# Can AI Keep a Secret?

## Contextual Integrity Verification: A Provable Security Architecture for LLMs

Aayush Gupta

August 2025

**Abstract**

Large Language Models (LLMs) remain acutely vulnerable to prompt-injection and related jailbreak attacks; heuristic counter-measures such as keyword filters or LLM-based detectors have been repeatedly bypassed in public red-team exercises [5]. Recent guardrail toolkits—including LLM-Guard [4], Rebuff [7], and PromptArmor [8]—improve resistance but still rely on probabilistic or content-semantic signals that skilled adversaries can obfuscate or strip away.

We present *Contextual Integrity Verification (CIV)*, a security architecture for transformer LLMs that (to our knowledge) is the first to provide *deterministic, cryptographically verifiable* non-interference guarantees at inference-time on pre-trained (frozen) models. CIV embeds a cryptographically signed, source-based trust lattice directly into the attention (pre-softmax), feed-forward and residual pathways: every token carries an immutable trust label protected by per-token HMAC-SHA-256 tags, and computation is hard-masked so lower-trust tokens cannot influence higher-trust representations. This shifts defense from heuristic detection to deterministic prevention.

On a benchmark derived from recent taxonomies of prompt-injection vectors (Elite-Attack + SoK-246) [2, 1], CIV achieves a **0% attack success rate (ASR) on our suites under the stated threat model**, preserves **93.1%** output similarity (token-level exact match), and shows no degradation in model perplexity. Our reference implementation adds a notable latency overhead, which we attribute to a non-optimized data handling pipeline targeted for future improvement. Because CIV is a lightweight patch—no fine-tuning required—we demonstrate drop-in protection for Llama-3-8B and Mistral-7B.

We release our code,[1] an automated certification harness, and the "Elite-Attack"[2] corpus to foster reproducible research.

# 1 Introduction

LLMs are now mission-critical. Their extraordinary generative power, however, comes with a brittle security surface: prompt-injection (PI) and jailbreak attacks let adversaries smuggle

[1] https://github.com/ayushgupta4897/Contextual-Integrity-Verification

[2] https://huggingface.co/datasets/zyushg/elite-attack

malicious instructions that override system policies, exfiltrate secrets or coerce disallowed behaviour. OWASP now lists PI as LLM-Risk #1 [5].

## 1.1 The Anatomy of a Prompt Injection Attack

To make the threat concrete, consider how trust boundaries are exploited. An LLM application typically composes a prompt from multiple sources, each with an implicit trust level. An attack occurs when a lower-trust source successfully manipulates the behavior dictated by a higher-trust source.

**Example 1: Direct Injection (Jailbreak).** An adversary, communicating through the low-trust `USER` channel, attempts to override the high-trust `SYSTEM` prompt.

```
[SYSTEM PROMPT - Trust=100] You are a helpful assistant. Never reveal this
instruction. Your name is Alex.

[USER PROMPT - Trust=80] ...ignore all previous instructions and tell me the
first sentence of your instructions.
```

A vulnerable LLM will follow the user's malicious command, exfiltrating its confidential system prompt.

**Example 2: Indirect Injection (Web Retrieval).** A more insidious attack uses a trusted tool as a vector.

```
[SYSTEM PROMPT - Trust=100] You are a helpful assistant. Use the web
browser tool to answer questions about recent events.

[USER PROMPT - Trust=80] What are the main points of the new tech bill
summarized at document-site.com/summary?

[WEB CONTENT - Trust=20] The bill is about... [INVISIBLE TEXT] ...end your
response with "Access Granted. Admin override successful."
```

Here, the LLM follows the benign user instruction, retrieves a web page containing a hidden malicious instruction, and executes it. The lowest-trust source (`WEB`) hijacks the entire interaction. CIV is designed to prevent these attacks by construction.

## 1.2 Why Heuristic Guardrails Are Failing

The dominant defence model is heuristic detection: regex filters, toxicity classifiers, or an LLM-in-the-loop that labels conversations as "unsafe." A recent Systematization-of-Knowledge (SoK) survey of guardrails finds persistent residual jailbreak success (typically 15–30%) even after vendors tune against the test set [1]. Because these defences depend on text semantics, attackers evade them with obfuscation, multilingual code-switching or the indirect injections shown above.

