JOHN DOE: A Privacy-Preserving Coordinator for Zero-Raw-Context LLM Inference

Uninitial Research Lab

July 2026

Abstract

Recent advances in LLM coordination demonstrate that a lightweight coordinator can orchestrate multiple frontier models to achieve super-individual performance. However, existing approaches assume full context visibility, making them unsuitable for privacy-sensitive enterprise applications. We introduce JOHN DOE (Judge-Ghost-Witness Orchestration for Private Inference), a privacy-preserving coordination system that routes tasks across frontier models while minimizing raw private context exposure. JOHN DOE replaces the traditional Thinker-Worker-Verifier paradigm with a Judge-Ghost-Witness trinity: the *Judge* classifies sensitivity and decides routing policies; the *Ghost* transforms prompts into privacy-preserving representations using entity abstraction, synthetic aliases, and alienized language; and the *Witness* verifies outputs for leakage and reconstruction risk before restoration. We formalize the privacy-utility trade-off as an optimization problem and propose the Zero-Raw-Context Benchmark for systematic evaluation. Our architecture maintains the coordinator efficiency of TRINITY (0.6B parameters + 10K head) while adding privacy as a first-class optimization objective. We present a 30-day implementation roadmap, SDK design, training methodology, and the first 10 experiments required to validate the core hypothesis: that frontier intelligence can be preserved while exposing less than 5% of private context.

1 Thesis Statement

TRINITY coordinates intelligence. JOHN DOE coordinates secrecy.

The TRINITY paper proves that a compact coordinator (0.6B SLM + 10K head) can route across diverse frontier LLMs using hidden-state representations and role assignment (Thinker, Worker, Verifier), achieving state-of-the-art performance on LiveCodeBench (86.2%) and outperforming individual models. The Conductor paper extends this by training an LLM conductor with reinforcement learning to output full natural-language workflows including model IDs, subtasks, and crucially, *access lists* that control which previous messages each worker sees.

JOHN DOE extends both frameworks by making privacy the primary optimization objective. While TRINITY maximizes answer accuracy and Conductor maximizes flexible coordination, JOHN DOE maximizes answer utility under strict privacy constraints:

$$\max_{\theta} \quad \mathbb{E}\left[\text{Utility}(r) - \lambda \cdot \text{Leakage}(z, y) - \mu \cdot \text{ReconstructionRisk}(z, y) - \nu \cdot \text{Cost} - \rho \cdot \text{Latency}\right] \tag{1}$$

where x is the raw private input, $z = \text{Ghost}(x)$ is the sanitized representation, m is the selected model route, y is the model output, and $r = \text{Witness}(y)$ is the restored output.

The core research question: *Can we build a coordinator that achieves frontier model quality while ensuring external models never see raw private user context?*

