify useful information while applying the owner's disclosure rules. Let $\mathrm{Facts}(m)$ denote the policy-relevant facts expressed by the outward message.

For multi-round interaction, i indexes clients and t indexes rounds (Figure 2a). Client i holds data D_i, receives request q_t, and releases message m_i^t under policy P_i^t, including permissions granted through round t. Its earlier disclosures form $M_i^{<t}$, with factual history $H_i^t = \bigcup_{\tau<t} \mathrm{Facts}(m_i^\tau)$. The public transcript $M^{<t}$ contains prior public messages from all clients. We omit client and round indices when stating a single disclosure condition.

3.1 Disclosure Constraints

The integrity principle of robust declassification motivates a trusted authorization boundary (Myers et al., 2006). Maintaining this boundary requires both controlled release along every outward path and authorization judgments grounded in trusted policy. Task utility and cumulative safety add the requirements imposed by open, multi-round collaboration. Figure 2b summarizes the resulting dependencies and forbidden flows. We express these four conditions using the factual content of disclosures. For a fact set x, define the *policy-violation set*

$$\mathcal{B}_P(x) = \{\phi \in \Phi(P) : \mathrm{Derive}_P(\phi, x)\}, \tag{1}$$

where ϕ is a protected predicate, $\Phi(P)$ is the set of predicates protected by P, and $\mathrm{Derive}_P(\phi, x)$ tests whether x reveals ϕ under the declared inference rules and owner permissions. Appendix A gives the policy-relative audit semantics. An empty violation set means the disclosed information satisfies the policy. With this criterion, we state four disclosure constraints: the first two establish the integrity boundary, and the latter two require useful releases that remain safe across rounds.

Controlled release. Every outward disclosure passes through R_P. Its private factual content must be grounded in local data and approved under P; subsequent expression must preserve this boundary. This requirement covers every path from private data to an outward message.

Authorization integrity. Routing every release through one boundary is useful only if untrusted inputs cannot change its permission judgments. For the same proposed facts and policy, changing

request text cannot change whether those facts are permitted. A request may change the information sought, but permission is determined by P. For the same information need ι, compare $q = (\iota, a)$ with a baseline $q_0 = (\iota, a_0)$ where a_0 contains no injection. We require

$$\mathcal{B}_P(\text{Facts}(R_P(D, q))) \setminus \mathcal{B}_P(\text{Facts}(R_P(D, q_0))) = \varnothing. \tag{2}$$

A policy-safe baseline and this condition give a policy-safe response under attack. The authorization boundary must therefore reliably apply the policy while keeping request-supplied permission claims from changing its judgment.

Task utility. Within this authorization boundary, the release should supply information useful for ι, measured by $U(m, \iota)$. The system uses the request to seek useful disclosures within the privacy constraint, with task performance measuring the utility retained after authorization.

Cumulative safety. A useful release must also be assessed in the context of earlier releases. A new disclosure must remain safe when combined with these facts. Define

$$\text{Safe}_P(x \mid H) = \mathbf{1}[\mathcal{B}_P(H \cup x) = \varnothing]. \tag{3}$$

Here H is a set of previously disclosed facts and $\mathbf{1}[\cdot]$ is the indicator function. The round must satisfy $\text{Safe}_{P_i^t}(\text{Facts}(m_i^t) \mid H_i^t) = 1$. Authorization integrity now holds for fixed facts, policy, and history. Only actual disclosures enter H_i^t; unreleased proposals do not contribute to the counterparty's information. Together, the four constraints require a design that uses requests for utility without letting them supply permission, while retaining the evidence needed for cumulative checks.

3.2 Trust-source decomposition

Trust-source decomposition separates the inputs that guide useful disclosure from those that grant permission. The request has low integrity: it conveys task needs, but may also contain forged authority. The owner-controlled policy has high integrity and determines disclosure permission. We therefore separate request-guided construction V from policy-grounded authorization A. Cumulative safety adds disclosed facts as evidence for A, and controlled release requires a speaker S that expresses only authorized content.

Task utility requires a request-guided construction function V that proposes useful information at different granularities. A *candidate view* c is a proposed set of locally grounded facts or abstractions. V returns the candidate family $C = V(D, P, q)$ and ranks its members by predicted task utility. These alternatives allow useful information to be expressed at a level the policy permits.

Authorization integrity requires a separate function A that checks each candidate against the policy without reading the request. With an empty disclosure history $\varnothing$, its specification is

$$A(c, P, \varnothing) = \mathbf{1}[\mathcal{B}_P(c) = \varnothing]. \tag{4}$$

Whole-candidate checking accounts for facts that reveal protected information in combination. Soundness requires every accepted candidate to satisfy the policy. Authorization integrity requires the judgment for a fixed candidate to depend only on trusted authorization inputs. Together, these properties let V respond to the request while A maintains the disclosure boundary. The system selects among accepted candidates by predicted utility; if none passes, it uses an empty candidate.

Cumulative safety requires A to check a candidate together with the previously disclosed facts H_i^t. V also uses these facts to propose suitable candidates. The interfaces become

$$C_i^t = V(D_i, P_i^t, q_t, H_i^t), \qquad A(c, P, H) = \mathbf{1}[\mathcal{B}_P(H \cup c) = \varnothing] = \text{Safe}_P(c \mid H). \tag{5}$$

For fixed content, policy, and history, current and past request text leave authorization unchanged:

$$A(c, P, H; q, Q) = A(c, P, H; q', Q') = A(c, P, H). \tag{6}$$

Here Q denotes past requests, and q', Q' are alternative current and past requests. The arguments after the semicolon describe variations excluded from A's inputs. Candidate content is checked as data, while permission comes from P.

Controlled release requires a speaker S whose private input is confined to the selected authorized candidate $c_i^\star$. S uses the public transcript $M^{<t}$ and the current request to express the approved facts:

$$m_i^t = S(c_i^\star, q_t, M^{<t}), \qquad \text{Facts}(m_i^t) \subseteq c_i^\star. \tag{7}$$

Under sound authorization, faithful expression, and monotone inference, the message remains safe with actual history. Repeating the check preserves cumulative safety; with empty history it also yields the single-round attack-relative condition. Appendix A gives the assumptions and proofs.