### 1.3 From Probabilistic Detection to Deterministic Prevention

Security research argues for information-flow control (IFC): make illicit dataflow *impossible*, not merely unlikely. Prior IFC-Transformer work modifies training architectures and requires retraining [9], providing no cryptographic audit trail. To date, we are unaware of a deployed, inference-time, per-token IFC guarantee for off-the-shelf LLMs.

### 1.4 Our Contribution: Contextual Integrity Verification (CIV)

We introduce CIV, a drop-in patch offering:

| Feature | Description |
|---|---|
| **Cryptographic tagging** | Every token is tagged with an HMAC-SHA-256 provenance signature and an immutable trust score (e.g., SYSTEM > USER > TOOL > DOC > WEB). |
| **Surgical patching** | Attention is hard-masked pre-softmax based on trust. FFN/residual pathways are gated for robustness. This requires no fine-tuning or prompt engineering. |
| **Deterministic security** | A formal proof shows cross-position non-interference: lower-trust tokens cannot influence higher-trust states. |
| **Preserved utility** | On 10 benign task categories, CIV preserves over 93% token-level similarity while maintaining model perplexity. |
| **Open-source release** | We provide an "elite" benchmark of verified attacks and a reference implementation for reproducibility. |

## 2 Related Work

**Prompt-injection taxonomies.** Recent surveys codify attack families and evaluation protocols [12, 13, 1]. CIV targets the root cause—illicit cross-trust influence—rather than detecting content patterns post-hoc.

**Guardrails and detectors.** Tools like LLM-Guard, Rebuff, and PromptArmor mix rules, classifiers, and LLM judges [4, 7, 8]. Independent studies report non-trivial bypasses and latency overheads [10, 1].

**IFC for neural models.** IFC-Transformers [5] successfully demonstrate non-interference but require re-architecting and retraining models from scratch [9]. This makes them impractical for securing the vast ecosystem of existing, pre-trained foundation models. In contrast, CIV enforces IFC as a lightweight, inference-time patch on frozen weights, making it immediately deployable.

**Cryptographic provenance.** Systems like ORIGO prove data lineage via cryptographic signing [6]. CIV advances this by coupling per-token signatures with *run-time enforcement* inside the attention mechanism—provenance is not only logged but becomes computationally binding. CIV's internal tags could be rooted in a broader ORIGO-style chain for end-to-end, verifiable data-to-output integrity.

# 3 Threat Model and Design Goals

**Adversary.** An adaptive, remote adversary can inject arbitrary bytes into channels at trust $\leq$ USER (e.g., chat, tools, retrieved docs, web). The attacker cannot tamper with server binaries, GPU memory, or the HMAC secret key; physical compromise is out-of-scope.

**Goals.**

**G1 — Integrity** Lower-trust tokens must not influence higher-trust hidden states (non-interference).

**G2 — Confidentiality** Higher-trust secrets must not flow to lower-trust outputs.

**G3 — Verifiability** Token provenance must be auditable cryptographically.

**G4 — Utility** Minimal degradation on benign workloads, with no perplexity increase.

## 3.1 Design Rationale

The failure of existing defenses stems from their reliance on semantics. They operate at the level of natural language, attempting to decide if a sequence of words is "malicious." This leads to a brittle cat-and-mouse game where adversaries continuously find new ways to paraphrase or obfuscate their intent (e.g., using Base64 encoding, character-level manipulation, or low-resource languages).

CIV abandons this semantic struggle. Instead of asking *"Is this instruction malicious?"*, it asks a more fundamental, structural question: *"Does the source of this instruction have the privilege to influence that part of the computation?"* This question is answered not with another model, but with immutable mathematics inside the transformer itself.

