GRAIL: Verifiable LLM Inference for Decentralized Networks

Anonymous Authors

Abstract

Decentralized AI networks require verification mechanisms to ensure miners execute claimed computations honestly. We present GRAIL (Guaranteed Rollout Authenticity via Inference Ledger), a cryptographic protocol for verifiable LLM inference that combines beacon randomness, hidden state sketching, and challenge-response verification. GRAIL achieves 100% attack detection with minimal overhead (38ms commitment, 2ms verification) while outperforming existing methods by 58-1000×.

1 Introduction

Decentralized networks like Bittensor distribute LLM inference across miners who are incentivized to provide computational resources. However, miners can cheat by using smaller models, tampering with prompts, or fabricating outputs to reduce costs while claiming full rewards.

Existing verification approaches have critical limitations:

- **Zero-knowledge proofs** (zkLLM): Prohibitive 1000× overhead
- **Statistical methods** (TopLoc): Only probabilistic guarantees, poor scaling
- **Trusted execution**: Requires specialized hardware

GRAIL provides practical verification with cryptographic security guarantees. Key innovations:

1. **Beacon randomness**: Prevents pre-computation attacks
2. **Hidden state sketching**: Captures computational fingerprints efficiently
3. **Tolerance mechanisms**: Handles legitimate GPU computational variations
4. **Constant-size proofs**: 64-byte proofs regardless of sequence length

2 The GRAIL Protocol

2.1 Protocol Overview

GRAIL operates in three phases between a Prover (miner) and Verifier (validator):

Phase 1 - Commitment: Prover executes model on input, computes sketch values from hidden states, and creates cryptographic commitment.

Phase 2 - Challenge: Using fresh beacon randomness, verifier selects random token positions to verify.

Phase 3 - Verification: Verifier recomputes selected positions and checks against prover's commitment within tolerance bounds.

2.2 Hidden State Sketching

For each token position t, we compute a sketch value:

$$s_t = \langle h_t, r \rangle \bmod q \tag{1}$$

where h_t is the hidden state, r is a pseudorandom vector derived from beacon randomness, and $q = 2^{31} - 1$.

The sketch vector r is generated deterministically:

$$r = \text{PRF}(\text{“sketch”}, \text{beacon randomness}, d_{\text{model}}) \tag{2}$$

2.3 Challenge Generation

Challenge indices are selected using fresh beacon randomness:

$$\text{indices} = \text{Sample}_k(\text{PRF}(\text{“open”}, \text{tokens}, \text{beacon}_{R+1})) \tag{3}$$

Using the token sequence (not sketch values) ensures indices remain stable despite numerical variations.

2.4 Tolerance Mechanism

To accommodate GPU computational variations while maintaining security, verification succeeds if:

$$|\text{local} - \text{committed}|_q \leq \tau \tag{4}$$

where $\tau = 3$ provides optimal balance between hardware compatibility and security.

Algorithm 1 GRAIL Protocol

Commitment Phase:
$\text{beacon}_R \leftarrow \text{GetBeacon}()$
$r \leftarrow \text{PRF}(\text{beacon}_R)$
tokens $\leftarrow$ Model.Generate(prompt)
s_vals $\leftarrow [\langle h_t, r \rangle \bmod q \text{ for } t]$
signature $\leftarrow$ HMAC(s_vals)
Challenge Phase:
$\text{beacon}_{R+1} \leftarrow \text{GetBeacon}()$
indices $\leftarrow \text{Sample}_k(\text{PRF}(\text{tokens, beacon}_{R+1}))$
Verification Phase:
Verify HMAC signature
Recompute hidden states for challenged indices
Check each index within tolerance τ

3 Security Analysis

GRAIL provides strong security guarantees against adaptive adversaries:

Attack Detection: For k challenge indices with tolerance τ, adversary success probability is bounded by $\left(\frac{2\tau+1}{q}\right)^k \approx 2^{-232}$ for our parameters.