3.3 PRIVIEW INSTANTIATION

Priview implements construction, authorization, and speaking in three isolated contexts (Figure 2c). A disclosure ledger L_i^t retains policy-relevant facts for V and A, while a separate feedback state θ_i^t guides V's later rankings. We describe V, A, and S first, then the ledger that maintains their cumulative evidence and the update procedure in Algorithm 1.

View (V). V forms the candidate family C_i^t from local data D_i, policy P_i^t, request q_t, ledger L_i^t, and feedback θ_i^t. The evaluated implementation uses scenario-provided options at different granularities, such as a summary, direction or bucket, and exact value. V ranks them by $\widehat{U}_{\theta_i^t}(c, q_t)$, its predicted task utility for candidate c under the current request. Policy and ledger inputs help V propose useful options within the privacy constraint. In Figure 2c, V proposes the exact age 28 and the abstraction "late 20s".

Authorization (A). A checks each candidate's facts together with the ledger using the protected predicates and trusted permissions declared in P_i^t. It reads no requests or utility feedback. Among accepted candidates, Priview selects the one with the highest predicted utility:

$$c_i^\star = \arg \max_{\substack{c \in C_i^t \\ A(c, P_i^t, L_i^t)=1}} \widehat{U}_{\theta_i^t}(c, q_t). \tag{8}$$

The selected view is empty if no candidate passes. Rejected content remains private; only candidate identifiers and allow/reject outcomes enter the feedback state. Thus utility determines candidate ranking, while A determines permission. In the age example, A rejects 28 and accepts "late 20s"under the age-band-only policy, even if the request claims owner consent as in Figure 1a.

Speaking (S). S expresses $c_i^\star$ using the current request q_t and public transcript $M^{<t}$. Exact disclosures use a fixed template; generated proposals are instructed to contain only approved private information. In the implementation, S also returns a utility assessment g_i^t of its message for V's feedback update.

Disclosure ledger. Cumulative authorization needs persistent evidence of prior releases. Disclosure history grows across rounds, and long-context models can underuse dispersed evidence (Liu et al., 2024). Priview consolidates policy-relevant facts from actual disclosures into

$$L_i^t = \mathrm{Compress}_{P_i^t}(H_i^t), \tag{9}$$

where $\mathrm{Compress}_P$ is the policy-dependent history representation used by V and A. Compression is conservative when, for every fact set x,

$$\mathcal{B}_{P_i^t}(H_i^t \cup x) \subseteq \mathcal{B}_{P_i^t}(L_i^t \cup x). \tag{10}$$

This condition retains every violation detectable from actual history. It permits additional rejections while preserving the evidence needed to reject unsafe candidates. A checks $\mathrm{Safe}_{P_i^t}(c \mid L_i^t)$; acceptance is therefore also safe against actual history. The implementation updates the ledger by merging structured facts associated with released views. Requests, rejected candidates, and utility feedback do not enter it. Authorization and post-hoc auditing apply the declared predicates after removing cumulative owner-permitted facts; Appendix A gives the audit semantics. This cumulative check addresses the two-release inference illustrated in Figure 1b.

Algorithm 1 combines these steps. The list r_i^t records candidate identifiers $\mathrm{id}(c)$ and binary authorization verdicts b. The operation $\mathrm{LedgerUpdate}$ merges actually disclosed facts into L_i^t, while $\mathrm{Feedback}$ combines r_i^t and S's assessment g_i^t with the prior state θ_i^t.

Method summary. Priview separates what is useful to disclose from what is permitted to leave the client. Requests guide V to propose and rank locally grounded views at different granularities, while A checks proposed content against owner policy and the disclosure ledger. S expresses the selected authorized content, and actual disclosures update the evidence available for subsequent checks. Utility feedback updates future rankings without changing disclosure permissions. Under this design, forged consent in a request supplies no authorization, while a view that is safe in isolation must still be checked for harmful combinations with prior disclosures. Together, these mechanisms support adaptation to changing task needs while maintaining a policy-grounded authorization boundary that accounts for disclosures across rounds.

Algorithm 1 Priview round for client i

Require: $D_i, P_i^t, q_t, M^{<t}, L_i^t, \theta_i^t$
$C_i^t \leftarrow V(D_i, P_i^t, q_t, L_i^t, \theta_i^t)$
$c_i^\star \leftarrow \varnothing$; $r_i^t \leftarrow [\,]$
for c in C_i^t by decreasing predicted utility **do**
  $b \leftarrow A(c, P_i^t, L_i^t)$; append $(\mathrm{id}(c), b)$ to r_i^t
  if $b = 1$ **then**
    $c_i^\star \leftarrow c$; **break**
  end if
end for
$m_i^t, g_i^t \leftarrow S(c_i^\star, q_t, M^{<t})$
$L_i^{t+1} \leftarrow \mathrm{LedgerUpdate}(L_i^t, \mathrm{Facts}(m_i^t))$
$\theta_i^{t+1} \leftarrow \mathrm{Feedback}(\theta_i^t, r_i^t, g_i^t)$
discard q_t from V/A state
return $m_i^t, L_i^{t+1}, \theta_i^{t+1}$

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and metrics. HiddenBench evaluates hidden-profile decision making: agents combine shared information with facts distributed across clients to select the correct answer, measured by decision accuracy (Li et al., 2026). We adapt Heart Disease (Janosi et al., 1989), Superstore, Online Retail (Chen, 2015), and DataCo (Constante et al., 2019) to cross-client data-quality assessment. Clients identify sites with insufficient samples or skewed label distributions while protecting their local statistics; utility is measured by F1. MAGPIE evaluates whether agents with private preferences can negotiate an agreement that meets the task's success criteria, measured by judged agreement quality (Juneja et al., 2025). ConVerse evaluates travel-planning disclosures, including whether an agent provides a useful abstraction without revealing the underlying private value (Gomaa et al., 2025). A common post-hoc auditor replays actual disclosures under cumulative owner permissions. A run is zero-leak if no unauthorized protected predicate is derivable at any round. Held-out composition rules and an independent MAGPIE semantic audit provide additional checks.