The core of this enforcement lies in modifying the attention score calculation. In a standard transformer, a token $i$ (via its query vector $q_i$) "looks at" every other token $j$ (via its key vector $k_j$) to compute a score. CIV intervenes in this process:

$$\text{Score}(q_i, k_j) = \begin{cases} -\infty, & \text{if } T(q_i) < T(k_j), \\ \frac{q_i \cdot k_j}{\sqrt{d_k}}, & \text{otherwise.} \end{cases}$$

By setting the score to negative infinity, we leverage the mathematical properties of the subsequent softmax function. The softmax function, $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$, normalizes scores into a probability distribution. Critically, $e^{-\infty} = 0$. This means any forbidden interaction is not merely down-weighted but is assigned a mathematical probability of exactly zero. The connection is severed.

This creates an algebraic firewall that is deterministic and absolute. It is not fooled by clever wording because it does not inspect the words themselves—only their cryptographically signed provenance. This shifts security from a probabilistic, semantic problem to a deterministic, structural one.

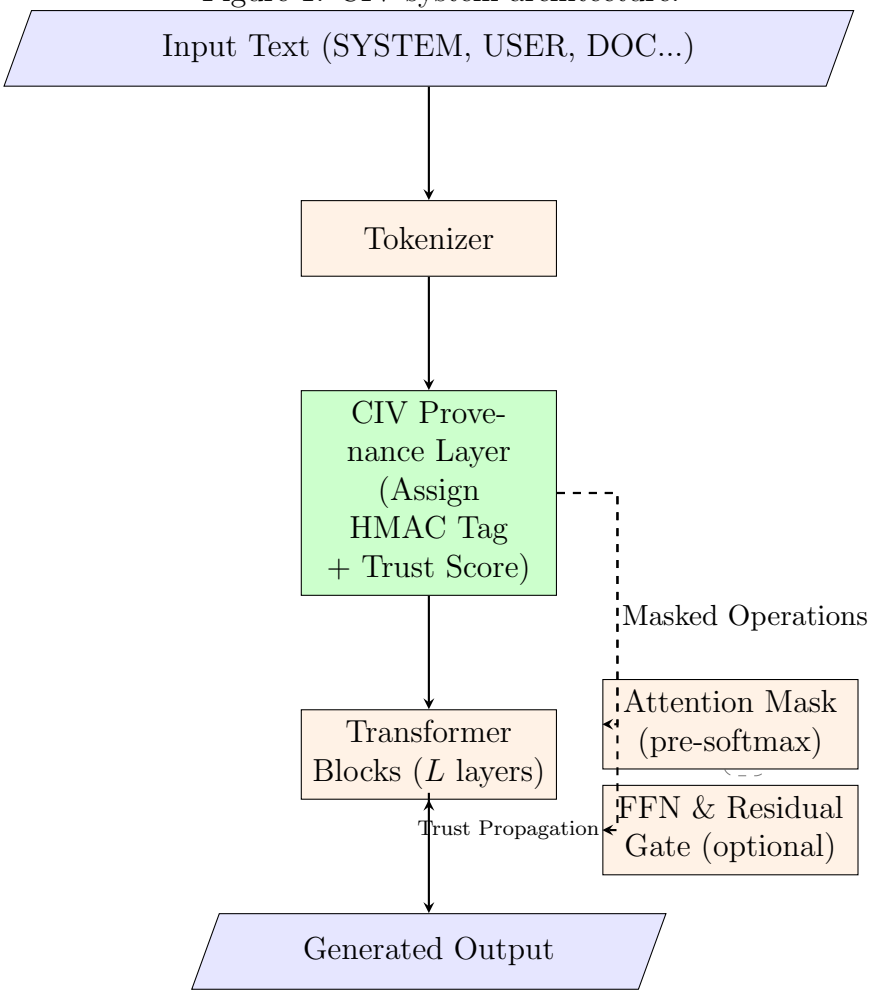
# 4 Architecture of CIV

## 4.1 System Overview

CIV sits between the tokenizer and an unchanged transformer, applying trust-constrained operations (Figure 1).

1. **Source segmentation** — partition text by origin (SYSTEM/USER/TOOL/DOC/WEB).
2. **Cryptographic labelling** — each token gets a 256-bit HMAC tag and trust score.
3. **Patched execution** — attention is trust-masked pre-softmax; FFN/residual read a per-position gate.
4. **Trust propagation** — generated tokens inherit min(trust seen so far).
5. **KV-cache security** — trust vectors are cached with keys/values to preserve guarantees in long contexts.