Beacon Security: Unpredictable randomness prevents pre-computation attacks.

Cryptographic Integrity: HMAC signatures ensure commitment tampering detection.

3.1 Hyperplane Non-Uniqueness Analysis

A theoretical concern with sketch-based verification is that multiple hidden states can produce identical sketch values, as the dot product maps high-dimensional spaces to scalars. We empirically validate GRAIL's security against this mathematical property through comprehensive experiments.

Key Findings:

1. **Single Constraint ($k = 1$):** Alternative hidden states exist with $\sim$0.01% collision rate, confirming the theoretical concern. However, finding specific sketch values remains computationally hard.

2. **Multi-Constraint Scaling:** Success rate drops exponentially with k:

   - $k = 1$: 1.40% success rate
   - $k = 2$: 0.10% success rate
   - $k \geq 4$: 0.00% success rate

   With $k = 16$, finding valid alternatives becomes computationally infeasible.

3. **Model Perturbation Robustness:** Even minimal weight perturbations (scale 0.1) destroy sketch preservation (27.78% rate) while maintaining token accuracy (100%), demonstrating that preserving both simultaneously is extremely difficult.

4. **Attack Infeasibility:** Gradient-based attacks achieved 0% success rate across all tested prompts. The computational complexity for $k = 16$ yields success probability $\approx 10^{-135.8}$.

These results validate that while mathematical non-uniqueness exists, GRAIL's multi-position verification ($k = 16$) creates an exponentially constrained optimization problem that is computationally infeasible to solve, ensuring practical security.

4 Experimental Results

We evaluate GRAIL across models from 40M to 774M parameters, comparing against TopLoc (current state-of-the-art).

4.1 Performance Comparison

Metric	GRAIL	TopLoc	Improvement
Commitment Time	0.038s	2.215s	58.2×
Verification Time	0.002s	2.206s	1103×
Proof Size	64 bytes	238 bytes	3.7×
Scalability	O(1)	O($k \cdot n$)	Constant

Table 1: Performance comparison with TopLoc

4.2 Security Validation

4.3 Hardware Robustness

GRAIL maintains high success rates across different hardware configurations:

Attack Type	Tests	Detection Rate
Model Substitution	150	100%
Prompt Tampering	100	100%
Token Manipulation	200	100%
Signature Forgery	100	100%
Replay Attacks	100	100%

Table 2: Attack detection results

Hardware	Precision	Success Rate	Tolerance
NVIDIA H200	fp32	100%	$\tau = 2$
NVIDIA A100	fp32	99.7%	$\tau = 3$
CPU (x86_64)	fp32	98.1%	$\tau = 3$
Mixed Precision	bf16	95%	$\tau = 4$

Table 3: Cross-platform compatibility

5 Deployment Considerations

Bittensor Integration: GRAIL integrates as a subnet with minimal protocol changes. Verification failures trigger graduated penalties from warning (10% incentive reduction) to stake slashing (50% loss).

Economic Security: Detection probability near 1.0 makes attacks economically irrational under any reasonable penalty structure.

Scalability: Constant proof size and sub-linear verification complexity enable deployment at scale without bandwidth constraints.

6 Limitations

- **Beacon Dependency:** Requires secure beacon infrastructure
- **Implementation Sensitivity:** Vulnerable to compiler/hardware variations
- **Model Evolution:** Cannot distinguish legitimate updates from substitution
- **Scale Limits:** Evaluation limited to 774M parameters; ultra-large models (175B+) require further analysis

7 Conclusion

GRAIL provides the first practical solution for verifiable LLM inference in decentralized networks. With 100% attack detection, minimal overhead, and strong scalability properties, GRAIL enables trustless AI computation essential for decentralized AI infrastructure.

The protocol's cryptographic security guarantees combined with practical efficiency make it suitable for production deployment in networks like Bittensor, advancing the development of incentive-aligned machine learning systems.