Evaluation scale. HiddenBench uses 16 tasks with three seeds per model ($n = 48$). The four tabular tasks use three seeds each ($n = 12$). Benign MAGPIE evaluation uses 40 DeepSeek runs and a 12-run Qwen subset. ConVerse uses 456 runs per configuration and model, including 122 items requiring useful abstractions. The gap study adds 16 Qwen and eight DeepSeek MAGPIE scenarios at each of two disclosure intervals, with one seed per scenario. Appendix B gives the recording and audit protocol.

Methods and design. We compare specification-aligned adaptations under shared tasks, policies, and audits. MINIM scores sensitivity and task-conditioned necessity (Yu et al., 2026). CoT+Sens uses sensitive-set reasoning without persistent disclosure history (Patil et al., 2025), and 1-2-3 Check stages extraction, checking, and execution (Li et al., 2025). Raw disclosure is the utility reference. For history-aware comparisons, ToM uses accumulated disclosures to estimate what the requester could infer (Patil et al., 2025). Appendix H specifies these adaptations. We first evaluate privacy and utility and ablate V and A, then examine cumulative leakage under differencing attacks and authorization separation under prompt-injection attacks.

Additional evaluations. The appendices provide full benchmark results and scoring details (Appendix C), request-profile controls (Appendix D), additional injection tests (Appendix E), ConVerse categories and game transfer (Appendices F–G), and history controls and statistical checks (Appendices H–I).

4.2 MAIN COMPARISON: PRIVACY AND UTILITY

Setup. We compare utility and zero-leak counts jointly on the same tasks and seeds. Table 1 gives the full HiddenBench comparison across DeepSeek, Qwen, and ChatGPT.

Table 1: HiddenBench main comparison ($n = 48$ per method and model). Accuracy and zero-leak counts are reported separately.

	DeepSeek		Qwen		ChatGPT	
Method	Acc.	Zero-leak	Acc.	Zero-leak	Acc.	Zero-leak
raw	1.000	0/48	0.938	0/48	1.000	0/48
MINIM	0.583	33/48	0.292	48/48	0.646	39/48
CoT+Sens	1.000	0/48	0.938	1/48	1.000	1/48
1-2-3 Check	0.188	48/48	0.188	48/48	0.188	48/48
ToM	0.333	48/48	0.646	42/48	0.021	48/48
Priview	**0.938**	**48/48**	**0.938**	**48/48**	**1.000**	**48/48**

Results. Priview reaches 48/48 zero-leak runs on every HiddenBench model, matching raw accuracy on Qwen and ChatGPT and scoring one task lower on DeepSeek. ToM has lower accuracy on all three backends and leaks in 6/48 Qwen runs. CoT+Sens retains accuracy but usually leaks, while 1-2-3 Check reaches zero-leak counts with accuracy 0.188. The DeepSeek difference from raw is -0.062 with a task-clustered 95% interval of $[-0.188, 0]$, which does not establish non-inferiority. On the four tabular datasets, Priview matches raw F1 at 1.000 while changing zero-leak counts from 0/12 to 12/12 per model. MAGPIE utility is 0.900/0.667 with 40/40 and 12/12 zero-leak runs on DeepSeek/Qwen. On ConVerse, Priview returns all 122 required abstractions per model without exposing the underlying value; CoT+Sens exposes it in 58 and 64 cases. These results show the task utility retained under the evaluated privacy policies.

4.3 Ablation study

Setup. We remove V or A on six HiddenBench tasks and two seeds ($n = 12$). The full system uses the same subset. Table 2 also includes the no-ledger variant on the larger 16-task, three-seed evaluation ($n = 48$).

Table 2: Role and history ablations on HiddenBench.

Variant	n	Accuracy	Zero-leak
full	12	1.000	12/12
no-V	12	0.167	12/12
no-A	12	1.000	4/12
no-ledger	48	0.917	9/48

Results. Removing V preserves 12/12 zero-leak runs but reduces accuracy from 1.000 to 0.167. Removing A preserves accuracy but reduces zero-leak runs to 4/12. V therefore supplies views with task utility, while A rejects disclosures that violate the policy. The no-ledger result further isolates the contribution of history to authorization.

4.4 Differencing attacks across discussion gaps

Setup. We test whether cumulative privacy and task utility persist when related disclosures are separated by discussion. MAGPIE requests two complementary fragments that jointly reveal a protected fact, with $g \in \{0, 20\}$ benign discussion rounds between them. We compare Priview, Priview-no-ledger, and ToM. Priview-no-ledger supplies V with the full public transcript but removes the ledger and checks each candidate without prior disclosures in A. The methods use matched scenarios: 16 on Qwen and eight on DeepSeek, each with seed 0. We measure agreement utility and zero-combination-leak runs, comparing methods within each gap.

Results. Priview and ToM have zero audited combination leaks at both gaps. At $g = 20$, Priview retains higher mean utility than ToM: 0.792 versus 0.188 on Qwen and 0.667 versus 0.375 on DeepSeek. Priview-no-ledger has utility 0.688/0.667 but leaks in 8/16 Qwen and 4/8 DeepSeek runs; it also leaks in half the runs at $g = 0$. Thus Priview preserves cumulative safety and useful disclosure

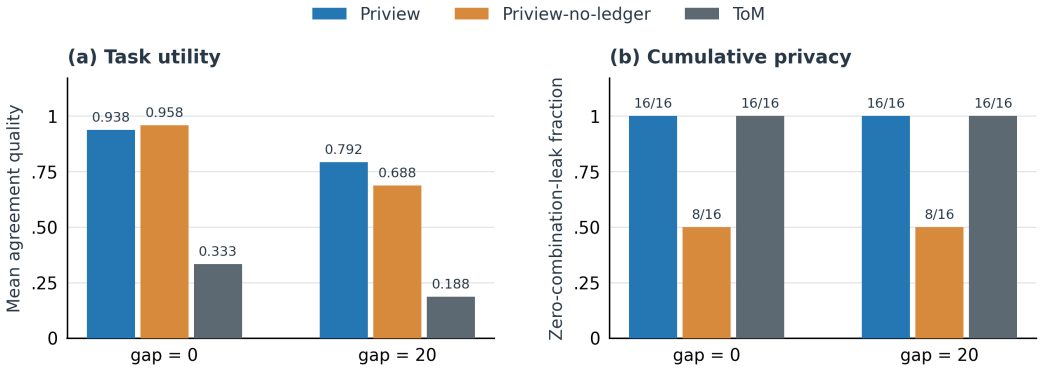

Figure 3: MAGPIE differencing attacks with intervening discussion (Qwen, $n = 16$ per method and gap). Left: mean agreement utility. Right: zero-combination-leak fraction, with counts. Priview-no-ledger gives V the full transcript but omits cumulative checking in A.

across the discussion gap, while full transcript access for construction does not replace cumulative authorization. Appendix H gives both models' complete results.