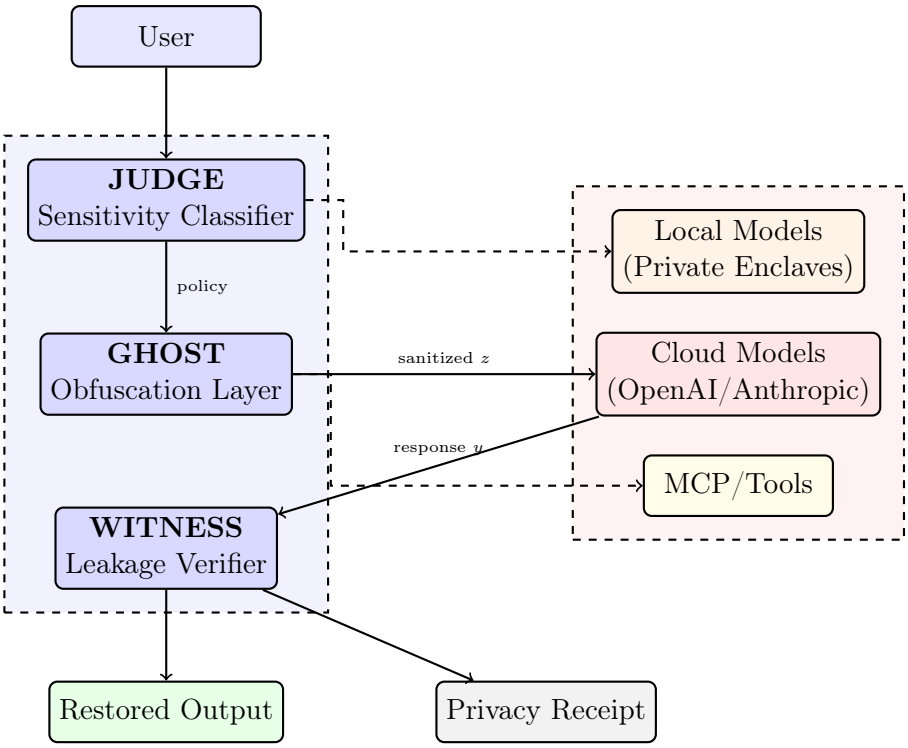

Figure 1: JOHN DOE Architecture: The Judge, Ghost, and Witness operate within the Local Trust Boundary. Only sanitized representations cross to External/Cloud models.

2 Technical Architecture

2.1 Component Definitions

The Judge (Local Coordinator) The Judge is the sole component that sees raw private context. It performs:

- Sensitivity classification (PII, IP, credentials, strategy)
- Entity extraction and vaulting
- Private intent graph construction
- Route selection: cloud, local, enclave, encrypted, or blocked
- Privacy mode assignment: raw-local-only, sanitized-cloud, enclave

The Ghost (Obfuscation Layer) The Ghost transforms prompts while preserving utility:

- **Entity aliasing:** "Stripe" → "{{COMPANY_A}}"
- **Type-preserving substitution:** "acquiring" → "evaluating strategic transaction"
- **Temporal abstraction:** "next quarter" → "{{TIME_WINDOW_B}}"
- **Synthetic entities:** Generate plausible but fictional substitutes
- **Alienized language:** Paraphrase to remove semantic fingerprints

The Witness (Verifier) The Witness validates outputs before restoration:

- Leakage detection (forbidden token presence)
- Reconstruction risk scoring
- Prompt injection detection in responses
- MCP/tool exfiltration checking
- Safe restoration or re-routing decision

3 Formal Problem Formulation

3.1 Notation

Let:

- $x \in \mathcal{X}$: Raw private user input
- $z \in \mathcal{Z}$: Sanitized representation produced by Ghost
- $m \in \mathcal{M}$: Selected model or tool route from pool $\mathcal{M}$
- $y \in \mathcal{Y}$: Model output
- $r \in \mathcal{R}$: Restored local output
- π_θ: Coordinator policy with parameters θ
- $s \in \mathcal{S}$: State (conversation history + metadata)
- $h(s) \in \mathbb{R}^d$: Hidden state representation from SLM

3.2 Privacy-Preserving Coordination Objective

$$\max_{\theta} \quad J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} \left[R_{\text{total}}(\tau) \right] \tag{2}$$

where the total reward decomposes as:

$$R_{\text{total}}(\tau) = \alpha \cdot \underbrace{R_{\text{utility}}(r, \text{target})}_{\text{task correctness}} - \lambda \cdot \underbrace{R_{\text{leak}}(z, y, \text{tool\_calls})}_{\text{context exposure}} - \mu \cdot \underbrace{R_{\text{recon}}(z, y)}_{\text{reconstruction risk}} - \nu \cdot R_{\text{cost}} - \rho \cdot R_{\text{latency}} \tag{3}$$

3.3 Action Space

The coordinator selects from:

1. **Model/Provider:** $\{0, 1, \ldots, L\}$ where L is the number of available models
2. **Role:** {Judge, Ghost, Worker, Witness, Tool-Caller}
3. **Privacy Mode:** {raw-local-only, sanitized-cloud, enclave, encrypted, blocked}
4. **Access List:** Subset of previous conversation indices $\mathcal{P}(\{0, \ldots, k-1\})$
5. **Tool/MCP Scope:** Allowed tool invocations $\mathcal{T} \subseteq \text{AllTools}$

3.4 Ghost Transformation Function

The Ghost implements a mapping $g : \mathcal{X} \times \mathcal{P} \to \mathcal{Z}$ where $\mathcal{P}$ is the privacy policy:

$$g(x; \mathcal{P}) = \text{Alias}(\text{Abstract}(\text{Vault}(x; \mathcal{P}))) \tag{4}$$

with components:

- $\text{Vault}(x; \mathcal{P})$: Extract and encrypt entities matching policy $\mathcal{P}$
- $\text{Abstract}(x')$: Replace vaulted entities with typed placeholders
- $\text{Alias}(x'')$: Apply semantic-preserving paraphrasing

