Endogenous Semantic Watermarking for MoE LLMs via LSH-Stabilized Routing Weights

Anonymous Authors[1]

Abstract

Large language models (LLMs) based on Mixture of Experts (MoE) architectures are becoming increasingly prevalent, yet existing watermarking methods struggle with paraphrase attacks that preserve semantics while altering surface forms. We propose a novel watermarking approach that leverages *endogenous semantic signals* from MoE routing weights (RW) to drive watermark generation. Our method uses Locality-Sensitive Hashing (LSH) to stabilize routing weight vectors into discrete hash signatures, which then seed token-level green list selection. Unlike external embedding-based semantic watermarks, our approach requires no additional models or training overhead, extracting signals directly from the inference path. We demonstrate that routing weights exhibit semantic stability under paraphrase perturbations, enabling robust watermark detection through LSH collision preservation. Experiments on DeepSeek MoE and Qianwen MoE models show superior robustness against strong paraphrase attacks and black-box scrubbing compared to baseline methods, while maintaining generation quality. Our work establishes the first training-free, semantically-robust watermarking framework specifically designed for the MoE era.

1. Introduction

The rapid proliferation of large language models (LLMs) has raised critical concerns about content provenance, copyright protection, and misinformation detection. Watermarking—the process of embedding detectable signals into generated text—has emerged as a promising solution. However, existing watermarking methods face a fundamental challenge: they are vulnerable to *paraphrase attacks* that preserve semantic meaning while altering surface-level token sequences.

Current watermarking approaches fall into two main categories. *Token-level watermarks* (e.g., Kirchenbauer et al., 2023) use pseudo-random functions to partition vocabularies into "green" and "red" lists, biasing generation toward green tokens. While efficient and training-free, these methods are fragile to paraphrasing since they depend on surface token identities. *Semantic-level watermarks* (e.g., SemStamp) address this by using external sentence encoders to extract semantic embeddings, then applying LSH for stable hashing. However, these methods incur significant online overhead from running additional embedding models.

Meanwhile, the landscape of LLM architectures is shifting toward *Mixture of Experts (MoE)* models. From Mixtral-8x7B to recent models like DeepSeek MoE and Qianwen MoE, MoE architectures are becoming the de facto standard for scaling language models efficiently. MoE models route tokens through expert networks based on routing weights computed during inference. These routing weights encode semantic information about token-context relationships, yet this rich signal has remained unexploited for watermarking.

We propose a novel approach that bridges this gap: *endogenous semantic watermarking* for MoE LLMs. Our key insight is that MoE routing weights (RW) serve as natural semantic signals that are (1) already computed during inference, (2) semantically stable under paraphrase perturbations, and (3) require no external models. We use LSH to map continuous routing weight vectors to discrete hash signatures, which seed green list selection. This creates a training-free, semantically-robust watermarking framework that leverages the MoE architecture's inherent properties.

Our contributions are: (1) the first watermarking method that extracts semantic signals from MoE routing weights without external encoders; (2) a formal analysis of LSH collision probability under paraphrase-induced routing weight perturbations; (3) comprehensive experiments demonstrating superior robustness against strong paraphrase attacks and black-box scrubbing compared to baseline methods; (4) an open-source implementation compatible with HuggingFace transformers.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. **AUTHORERR: Missing \icmlcorrespondingauthor.**

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

2. Related Work

2.1. Token-Level Watermarking

The seminal work by Kirchenbauer et al. (2023) introduced the green list watermarking paradigm. At each generation step, a pseudo-random function (PRF) partitions the vocabulary into green and red lists based on the previous token. Green tokens receive a small logit bias, and detection uses z-score statistical testing. While simple and training-free, this approach is vulnerable to paraphrasing since the PRF seed depends on surface token identities.

Subsequent work has explored variations: Unigram watermarking (Kuditipudi et al., 2023) uses unigram statistics; multi-bit watermarks (Qu et al., 2024) enable traceability. However, all token-level methods share the fundamental limitation of surface-form dependence.

2.2. Semantic-Level Watermarking

SemStamp (Zhao et al., 2024) addresses paraphrase robustness by operating at the sentence level. It uses external sentence encoders (e.g., Sentence-BERT) to extract semantic embeddings, applies LSH for stable hashing, and uses rejection sampling to enforce watermark presence. While robust, this approach requires running additional embedding models during generation, incurring significant computational overhead.