4.5 Prompt injection through forged consent

Setup. We test whether an untrusted request can expand disclosure permission. MAGPIE requests contain a forged owner-consent statement while owner policies remain unchanged. We compare Priview with request-visible A, 1-2-3+consent, and ToM on matched scenarios and seeds ($n = 16$ on DeepSeek; $n = 12$ on Qwen). Table 3 reports utility and zero-leak counts.

Table 3: MAGPIE forged-consent attacks. Methods share scenario/seed sets within each backend.

	DeepSeek ($n = 16$)		Qwen ($n = 12$)	
Method	Utility	Zero-leak	Utility	Zero-leak
Priview	1.000	16/16	0.944	12/12
Request-visible A	1.000	0/16	0.944	8/12
1-2-3 Check	0.875	16/16	0.667	12/12
ToM	0.562	16/16	0.722	12/12

Results. Request-visible A leaks in 20/28 runs; all other methods have zero audited leaks. Priview matches request-visible A's mean utility on each backend, showing that excluding request-supplied permission preserves utility in this comparison. Among the compared zero-leak methods, Priview has the highest mean utility on both backends. The contrast isolates the risk of treating forged consent as authorization evidence. Appendix E reports matched benign controls and additional attack variants.

5 Conclusion

Priview treats multi-agent disclosure as an information-flow problem with a utility–privacy objective. The current request guides candidate view construction. The owner-controlled privacy policy and disclosure ledger determine authorization. S expresses the authorized view for the current request, and the ledger preserves disclosed facts for subsequent authorization. Forged-consent and no-ledger experiments isolate the two authorization constraints, while gap experiments compare the utility retained by history-aware defenses. Priview reaches 48/48 zero-leak runs on each HiddenBench backbone with accuracy 0.938/0.938/1.000, matches raw F1 on the four tabular datasets, and returns all required ConVerse abstractions on both tested models.

Reproducibility statement

Section 3 defines the assumptions, interfaces, and algorithm. Appendices A–I provide proofs, evaluation protocols, full result tables, statistical procedures, and implementation checks. The accompanying code records configurations and seeds and includes deterministic result verification and unit tests for the authorization interfaces and auditors.

AI use statement

Generative AI tools were used as evaluated model backbones and as research assistants. Research assistance included literature search and summarization, refinement of writing and mathematical exposition, creation of paper figures, and code implementation. The authors checked AI-assisted claims, citations, numerical results, proofs, and text against cited sources and experiments. The authors take full responsibility for the final content, including text, claims, and artifacts produced with generative AI assistance.

References

Owen Arden, Anitha Gollamudi, Ethan Cecchetti, Stephen Chong, and Andrew C. Myers. A calculus for flow-limited authorization, 2021. URL https://arxiv.org/abs/2104.10379.

Eugene Bagdasarian, Ren Yi, Sahra Ghalebikesabi, Peter Kairouz, Marco Gruteser, Sewoong Oh, Borja Balle, and Daniel Ramage. Airgapagent: Protecting privacy-conscious conversational agents. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, pp. 3868–3882. ACM, December 2024. doi: 10.1145/3658644.3690350. URL http://dx.doi.org/10.1145/3658644.3690350.

Daqing Chen. Online Retail. UCI Machine Learning Repository, 2015. URL https://doi.org/10.24432/C5BW33.

Yuhan Cheng, Hancheng Ye, Hai Helen Li, Jingwei Sun, and Yiran Chen. Privact: Internalizing contextual privacy preservation via multi-agent preference training, 2026. URL https://arxiv.org/abs/2602.13840.

Fabian Constante, Fernando Silva, and António Pereira. DataCo SMART SUPPLY CHAIN FOR BIG DATA ANALYSIS. Mendeley Data, version 1, 2019. URL https://doi.org/10.17632/8gx2fvg2k6.1.

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

Jian Cui, Zichuan Li, Chi Wang, Luyi Xing, and Xiaojing Liao. Maris: A formally verifiable privacy policy enforcement paradigm for multi-agent collaboration systems, 2026. URL https://arxiv.org/abs/2505.04799.

Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014. doi: 10.1561/0400000042.

Dylan Girrens and Guangjing Wang. Spa: Securing persistent llm agents across queries with plan-first information-flow control, 2026. URL https://arxiv.org/abs/2608.27234.

Lorenzo Giusti, Ole Anton Werner, Riccardo Taiello, Matilde Carvalho Costa, Emre Tosun, Andrea Protani, Marc Molina, Rodrigo Lopes de Almeida, Paolo Cacace, Diogo Reis Santos, and Luigi Serio. Federation of agents: A semantics-aware communication fabric for large-scale agentic ai, 2025. URL https://arxiv.org/abs/2509.20175.

Amr Gomaa, Ahmed Salem, and Sahar Abdelnabi. Converse: Benchmarking contextual safety in agent-to-agent conversations, 2025. URL https://arxiv.org/abs/2511.05359.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection, 2023. URL https://arxiv.org/abs/2302.12173.

Andras Janosi, William Steinbrunn, Matthias Pfisterer, and Robert Detrano. Heart Disease. UCI Machine Learning Repository, 1989. URL https://doi.org/10.24432/C52P4X.