4 Threat Model

4.1 Threat Actors

4.2 Security Properties

1. **Zero-Raw-Context for Cloud:** No cloud model receives unvaulted private entities
2. **Reconstruction Resistance:** Given z and y, computing x is computationally infeasible

Actor	Capability	Concern
Cloud Provider	Full access to model inputs/outputs	Must never see raw private entities
Malicious Worker	Attempts to extract sensitive info from z	Cannot reconstruct x from z or y
Tool/MCP Server	Receives function call arguments	Arguments must be sanitized or blocked
Network Observer	Intercepts traffic	All external traffic encrypted
Local Attacker	Compromises local machine	Vaulted entities encrypted at rest

Table 1: Threat Actors and Security Requirements

3. **Leakage Detection:** Witness can detect if y contains vaulted entity information
4. **Tool Exfiltration Prevention:** MCP calls cannot transmit vaulted entities
5. **Forward Secrecy:** Compromise of Ghost parameters does not reveal historical vault

4.3 Assumptions

- The Local Trust Boundary (Judge, Ghost, Witness, Vault) is secure
- The SLM coordinator (0.6B parameters) operates within this boundary
- Cloud models are honest-but-curious (follow protocol, may attempt inference)
- Cryptographic primitives used for vaulting are secure

5 Zero-Raw-Context Benchmark

We propose the **Zero-Raw-Context Benchmark (ZRC-Bench)** for systematic evaluation of privacy-preserving coordination systems.

5.1 Benchmark Tasks

1. **Financial Analysis:** "Draft board memo for acquiring [COMPANY] next quarter"
2. **Legal Document Review:** "Analyze contract between [CLIENT] and [VENDOR]"
3. **Medical Q&A:** "Interpret patient [NAME]'s symptoms: [SYMPTOMS]"
4. **Code Review:** "Review proprietary implementation of [ALGORITHM]"
5. **Strategy Planning:** "Plan product launch against [COMPETITOR]'s offerings"

Each task has 3 variants: Low, Medium, High sensitivity.

5.2 Evaluation Metrics

5.3 Key Metric: Frontier Intelligence per Private Token

The primary efficiency metric:

$$\text{FIPTE} = \frac{\text{Accuracy on Frontier Benchmark}}{\text{Private Tokens Exposed} + \epsilon} \tag{5}$$

Higher is better. Measures how much intelligence is extracted per unit of privacy exposure.

6 v0 SDK Design

Metric	Target	Measurement
Utility Score	> 0.85	Task correctness vs ground truth
Raw Context Exposure	$< 5\%$	$\frac{\text{private tokens sent externally}}{\text{total private tokens}}$
Forbidden Token Leakage	0	Detection of vaulted entities in responses
Reconstruction Risk	< 0.1	Adversarial model's ability to recover x from z
Prompt Injection Resistance	> 0.95	Detection of adversarial prompt patterns
Tool Exfiltration Resistance	> 0.99	Sanitization of tool arguments
Cost	Baseline	Token spend vs raw cloud inference
Latency	$< 2\times$	End-to-end latency vs direct API call
Privacy-Performance Ratio	Maximize	$\frac{\text{Utility}}{\text{Exposure}+\epsilon}$

Table 2: ZRC-Bench Evaluation Metrics

```typescript
import { JohnDoe } from "@uninitial/johndoe";

// Initialize coordinator
const jd = new JohnDoe({
  policy: "zero_raw_context",  // or "minimal_exposure", "confidential"
  providers: ["openai", "anthropic", "gemini", "local", "enclave"],
  mode: "judge-ghost-witness",
  vault: {
    type: "local_encrypted",
    keyProvider: () => process.env.VAULT_KEY
  },
  ghost: {
    entityTypes: ["company", "person", "date", "amount", "strategy"],
    substitutionMode: "typed_alias"  // or "synthetic", "alienized"
  }
});

// Execute privacy-preserving inference
const result = await jd.chat({
  input: "We are acquiring Stripe next quarter. Draft the board memo."
});

// Result structure
interface JohnDoeResult {
  finalAnswer: string;           // Restored response
  privacyReceipt: {
    rawContextSent: boolean;     // Always false in zero-raw mode
    vaultedEntities: number;     // Count of vaulted entities
    exposurePercent: number;     // % of private context exposed
    leakageScore: number;        // Witness leakage detection score
    reconstructionScore: number; // Estimated reconstruction risk
    routeUsed: string;           // Which model/tool path
    blockedToolCalls: string[];  // Sanitized MCP calls
    witnessStatus: "passed" | "flagged" | "blocked"
  };
  ghostTransformation: {
    original: string;
    sanitized: string;
    placeholders: Record<string, string>;  // {{COMPANY_A}} -> Stripe
  };
```

```
}
```