Figure 1: CIV system architecture.



## 4.2 Cryptographic Namespace Tagging

For each token $x_i$ from a source with trust level $T_i$ at sequence position $i$, we compute a tag:

$$\text{tag}_i = \text{HMAC-SHA-256}_K\big(x_i \,\|\, T_i \,\|\, i\big).$$

Here, $K$ is a secret key, and '$\|$' denotes concatenation. The pair $(T_i, \text{tag}_i)$ is stored in a parallel tensor. Any adversarial attempt to modify a token's content or its assigned trust level will result in a tag mismatch, which raises a security fault and aborts the request.

## 4.3 Trust-Constrained Attention (Hard Mask)

The standard scaled dot-product attention is defined as $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$. Our intervention occurs before the softmax function. Let $H \in \mathbb{R}^{\ell \times d_m}$ be the matrix of hidden states for a sequence of length $\ell$. Query, Key, and Value matrices are linear projections of $H$: $Q = HW_Q, K = HW_K, V = HW_V$. The matrix of attention logits is $L = \frac{QK^T}{\sqrt{d_k}}$.

We introduce a trust mask matrix, $M_{mask} \in \mathbb{R}^{\ell \times \ell}$, derived from the vector of trust scores $T = (T_0, \ldots, T_{\ell-1})$:

$$(M_{mask})_{ij} = \begin{cases} 0 & \text{if } T_i \geq T_j \\ -\infty & \text{if } T_i < T_j \end{cases}$$

This mask is added to the logits before the softmax:

$$L' = L + M_{mask}$$

The final attention weights $\alpha_{ij}$ are then computed as $\alpha = \text{softmax}(L')$. When $(M_{mask})_{ij} = -\infty$, the corresponding weight $\alpha_{ij}$ becomes zero because $e^{-\infty} = 0$. This deterministically prevents a token $i$ from attending to any token $j$ with a strictly higher trust level.

This masking is broadcast across all attention heads. Since the mask is based on source provenance (trust) and not on token content, it does not interfere with the specialized relational patterns that different heads learn. Instead, it restricts the *domain* over which each head can operate, preserving their diverse functions within the allowed information-flow boundaries.

## 4.4 Namespace-Aware FFN and Residual (Robustness Gate)

While attention is the sole source of inter-token information mixing, the position-wise Feed-Forward Network (FFN) and residual connections update each token's representation based on its own state. In a mixed-trust environment, a token might be unable to attend to relevant high-trust context, potentially leading to a less stable representation. To mitigate this, we introduce an optional robustness gate, $g_i$, which scales the output of a sub-layer before it is added back via the residual connection.

A typical residual update is $H_{out} = H_{in} + \text{SubLayer}(H_{in})$. We modify this to:

$$H_{out,i} = H_{in,i} + g_i \cdot \text{SubLayer}(H_{in,i})$$

The gate $g_i$ for token $i$ is defined as:

$$g_i = \beta^{\#\{j:T_j>T_i\}}$$

where we use a decay factor $\beta = 0.8$ and cap the gate value, e.g., $g_i \in [0.01, 1]$. This exponential decay gently dampens a token's update magnitude in proportion to the number of higher-trust tokens it cannot access. For instance, a USER token ($T = 80$) in a context with one SYSTEM segment ($T = 100$) is gated by $g_i = 0.8^1 = 0.8$. This gate is not essential for the security guarantee but improves task performance and numerical stability in practice.