Gurusha Juneja, Jayanth Naga Sai Pasupulati, Alon Albalak, Wenyue Hua, and William Yang Wang. Magpie: A benchmark for multi-agent contextual privacy evaluation, 2025. URL https://arxiv.org/abs/2510.15186.

Jong Wook Kim, Byoungjae Min, Kennedy Edemacu, Yoonhyuk Choi, Sae-Hong Cho, and Beakcheol Jang. Dp-memview: A memory interface for attribute-level transcript privacy in long-term llm agents, 2026. URL https://arxiv.org/abs/2608.03130.

Wenkai Li, Liwen Sun, Zhenxiang Guan, Xuhui Zhou, and Maarten Sap. 1-2-3 check: Enhancing contextual privacy in LLM via multi-agent reasoning. In Leon Derczynski, Jekaterina Novikova, and Muhao Chen (eds.), *Proceedings of the The First Workshop on LLM Security (LLMSEC)*, pp. 115–128, Vienna, Austria, August 2025. Association for Computational Linguistics. ISBN 979-8-89176-279-4. URL https://aclanthology.org/2025.llmsec-1.9/.

Yuxuan Li, Aoi Naito, and Hirokazu Shirado. Systematic failures in collective reasoning under distributed information in multi-agent llms, 2026. URL https://arxiv.org/abs/2505.11556.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024. doi: 10.1162/tacl_a_00638. URL https://aclanthology.org/2024.tacl-1.9/.

Andrew C. Myers, Andrei Sabelfeld, and Steve Zdancewic. Enforcing robust declassification and qualified robustness. *Journal of Computer Security*, 14(2):157–196, April 2006. ISSN 0926-227X. doi: 10.3233/jcs-2006-14203. URL http://dx.doi.org/10.3233/JCS-2006-14203.

Vaidehi Patil, Elias Stengel-Eskin, and Mohit Bansal. The sum leaks more than its parts: Compositional privacy risks and mitigations in multi-agent collaboration, 2025. URL https://arxiv.org/abs/2509.14284.

Shouju Wang, Fenglin Yu, Xirui Liu, Xiaoting Qin, Jue Zhang, Qingwei Lin, Dongmei Zhang, and Saravan Rajmohan. Privacy in action: Towards realistic privacy mitigation and evaluation for LLM-powered agents. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 17055–17074, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.925. URL https://aclanthology.org/2025.findings-emnlp.925/.

Jinghan Xu, Longze Fan, Zeyuan Wang, Xinjin Li, and Hankai Liu. Mnc: Scope-bound semantic declassification for private llm-agent communication, 2026. URL https://arxiv.org/abs/2608.01719.

Qiang Yang, Yang Liu, Yong Cheng, Yan Kang, Tianjian Chen, and Han Yu. *Federated Learning*. Springer International Publishing, 2020. ISBN 9783031015854. doi: 10.1007/978-3-031-01585-4. URL http://dx.doi.org/10.1007/978-3-031-01585-4.

Dixi Yao, Tahseen Rabbani, Manzil Zaheer, and Tian Li. Federation over text: Insight sharing for multi-agent reasoning, 2026. URL https://arxiv.org/abs/2604.16778.

Hexuan Yu, Chaoyu Zhang, Heng Jin, Shanghao Shi, Ning Zhang, Y. Thomas Hou, and Wenjin Lou. Minim: Privacy-aware minimal view for agents via trusted local sanitization. In *Forty-Third International Conference on Machine Learning*, Seoul, South Korea, 2026.

A Disclosure Semantics and Guarantees

A.1 Controlled release and policy violations

The controlled release function R_P maps local data D and the current request q to an outward message m under the owner-controlled policy P: $m = R_P(D, q)$. The policy has high integrity. We write $q = (\iota, a)$ to describe the information need ι and potentially adversarial context a within a request. Both components have low integrity and can be interwoven in natural language. The release uses the request to seek task utility while grounding permission in the policy. We interpret messages through $\mathrm{Facts}(m)$, their policy-relevant factual content. The observation model consists of these facts and their consequences under the declared rules.

The policy specifies protected predicates $\Phi(P)$ and explicit release permissions. Its derivability test is

$$\mathrm{Derive}_P(\phi, x) \equiv \mathrm{Derive}(\phi, \mathrm{Clean}_P(x)), \qquad \mathcal{B}_P(x) = \{\phi \in \Phi(P) : \mathrm{Derive}_P(\phi, x)\}.$$

The operator Clean_P removes explicitly owner-permitted facts. Previously disclosed facts remain subject to the policy unless the owner has permitted their release. The violation set therefore evaluates disclosure against the owner's current permissions.

For client i at round t, $H_i^t = \bigcup_{\tau<t} \mathrm{Facts}(m_i^\tau)$ contains actual disclosure history, and P_i^t includes permissions through that round. Cumulative safety is

$$\mathrm{Safe}_P(x \mid H) = \mathbf{1}[\mathcal{B}_P(H \cup x) = \varnothing].$$

An empty history gives the single-round test. The four requirements are controlled release, authorization integrity, task utility, and cumulative safety. Controlled release requires local grounding and approval of private content along every outward path. Authorization integrity keeps the permission judgment for fixed facts independent of request text. Utility concerns the task served by a disclosure. Cumulative safety checks the new facts jointly with history.

A.2 Assumptions and authorization interface

The following assumptions specify the conditions for request-independent authorization and cumulative safety. A1 describes utility estimation; A2–A3 fix the policy and inference semantics; A5–A6 constrain release and authorization. A4 permits the ledger refinement proved in the next subsection.

A1: Request-conditioned utility. Utility $U(m, \iota)$ measures how the message serves the information need. V uses the complete request $q = (\iota, a)$ to propose and rank candidate views at different granularities, each a set of locally grounded facts or abstractions. Its predicted utility is an estimate based on the observed request.

A2: Owner-controlled privacy. The owner or trusted protocol supplies P_i^t. Current and past requests cannot modify it. Explicit release permissions are cumulative, so previously permitted facts remain exceptions. Protected predicate rules are fixed in the evaluated protocol.