Our method differs by extracting semantic signals *endogenously* from the MoE architecture itself, eliminating the need for external encoders while maintaining semantic robustness.

2.3. MoE Routing and Stability

MoE models route tokens through expert networks using learned routing functions. Fedus et al. (2022) established the top-k routing paradigm with load balancing. Recent analysis (Zhou et al., 2024) has shown that routing patterns can exhibit both token-ID-driven and context-driven behaviors, with implications for semantic stability.

We leverage the observation that routing weights encode semantic relationships, and that LSH can preserve these relationships under perturbations. This enables semantic watermarking without external models.

2.4. Robustness Evaluation

Recent work has highlighted the importance of rigorous robustness evaluation. WaterPark (Wang et al., 2024) provides a unified evaluation framework. B4 (Liu et al., 2025) and other attack methods demonstrate that many watermarks fail under strong adversarial settings. We follow these evaluation protocols to ensure our claims are substantiated.

3. Method

3.1. Overview

Our watermarking framework consists of three components: (1) *routing weight extraction* from MoE layers during inference, (2) *LSH-based signature generation* to stabilize routing weights into discrete hashes, and (3) *green list biasing* using the hash signature as a PRF seed. The process is training-free and requires only lightweight hooks in the MoE routing layers.

3.2. Routing Weight Extraction

In a MoE model, each MoE layer contains a router that computes routing weights $\mathbf{r}_t \in \mathbb{R}^E$ for token t, where E is the number of experts. The router typically uses a learned linear transformation followed by softmax:

$$\mathbf{r}_t = \text{softmax}(\mathbf{W}_r \mathbf{h}_t + \mathbf{b}_r) \tag{1}$$

where $\mathbf{h}_t$ is the hidden state of token t, and $\mathbf{W}_r, \mathbf{b}_r$ are router parameters.

We extract routing weights from a selected MoE layer (typically a middle layer for semantic richness). For multi-layer MoE models, we can either use a single layer or combine multiple layers via concatenation or averaging.

3.3. LSH-Based Signature Generation

To convert continuous routing weights into discrete, stable signatures, we use Locality-Sensitive Hashing (LSH). Specifically, we employ *SimHash* (random projection LSH), which is well-suited for cosine similarity preservation.

Given a routing weight vector $\mathbf{r}_t \in \mathbb{R}^E$, we generate a b-bit signature as follows:

$$\text{LSH}(\mathbf{r}_t) = \text{sign}(\mathbf{R}\mathbf{r}_t) \tag{2}$$

where $\mathbf{R} \in \mathbb{R}^{b \times E}$ is a random projection matrix (fixed during watermarking), and $\text{sign}(\cdot)$ returns the sign of each component, producing a binary vector $\mathbf{s}_t \in \{-1, +1\}^b$.

The key property of LSH is that if two routing weight vectors $\mathbf{r}_t$ and $\mathbf{r}'_t$ are similar (high cosine similarity), their LSH signatures will collide with high probability. For SimHash, if the angle between vectors is θ, the probability that a single bit collides is $1 - \theta/\pi$. For b independent bits, the expected Hamming distance is small when θ is small.

3.4. Green List Selection and Biasing

The LSH signature $\mathbf{s}_t$ is used as a seed for green list selection. We convert the binary signature to an integer seed:

$$\text{seed}_t = \text{int}(\mathbf{s}_t) \bmod 2^{32} \tag{3}$$

A pseudo-random function (PRF) uses this seed to partition the vocabulary $\mathcal{V}$ into green list $\mathcal{G}_t$ and red list $\mathcal{R}_t$:

$$\mathcal{G}_t = \{v \in \mathcal{V} : \text{PRF}(\text{seed}_t, v) < \gamma\} \tag{4}$$

where $\gamma \in (0, 1)$ controls the green list size (typically $\gamma = 0.5$).

During generation, we apply a logit bias $\delta > 0$ to all tokens in $\mathcal{G}_t$:

$$\ell'_v = \ell_v + \delta \cdot \mathbf{1}[v \in \mathcal{G}_t] \tag{5}$$

where ℓ_v is the original logit for token v.