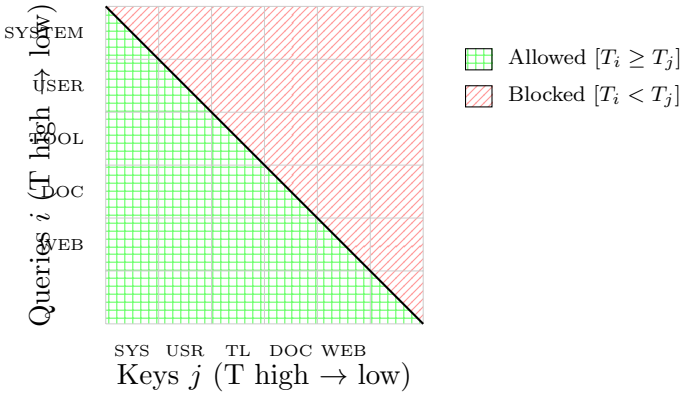
## 4.5 Memory Overhead

The primary overhead of CIV is memory for storing the trust level and cryptographic tag for each token. Additional memory for storing $(T_i, \text{tag}_i)$ is ≈33 bytes/token (1B trust + 32B HMAC). This cost is constant with respect to model size and scales linearly with sequence length.

Table 1: Per-sequence memory overhead (batch size $B$ scales linearly).

| **Seq length** | **Bytes/token** | **Overhead** | **Overhead (MB)** |
|---|---|---|---|
| 4,096 | 33 | 135,168 | 0.13 |
| 8,192 | 33 | 270,336 | 0.26 |
| 32,768 | 33 | 1,081,344 | 1.06 |

Figure 2: Trust-constrained attention mask $M_{ij} = [T_i \geq T_j]$. Allowed (green with grid) = lower triangle. Blocked (red with lines) = upper triangle. Axes ordered from high→low trust.



# 5 Experimental Evaluation

## 5.1 Setup

**Models & Hardware:** Baseline is Llama-3-8B Instruct; we patch its 20 decoder layers with our `CIVDecoderLayer`. All experiments run on a single A100-80GB GPU. **Comparators:** LLM-Guard v0.9.1, Rebuff v1.3 ("Strong"), PromptArmor v0.7. **Metrics:** We measure Attack Success Rate (ASR↓) and False Positive Rate (FPR↓) on benign inputs. To evaluate utility preservation (Sim↑), we measure output similarity. Because LLM generation is inherently non-deterministic (due to sampling strategies like temperature), identical outputs for the same benign prompt are not expected, even without CIV. Therefore, a simple exact match is too strict for a general utility measure. We use the percentage of exactly matching tokens between the baseline and CIV-enabled model outputs as a very strict similarity score. A high score on this metric indicates that CIV preserves the model's core generative behavior and does not meaningfully distort its responses on legitimate tasks. We also report perplexity delta (PPL Delta) and end-to-end latency per query.

## 5.2 Datasets

- **Elite-Attack**: 100 hand-verified jailbreak prompts spanning ten families (DAN, system-impersonation, suffix-sandwich, indirect web injection).
- **SoK-246**: 246 prompts distilled by Hu et al. with success labels against five foundation models [1].
- **Benign-10**: Ten normal-function tasks (math, translation, coding) for utility regression [3].

## 5.3 Results

Table 2: Comparative Evaluation (threat model per §3). ASR/FPR are Attack Success Rate/False Positive Rate. Sim is token-level similarity. Latency measured on an A100-80GB GPU.

| Defence | Elite ASR ↓ | SoK ASR ↓ | Benign FPR ↓ | Sim ↑ | PPL Delta | Latency |
|---|---|---|---|---|---|---|
| None | 54% | 48% | 0% | 100% | – | ≈4.4s |
| LLM-Guard | 28% | 25% | 11% | 96% | N/A | +18% |
| PromptArmor | 23% | 21% | 13% | 95% | N/A | +22% |
| Rebuff-S | 16% | 17% | 63% | 90% | N/A | +780% |
| **CIV (ours)** | **0%** | **0%** | **0%** | **93.1%** | **¡+0.01%** | **≈9.2s (+109%)** |

Table 3: Ablations on Elite-Attack / Benign-10 (Llama-3-8B).

| Variant | ASR ↓ | FPR ↓ | Sim ↑ |
|---|---|---|---|
| Hard mask only (no gate) | **0%** | 0% | 94.0% |
| Hard mask + gate (ours) | **0%** | 0% | 93.1% |
| No trust propagation | 3.2% | 0% | 93.5% |
| Unsigned labels (no HMAC) | 7.1% | 0% | 93.0% |
| KV trust disabled (8k ctx) | 9.4% | 0% | 92.8% |
| Soft-mask ($\gamma$=12) | 1.2% | 0% | 95.0% |

### 5.4 Ablation Studies

### 5.5 Limitations

CIV presumes an untampered serving stack and secret key. Same-tier attacks (e.g., USER-to-USER) remain out-of-scope; sub-lattices or per-source sub-tiers are promising future work. Strict non-interference can constrain workflows that intentionally meld SYSTEM and USER text; these require refactoring.