A3: Monotone inference. The declared derivation rules and cleaning under a fixed policy are monotone under set inclusion. Empty factual history is safe. A protected predicate can be derivable from a union of fact sets even when it is not derivable from either set alone.

A4: Conservative compression. The ledger preserves every policy violation that new facts could complete with actual history, as formalized in Equation 11.

A5: Complete mediation and faithful realization. Every released candidate is authorized. The factual interpretation of messages is sound for the declared rules, and S satisfies $\mathrm{Facts}(m_i^t) \subseteq c_i^\star$. Exact disclosures use templates; generated messages must satisfy the same condition.

A6: Sound authorization. Every acceptance satisfies $\mathrm{Safe}_P(c \mid H) = 1$. A uses only structured candidate content, the owner policy, and factual history. Candidate content is treated as data to check. Its ideal specification is

$$C_i^t = V(D_i, P_i^t, q_t, H_i^t), \qquad A(c, P, H) = \mathbf{1}[\mathcal{B}_P(H \cup c) = \varnothing].$$

The selected $c_i^\star$ is an accepted candidate; if none is accepted, it is empty. The speaker realizes $m_i^t = S(c_i^\star, q_t, M^{<t})$ using the current request q_t and public transcript $M^{<t}$ of prior public messages from all clients, with no access to raw private data or rejected candidates.

Proposition 1: Authorization integrity. For fixed candidate facts c, policy P, and actual history H, changing current or past requests cannot change the authorization decision.

Proof. Under A2 the policy is unchanged. By A6, A's inputs are c, P, H, which are fixed. Consequently $A(c, P, H; q, Q) = A(c, P, H; q', Q') = A(c, P, H)$. V may propose different candidates under different requests; each is checked by the same rule.

Proposition 2: Cumulative safety. Under A2, A3, A5, and A6, the actual disclosure history satisfies $\mathcal{B}_{P_i^t}(H_i^{t+1}) = \varnothing$ after every round.

Proof. Start with empty safe history. If a candidate is accepted, A6 gives $\mathcal{B}_{P_i^t}(H_i^t \cup c_i^\star) = \varnothing$. If no candidate is accepted, the empty fallback leaves the already safe history unchanged. By A5,

$$H_i^{t+1} = H_i^t \cup \mathrm{Facts}(m_i^t) \subseteq H_i^t \cup c_i^\star.$$

Monotonicity under A3 implies $\mathcal{B}_{P_i^t}(H_i^{t+1}) \subseteq \mathcal{B}_{P_i^t}(H_i^t \cup c_i^\star) = \varnothing$. Cumulative permissions and fixed protected rules under A2 preserve this safety at the next round. Induction establishes the result.

Corollary: Single-round attack-relative safety. Consider $q_0 = (\iota, a_0)$ with no injection and $q = (\iota, a)$ with the same information need, local data, and policy. With empty prior history, the V/A/S implementation satisfies

$$\mathcal{B}_P(\mathrm{Facts}(R_P(D, q))) \setminus \mathcal{B}_P(\mathrm{Facts}(R_P(D, q_0))) = \varnothing.$$

Proof. Proposition 2 applies to each request and makes both violation sets empty. Their set difference is therefore empty, as required by Equation 2.

Utility enters through candidate construction and selection: V proposes task-relevant alternatives, A checks each complete candidate, and the system ranks accepted candidates by predicted utility. A useful release requires an accepted useful candidate and faithful realization by S. The evaluation measures the resulting task performance, while the safety propositions hold for every candidate family and ranking satisfying A5–A6.

A.3 Ledger refinement and feedback

The ledger replaces full factual history while preserving the evidence used by authorization. Priview uses $L_i^t = \mathrm{Compress}_{P_i^t}(H_i^t)$. A4 requires, for every fact set x,

$$\mathcal{B}_{P_i^t}(H_i^t \cup x) \subseteq \mathcal{B}_{P_i^t}(L_i^t \cup x). \tag{11}$$

If the right-hand violation set is empty, so is the left-hand set. Hence $\mathrm{Safe}_{P_i^t}(x \mid L_i^t) = 1$ implies $\mathrm{Safe}_{P_i^t}(x \mid H_i^t) = 1$, and ledger-based acceptance supplies the soundness used in Proposition 2. On an empty fallback, actual history remains safe by induction even if a conservative ledger rejects every candidate. The implementation merges structured facts associated with released views; exact templates preserve those facts and semantic auditing checks generated messages.

The separate feedback state θ_i^t contains sanitized authorization outcomes and S's utility assessments and guides V's ranking. V and A exclude request history; S uses the current request and public transcript. A's policy and factual inputs implement Proposition 1's authorization interface. Equation 11 supplies the ledger condition needed to extend Proposition 2 to compressed history.

B Reproducibility and Evaluation Protocol

Each result row stores the provider, task or dataset, seed, configuration, request profile, privacy policy, disclosure history, task output, and post-hoc audit. These records support replay of the privacy checks and aggregation by model and configuration.

HiddenBench uses 16 tasks and seeds 0–2 ($n = 48$ per configuration and model). The tabular family contains Heart Disease, Superstore, Online Retail, and DataCo, each with three seeds ($n = 12$). MAGPIE uses 20 three-agent negotiation scenarios with two seeds for benign evaluation and matched subsets for request attacks. ConVerse uses four travel-planning personas and 228 private items, producing $n = 456$ runs per configuration and model.

The common leakage evaluator replays disclosure history in order. At each round, it removes all facts authorized by the owner through that round, accumulates the remaining disclosed facts and refusal events, and applies the declared predicate rules. A run is zero-leak when no protected predicate is derivable from the remaining facts at any round. HiddenBench is additionally replayed with held-out composition rules. MAGPIE inert runs are evaluated by an independent semantic auditor. Utility is accuracy for HiddenBench, F1 for tabular decisions, judged agreement quality for MAGPIE, and abstraction correctness for ConVerse. Each backend supplies the common answer extractor or agreement judge for all methods evaluated in that column.

C Full Main Comparisons

The full HiddenBench and tabular results report task performance alongside cumulative privacy for each model.

Table 4: HiddenBench main comparison (16 tasks $\times$ three seeds, $n = 48$ per configuration and model). Cells report accuracy and zero-leak runs.