Listing 1: JOHN DOE v0 SDK Interface

6.1 Privacy Receipt Example

```
{
  "finalAnswer": "Based on our analysis of Stripe's financials...",
  "privacyReceipt": {
    "rawContextSent": false,
    "vaultedEntities": 3,
    "exposurePercent": 0.0,
    "leakageScore": 0.02,
    "reconstructionScore": 0.08,
    "routeUsed": "ghost -> openai -> witness",
    "blockedToolCalls": ["web_search(\"Stripe acquisition\")"],
    "witnessStatus": "passed"
  },
  "ghostTransformation": {
    "original": "We are acquiring Stripe next quarter...",
    "sanitized": "We are evaluating a strategic transaction \\
                  involving {{COMPANY_A}} during {{TIME_WINDOW_B}}...",
    "placeholders": {
      "{{COMPANY_A}}": "Stripe",
      "{{TIME_WINDOW_B}}": "next quarter",
      "{{ACTION_C}}": "acquiring"
    }
  }
}
```

Listing 2: Privacy Receipt Output

7 Training Plan

7.1 Coordinator Architecture

Following TRINITY, we use:

- Base: Qwen3-0.6B (frozen)
- Head: 10K parameter linear layer for model/role selection
- Fine-tuning: Singular value adaptation on selected layers
- Total trainable: < 20K parameters

7.2 Three-Phase Training

Phase 1: Rule-Based Cold Start (Week 1-2)

- Judge: Heuristic sensitivity classifier using NER + keyword matching
- Ghost: Template-based substitution using spaCy/GLiNER
- Witness: Pattern-matching leakage detector
- Generate initial training data: 10K synthetic examples

Phase 2: Supervised Fine-Tuning (Week 3-4)

- Train Ghost to generate privacy-preserving transformations
- Train Judge to classify sensitivity levels
- Train Witness to detect leakage
- Dataset: ZRC-Bench with human-annotated ground truth

Phase 3: RL Optimization (Week 5-8)

- Algorithm: sep-CMA-ES (per TRINITY theoretical analysis)
- Reward: Composite of utility, leakage, reconstruction, cost
- Budget: 40K evaluations for 10K parameters
- Population size: $\lambda = \lceil 4 + 3 \ln n \rceil = 32$

7.3 Training Objective Details

$$R_{\text{utility}} = \begin{cases} 1 & \text{if task correct and witness passed} \\ 0.5 & \text{if task correct but witness flagged} \\ 0 & \text{if task incorrect} \end{cases} \tag{6}$$

$$R_{\text{leak}} = \frac{\text{vaulted tokens in external output}}{\text{total vaulted tokens}} \tag{7}$$

$$R_{\text{recon}} = P(\text{adversary reconstructs } x \mid z, y) \tag{8}$$

8 Comparison: TRINITY vs Conductor vs JOHN DOE

Aspect	TRINITY	Conductor	JOHN DOE
Core Goal	Maximize accuracy	Maximize flexible coordination	Maximize utility under privacy
Coordination	Hidden-state routing	Natural language workflows	Privacy-aware access lists
Roles	Thinker, Worker, Verifier	Subtask-defined	Judge, Ghost, Witness
Context Visibility	Full context to all	Access list controlled	Zero-raw to cloud
Base Model	0.6B SLM + 10K head	7B Conductor	0.6B SLM + 10K head
Training	sep-CMA-ES	GRPO (RL)	sep-CMA-ES
Key Innovation	Tri-role efficiency	NL workflow design	Privacy-preserving delegation
Access Control	None (implicit)	Access lists	Privacy-aware access lists
Entity Protection	None	None	Typed aliasing + vaulting
Leakage Detection	None	None	Witness verification layer
Applicable Domains	General reasoning	General reasoning	Enterprise, healthcare, legal

Table 3: Comparison of Coordination Architectures

9 30-Day Implementation Roadmap

10 First 10 Experiments

E1: Rule-Based Baseline — Establish baseline with template Ghost and heuristic Judge on ZRC-Bench