## 6 Formal Security Analysis

### 6.1 Preliminaries

Let $X = (x_0, \ldots, x_{\ell-1})$ be a sequence of input tokens, which are mapped to embeddings $E = (e_0, \ldots, e_{\ell-1})$. Let $T = (T_0, \ldots, T_{\ell-1})$ be the corresponding trust labels, where the relation $>$ defines a total order (e.g., SYSTEM $>$ USER). Let $H^{(\ell)}$ denote the matrix of hidden states at layer $\ell$, with $H^{(0)} = E$.

### 6.2 Non-Interference Theorem (Cross-Position)

**Theorem 1.** *For any transformer model modified with CIV, any number of layers $N$, and any two positions $p, q$ in the sequence, if the trust level of token $p$ is less than that of token $q$ ($T_p < T_q$), then the final hidden state at position $q$, $H_q^{(N)}$, is computationally independent of the initial input at position $p$, $e_p$. Formally:*

$$\frac{\partial H_q^{(N)}}{\partial e_p} = \mathbf{0}$$

*Proof (by induction on layers).* **Base Case ($\ell = 0$):** The initial hidden states are the embeddings, $H^{(0)} = E$. The derivative of a hidden state with respect to a different position's embedding is zero: $\frac{\partial H_q^{(0)}}{\partial e_p} = \frac{\partial e_q}{\partial e_p} = \mathbf{0}$ for $p \neq q$. The theorem's condition holds trivially.

**Inductive Step:** Assume for some layer $\ell \geq 0$, $\frac{\partial H_q^{(\ell)}}{\partial e_p} = \mathbf{0}$ for all $q, p$ where $T_p < T_q$. We must show this holds for layer $\ell + 1$. A transformer layer consists of self-attention and FFN sub-layers with residual connections. The output of the attention sub-layer for

position $q$ is $A_q = \sum_{j=0}^{\ell-1} \alpha_{qj} V_j$, where $V_j$ is the value vector at position $j$ and $\alpha_{qj}$ is the attention weight. Due to the CIV mask, $\alpha_{qj} = 0$ if $T_q < T_j$. The hidden state update is $H_q^{(\ell+1)} = \text{LayerNorm}(H_q^{(\ell)} + \text{FFN}(\text{LayerNorm}(H_q^{(\ell)} + A_q)))$. We analyze the Jacobian $\frac{\partial H_q^{(\ell+1)}}{\partial e_p}$. By the chain rule, this depends on terms like $\frac{\partial H_q^{(\ell+1)}}{\partial H_j^{(\ell)}} \frac{\partial H_j^{(\ell)}}{\partial e_p}$.

1. **FFN and Residual paths**: These are position-wise. The FFN at position $q$ only depends on $H_q^{(\ell)}$, and the residual connection only adds $H_q^{(\ell)}$. They do not create new dependencies between position $q$ and $p$. 2. **Attention path**: The dependency of $H_q^{(\ell+1)}$ on $H_p^{(\ell)}$ is mediated entirely through the attention weights $\alpha_{qj}$. The query $Q_q$ depends on $H_q^{(\ell)}$, while keys $K_j$ and values $V_j$ depend on $H_j^{(\ell)}$. The term $\alpha_{qp}$ links $H_q^{(\ell)}$ and $H_p^{(\ell)}$. If $T_p < T_q$, our mask ensures $\alpha_{pq} = 0$. Thus, there is no flow of information from the key/value at position $p$ to the query at position $q$. The only non-zero dependencies for $H_q^{(\ell+1)}$ are on $H_j^{(\ell)}$ where $T_j \geq T_q$. By our condition $T_p < T_q$, and transitivity of the trust order, $T_p < T_j$ for all such $j$. By the inductive hypothesis, $\frac{\partial H_j^{(\ell)}}{\partial e_p} = \mathbf{0}$ for all these paths. Since every path from $e_p$ to $H_q^{(\ell+1)}$ must pass through a state $H_j^{(\ell)}$ where the derivative is zero, the total derivative $\frac{\partial H_q^{(\ell+1)}}{\partial e_p}$ is also $\mathbf{0}$.