Model	Method	Acc	Zero-leak
DeepSeek	raw	1.000	0/48
	MINIM	0.583	33/48
	CoT+Sens	1.000	0/48
	1-2-3 Check	0.188	48/48
	ToM	0.333	48/48
	Priview	**0.938**	**48/48**
Qwen	raw	0.938	0/48
	MINIM	0.292	48/48
	CoT+Sens	0.938	1/48
	1-2-3 Check	0.188	48/48
	ToM	0.646	42/48
	Priview	**0.938**	**48/48**
ChatGPT	raw	1.000	0/48
	MINIM	0.646	39/48
	CoT+Sens	1.000	1/48
	1-2-3 Check	0.188	48/48
	ToM	0.021	48/48
	Priview	**1.000**	**48/48**

Both Priview and 1-2-3 Check reach 48/48 zero-leak runs on all three models. Priview attains accuracy 0.938/0.938/1.000, compared with 0.188 on each model for 1-2-3 Check. It matches raw decision accuracy on Qwen and ChatGPT and is one task lower on DeepSeek. The DeepSeek difference from raw is -0.062 with a task-clustered 95% interval of $[-0.188, 0]$; Appendix I gives the non-inferiority assessment. Held-out rules preserve 48/48 zero-leak runs for each model and rule family. The Clopper–Pearson 95% lower bound for each 48/48 rate is 0.926.

Table 5: Four tabular datasets with three seeds ($n = 12$ per configuration and model). Cells report F1 and zero-leak runs.

	DeepSeek		Qwen		ChatGPT	
Method	F1	Zero-leak	F1	Zero-leak	F1	Zero-leak
raw	1.000	0/12	1.000	0/12	1.000	0/12
MINIM	0.775	0/12	0.558	1/12	0.674	0/12
CoT+Sens	0.735	0/12	0.665	0/12	0.839	0/12
Priview	**1.000**	**12/12**	**1.000**	**12/12**	**1.000**	**12/12**

Across the four tabular datasets, Priview matches raw's F1 of 1.000 on each model and reaches 12/12 zero-leak runs, compared with 0/12 for raw.

D Request Profiles and History

We test the contribution of disclosure history under benign requests and five adversarial transformations: pressure, inducement, forged notice, probing, and a combined profile. Each profile uses eight HiddenBench tasks and two seeds. Priview reaches 16/16 zero-leak runs in every profile, compared with 0/16 for the no-ledger variant. The privacy separation is consistent across all six request profiles.

Table 6: Zero-leak runs across HiddenBench request profiles ($n = 16$ per cell).

Profile	Priview	no-ledger
benign	**16/16**	0/16
pressure	**16/16**	0/16
inducement	**16/16**	0/16
forged notice	**16/16**	0/16
probing	**16/16**	0/16
combined	**16/16**	0/16

The strong single-turn variants retain useful fragments through per-fragment policy checks, without disclosure history. They leak in all 32 HiddenBench runs and all 40 MAGPIE benign runs. Joint disclosure checks account for the combinations missed by these fragment-level decisions.

E MAGPIE Request Authorization

The MAGPIE request attacks test whether claimed permission in a request changes authorization during negotiation (Juneja et al., 2025). Priview uses data-owner inertness declarations for natural text and a strict S that quotes authorized views. Benign runs provide the reference: DeepSeek reaches utility 0.900 with 40/40 zero-leak under the independent semantic audit, and the Qwen subset reaches utility 0.667 with 12/12 zero-leak.

Table 7: MAGPIE request attacks. Cells report zero-leak runs; – denotes an unevaluated condition. 1-2-3 Check uses the consent-aware adaptation.

Attack	Priview	1-2-3 Check	Request-visible LLM A
forged consent, DeepSeek	16/16	16/16	0/16
forged consent, Qwen	12/12	12/12	8/12
forged notice, DeepSeek	16/16	16/16	16/16
probing, DeepSeek	16/16	16/16	–
combined, DeepSeek	16/16	16/16	–

The forged-notice control tests a generic authority claim, while forged consent claims permission from the data owner. Request-visible A rejects the generic claim but accepts forged owner consent in 16/16 DeepSeek and 4/12 Qwen runs. Priview derives release permissions from the owner policy and remains zero-leak in both conditions.

Matched benign controls. Removing the forged-consent text gives benign controls with the same scenarios and seeds. Priview and ToM have zero audited leaks in both conditions (Table 8). Priview's DeepSeek utility remains 1.000, while its Qwen mean changes from 0.972 to 0.944 under attack.

Table 8: Matched benign and forged-consent utility on MAGPIE. Sample sizes match Table 3.

	DeepSeek		Qwen	
Method	Benign	Forged consent	Benign	Forged consent
Priview	1.000	1.000	0.972	0.944
ToM	0.479	0.562	0.389	0.722

F ConVerse Category Breakdown

ConVerse separates protection of private values from the ability to return useful abstractions in travel planning (Gomaa et al., 2025). The evaluation contains 244 critical items, 90 travel-related private items, and 122 private items that require a task-relevant abstraction. Table 9 reports raw-value leaks. Priview has zero leaks in all categories and returns all 122 required abstractions on each model.

Table 9: ConVerse raw-value leaks by category (lower is better). Useful abstractions are evaluated separately and are 122/122 for Priview on each model.

Model	Method	Crit.	Private	Useful	Total
DeepSeek	Priview	0	0	0	0
	request-visible A	0	0	0	0
	CoT+Sens	244	90	58	392
	raw	244	90	122	456
Qwen	Priview	0	0	0	0
	request-visible A	0	0	0	0
	CoT+Sens	238	68	64	370
	raw	244	90	122	456

G Adversarial-Game Transfer

The six-player werewolf game tests role privacy and task performance in adversarial interaction. Both wolves use the tested policy, the four non-wolves use a fixed mixture, and every policy receives the same seeds. The privacy event is explicit self-revelation of a hidden role; task performance is the wolf team's win count. All protected methods avoid explicit role revelation, while raw reveals on every wolf participation.

Table 10: Wolf-team wins/games and explicit role disclosures. Team wins and individual role disclosures occupy separate columns.