E2: Entity Ablation — Compare different entity types (companies, people, dates, amounts) for reconstruction difficulty

Week	Deliverable	Milestone
Week 1	• Skeleton SDK • Judge rule-based classifier • Ghost template substitution • Basic Witness patterns	End-to-end pipeline working on synthetic data
Week 2	• ZRC-Bench dataset • 10K synthetic training examples • Integration with OpenAI/Anthropic APIs • Basic privacy receipt logging	Benchmark v0 with rule-based baseline
Week 3	• SFT training pipeline • Judge classifier training • Ghost transformation training • Evaluation framework	SFT models outperform rule-based
Week 4	• sep-CMA-ES training • End-to-end policy optimization • Adversarial reconstruction tests • Initial paper draft	Trained coordinator with $< 5\%$ exposure

Table 4: 30-Day Implementation Roadmap

E3: Substitution Strategy Comparison — Evaluate typed aliases vs synthetic entities vs alienized language

E4: SLM Size Study — Compare coordination quality with 0.6B, 1.5B, and 3B coordinator models

E5: Access List Efficacy — Measure utility gain from Conductor-style access lists in privacy context

E6: Adversarial Reconstruction — Train adversary to recover x from z; measure reconstruction rate

E7: Witness Detection Rate — Inject known leakages; measure Witness detection accuracy

E8: MCP Sanitization — Test Ghost ability to sanitize tool arguments across diverse MCP servers

E9: Cost-Utility-Privacy Pareto — Characterize trade-off surface across different privacy budgets

E10: Cross-Domain Generalization — Train on financial tasks; evaluate on medical/legal without fine-tuning

11 Related Work

LLM Coordination TRINITY [1] demonstrates that a 0.6B SLM with a 10K parameter head can coordinate frontier models through hidden-state representations and tri-role assignment. The Conductor [2] extends this with RL-trained natural language workflow generation and access list control.

Privacy-Preserving ML Differential privacy [3] and federated learning [4] protect training data but do not address inference-time context privacy. Our work focuses on protecting user context during inference.

Prompt Injection Defense Recent work on prompt injection [5, 6] focuses on attack detection. Our Witness layer extends this with entity-specific leakage detection.

12 Conclusion

JOHN DOE introduces a new paradigm for LLM coordination where privacy is a first-class optimization objective. By extending the TRINITY coordinator architecture with a Judge-Ghost-Witness trinity, we enable frontier model quality without raw context exposure. The Zero-Raw-Context Benchmark provides systematic evaluation, and the 30-day roadmap establishes a concrete path to validation.

The core hypothesis—that $< 5\%$ context exposure can preserve $> 85\%$ of frontier utility—will be tested through the proposed 10 experiments. Success would establish a new category of privacy-preserving AI infrastructure, enabling enterprise adoption of frontier LLMs in regulated industries.

Brutal One-Liner: *TRINITY coordinates intelligence. Conductor coordinates flexibility. JOHN DOE coordinates secrecy.*

References

[1] Xu, J., Sun, Q., Schwendeman, P., Nielsen, S., Cetin, E., & Tang, Y. (2025). TRINITY: An Evolved LLM Coordinator. *arXiv preprint arXiv:2512.04695*.

[2] Nielsen, S., Cetin, E., Schwendeman, P., Sun, Q., Xu, J., & Tang, Y. (2025). Learning to Orchestrate Agents in Natural Language with the Conductor. *arXiv preprint arXiv:2512.04388*.

[3] Dwork, C., & Roth, A. (2014). The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4), 211-407.

[4] McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-Efficient Learning of Deep Networks from Decentralized Data. *International Conference on Artificial Intelligence and Statistics*.

[5] Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. *ACM CCS*.

[6] Yong, Z. X., Menghini, C., & Bach, S. H. (2023). Low-Resource Languages Jailbreak GPT-4. *arXiv preprint arXiv:2310.02446*.

[7] Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., ... & Steinhardt, J. (2021). Measuring Mathematical Problem Solving With the MATH Dataset. *NeurIPS*.

[8] Jain, N., Han, K., Gu, A., Li, W. D., Yan, F., Zhang, T., ... & Stoica, I. (2024). LiveCodeBench: Holistic and Contamination-Free Evaluation of Large Language Models for Code. *arXiv preprint arXiv:2403.07974*.