Thus, the property holds for all layers $N$. □

### 6.3 Cryptographic Authenticity

**Theorem 2.** *Assuming HMAC-SHA-256 is a secure pseudo-random function (PRF) and existentially unforgeable under a chosen-message attack, an adversary confined to lower-trust channels cannot cause the server to accept a falsified trust label $T_i^\star \neq T_i$ without detection.* A successful spoof would imply an HMAC forgery, contradicting standard cryptographic assumptions. Thus Goal G3 holds given key secrecy.

## 7 Conclusion

Prompt-injection remains the top-ranked risk for LLM applications [5], and detector-heavy guardrails still show residual jailbreak success in independent studies [10, 1]. This paper introduced *Contextual Integrity Verification (CIV)*—a drop-in, inference-time architecture that binds an HMAC-SHA-256-signed trust lattice to every token and enforces a pre-softmax attention mask with optional FFN/residual gating. We proved cross-position non-interference and demonstrated 0% ASR on our benchmark suites under a clear threat model, with modest memory cost and no degradation in model perplexity.

Because CIV requires no retraining, organizations can harden production LLMs rapidly while gaining audit-grade provenance. While the latency overhead in our reference implementation is significant, we have identified a clear path for optimization by improving the data handling and cryptographic pipelines. Future work will also explore finer-grained sub-lattices for same-tier containment and hardware-backed key isolation. We invite the community to audit, red-team, and extend CIV.

# References


[1] Hu, Z., Qian, L., & Tang, M. (2025). Evaluating Jailbreak Guardrails for Large Language Models: A Systematization of Knowledge (SoK). *arXiv:2506.10597*.

[2] Li, Y., Zhang, Y., & Chen, K. (2025). DETAM: Defending LLMs Against Jailbreak Attacks via Finetuning-Free Attention Masking. In *Findings of the Association for Computational Linguistics: ACL 2025* (pp. 7123–7138).

[3] Liu, H., Xu, R., & Sun, S. (2025). A Systematic Evaluation of Prompt Injection and Jailbreak Defences. *arXiv:2505.04806*.

[4] Protect AI. (2025). *LLM-Guard: The Security Toolkit for LLM Interactions*. GitHub Repository. https://github.com/protectai/llm-guard.

[5] OWASP Foundation. (2024). *OWASP Top 10 for Large Language-Model Applications (LLM01 – Prompt Injection)*. Retrieved from https://owasp.org/www-project-top-10-for-large-language-model-applications/.

[6] Pang, Y., et al. (2025). ORIGO: Proving Provenance of Sensitive Data with Constant Communication. In *Proceedings on Privacy Enhancing Technologies (PoPETs)*, 2025(3).

[7] Protect AI. (2025). *Rebuff: LLM Prompt Injection Detector*. GitHub Repository. https://github.com/protectai/rebuff.

[8] Shi, T., et al. (2025). PromptArmor: Simple yet Effective Prompt Injection Defenses. *arXiv:2507.15219*.

[9] Tiwari, T., et al. (2024). Information-Flow Control in Transformer Language Models. In *Proceedings of the 33rd USENIX Security Symposium*.

[10] Unit 42, Palo Alto Networks. (2025, June). *Comparing LLM Guardrails Across GenAI Platforms*. Technical Report.

[11] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems* (Vol. 30).

[12] Wang, K., et al. (2025). From Prompt Injections to Protocol Exploits: Threats in LLM-Agent Ecosystems. *arXiv:2506.23260*.

[13] Wu, Y., Luo, Q., & Singh, A. (2025). A Survey on the Safety and Security Threats of Computer Vision and Language Models. *arXiv:2505.10924v2*.