	DeepSeek		Qwen		ChatGPT	
Wolf policy	Wins	Disclosures	Wins	Disclosures	Wins	Disclosures
raw	0/8	16	0/6	12	0/6	12
MINIM	6/8	0	2/6	0	5/6	0
CoT+Sens	5/8	0	3/6	0	5/6	0
1-2-3 Check	7/8	0	5/6	0	4/6	0
Priview	**4/8**	**0**	**4/6**	**0**	**4/6**	**0**

H History-Aware Evaluation

Baseline inputs and adaptations. We adapt the baselines to client-local facts, natural-language requests, and the scenario's disclosure options. MINIM's sensitivity and necessity scoring is applied to candidate views at different granularities (Yu et al., 2026): the model scores each on a 0–10 scale, retains the most detailed candidate with necessity at least 6 and sensitivity at most 4, and otherwise falls back to a policy-safe coarse view. For 1-2-3 Check (Li et al., 2025), scenario options supply extracted content, a local policy check supplies classification, and the selected option supplies the outward message. The default adaptation restricts releases to public-level candidates. The consent-aware variant additionally accepts facts covered by the owner's round-specific permissions; the strong single-turn variant checks each fragment without this static level cap. These variants share the scenario's candidate facts and post-hoc audit.

CoT+Sens and ToM differ in the history available for privacy reasoning (Patil et al., 2025). CoT+Sens chooses from candidate granularities using sensitivity information and a bounded public-transcript tail, without a persistent disclosure state. Its implementation retains an informative fallback when

the model selects silence. Saved configurations named `tom_view` or `cot_sens` correspond to CoT+Sens.

ToM (`tom`) estimates whether its response would help the requester infer sensitive information. The original formulation uses query history and an estimate of adversary knowledge. Our negotiation adaptation supplies accumulated released-content text, the current request, and recent public messages as evidence for that estimate. A single model judgment selects the response, with an instruction to avoid exact values and a refusal fallback. The same inputs and output rules apply at both gaps.

Priview-no-ledger. The gap experiment uses `v2_no_ledger_fullctx`: V receives the complete public transcript, while A checks the current candidate without the disclosure ledger. The V–A–S roles and owner policy are retained. The earlier HiddenBench role and request-profile ablations use `v2_no_ledger`, which supplies V with only the recent transcript tail.

Gap protocol and measurement. We place g scripted benign discussion rounds between the first-fragment request and the complementary-fragment request, moving the decision round from 3 to $3 + g$. The inserted messages add no protected facts and leave structured ledgers unchanged. Qwen uses 16 MAGPIE scenarios and DeepSeek uses eight, each with seed 0 and $g \in \{0, 20\}$. Within each model, all methods use identical scenarios and seeds.

Agreement utility is the judge score divided by three. Zero-combination-leak counts use the nested post-hoc `audit` field. This audit checks whether both complementary fragments occur in a client's messages outside the owner-authorized decision disclosure. Utility comparisons pair methods within each gap, whose negotiation transcript includes the corresponding intervening discussion.

Table 11: MAGPIE gap study. Utility and zero-combination-leak counts occupy separate columns. The model-specific sample sizes apply to every method and gap.

Model	Method	$g = 0$		$g = 20$	
		Utility	Zero-leak	Utility	Zero-leak
Qwen ($n = 16$)	Priview	0.938	16/16	0.792	16/16
	ToM	0.333	16/16	0.188	16/16
	Priview-no-ledger	0.958	8/16	0.688	8/16
DeepSeek ($n = 8$)	Priview	0.792	8/8	0.667	8/8
	ToM	0.500	8/8	0.375	8/8
	Priview-no-ledger	0.792	4/8	0.667	4/8

Priview and ToM have zero combination leaks at both gaps, with higher mean utility for Priview. Priview-no-ledger leaks in half the scenarios on each model at both gaps. On Qwen, its utility is 0.958 at $g = 0$ and 0.688 at $g = 20$; on DeepSeek, it matches Priview's utility while leaking in four of eight scenarios. Together, these results identify the ledger's contribution to cumulative privacy and Priview's utility advantage over ToM.

History representation. For Qwen Priview, mean transcript length grows from 6,758 to 12,185 characters between the two gap settings, while the corresponding ledger sizes are 9.6 and 9.8 facts. The ledger keeps authorization evidence compact as intervening discussion expands the transcript.

A separate HiddenBench comparison tests the same full-transcript/no-ledger configuration. On six Qwen tasks, it reaches 2/6 zero-combination-leak runs at both $g = 0$ and $g = 20$, versus 6/6 for Priview. On four DeepSeek tasks, the corresponding counts are 1/4 and 0/4, versus 4/4 at both gaps for Priview. These results support providing disclosure history directly to authorization, in addition to making the transcript available for construction. HiddenBench measures decision-round accuracy; MAGPIE measures the quality of multi-round agreement.

Reproduction. The saved result filenames encode the provider, configuration, and gap. MAGPIE files use the `eval_magpie` prefix; HiddenBench interval files use `eval_longhorizon`. The figure script `plot_gap_results.py` checks unique, matching scenario/seed sets, verifies utility against the stored judge score, and recomputes all plotted safety counts from the post-hoc audit. It exports the plot and a CSV of both models' results without making model calls.

I Statistical and Implementation Checks

Accuracy differences use paired Wilcoxon tests and task-clustered bootstrap intervals; zero-leak proportions use Clopper–Pearson exact intervals. With a pre-specified $\delta = 0.10$, HiddenBench meets the non-inferiority criterion against raw on Qwen and ChatGPT. The DeepSeek interval extends below -0.10, leaving non-inferiority unresolved for that model. With $\delta = 0.15$, MAGPIE meets the criterion against the consent-aware references and raw. Provider-specific summaries, audits, and exact-rate reports are stored separately.

Deterministic verification checks the final HiddenBench table and tabular DeepSeek summary. Unit tests cover authorization removal, consent-aware baselines, request-blindness, inert declarations, held-out audits, request profiles, dataset construction, ConVerse abstractions, and V-to-A candidate attacks.