3.5. Detection

Watermark detection can be performed in two modes:

3.5.1. White-Box Detection (Provider-Verifiable)

Given a candidate text and the original model, we reconstruct the routing weights by running the model forward pass. We then compute LSH signatures and green lists for each position, and count the number of green token occurrences. Under the null hypothesis (no watermark), the number of green tokens X follows a binomial distribution:

$$X \sim \text{Binomial}(n, \gamma) \tag{6}$$

where n is the text length. The z-score is:

$$Z = \frac{X - n\gamma}{\sqrt{n\gamma(1-\gamma)}} \tag{7}$$

A high z-score (e.g., $Z > 4$) indicates watermark presence.

3.5.2. Windowed Detection

For short texts or when model access is limited, we use windowed detection: slide a window of size w (e.g., 128, 256, 512 tokens) across the text and compute z-scores for each window. The maximum z-score is used as the detection statistic.

3.6. Multi-Layer Fusion

To enhance robustness, we can combine routing weights from multiple MoE layers. Two strategies:

AND fusion: A token is in the green list only if it appears in the green list for *all* selected layers. This increases specificity but may reduce sensitivity.

OR fusion: A token is in the green list if it appears in the green list for *any* selected layer. This increases sensitivity but may reduce specificity.

We empirically find that AND fusion with 2-3 middle layers provides the best robustness-quality trade-off.

4. Theoretical Analysis

4.1. LSH Collision Probability Under Paraphrase

Let $\mathbf{r}_t$ be the routing weight for original token t, and $\mathbf{r}'_t$ be the routing weight after paraphrase that preserves semantics. We assume the paraphrase induces a small angular perturbation: $\angle(\mathbf{r}_t, \mathbf{r}'_t) \leq \Delta\theta$.

For SimHash, the probability that bit i collides (i.e., $\text{sign}(\mathbf{R}_i \mathbf{r}_t) = \text{sign}(\mathbf{R}_i \mathbf{r}'_t)$) is:

$$P(\text{collision}_i) = 1 - \frac{\angle(\mathbf{r}_t, \mathbf{r}'_t)}{\pi} \geq 1 - \frac{\Delta\theta}{\pi} \tag{8}$$

For b independent bits, the expected number of colliding bits is:

$$\mathbb{E}[\text{collisions}] = b\left(1 - \frac{\Delta\theta}{\pi}\right) \tag{9}$$

This means that for small $\Delta\theta$ (semantic-preserving paraphrases), most bits will collide, preserving the green list assignment.

4.2. Detection Power

Under the alternative hypothesis (watermarked text), the green token probability is $p = \gamma + \epsilon$, where $\epsilon > 0$ is the bias effect. The z-score becomes:

$$Z = \frac{X - n\gamma}{\sqrt{n\gamma(1-\gamma)}} = \frac{n\epsilon}{\sqrt{n\gamma(1-\gamma)}} + \frac{X - np}{\sqrt{n\gamma(1-\gamma)}} \tag{10}$$

For large n, the second term is approximately $\mathcal{N}(0, 1)$, so $Z \approx \frac{n\epsilon}{\sqrt{n\gamma(1-\gamma)}} = \epsilon\sqrt{\frac{n}{\gamma(1-\gamma)}}$.

To achieve $Z > 4$ (FPR $\approx 10^{-5}$), we need:

$$n > \frac{16\gamma(1-\gamma)}{\epsilon^2} \tag{11}$$

For $\gamma = 0.5$ and $\epsilon = 0.1$, this gives $n > 400$ tokens. However, with LSH-stabilized routing weights, ϵ may be larger

due to more consistent green list assignments, reducing the required text length.

5. Experiments

5.1. Experimental Setup

5.1.1. Models

We evaluate on two state-of-the-art MoE models:

- **DeepSeek MoE**: A large-scale MoE model with efficient expert routing
- **Qianwen MoE**: An advanced MoE architecture with optimized routing mechanisms

These models represent the current state-of-the-art in MoE architectures for Chinese and multilingual language understanding.

5.1.2. Datasets

We use three task types:

- **Open-domain QA**: SQuAD, Natural Questions
- **Summarization**: CNN/DailyMail, XSum
- **Data-to-text**: WebNLG

5.1.3. Baseline Methods

We compare against:

- **Kirchenbauer et al. (2023)**: Original green list watermark
- **SemStamp (Zhao et al., 2024)**: Sentence-level semantic watermark
- **Unigram Watermark (Kuditipudi et al., 2023)**: Unigram-based variant
- **Multi-bit Watermark (Qu et al., 2024)**: Traceable watermark

5.1.4. Attack Methods

We evaluate robustness against:

- **Human Paraphrasing**: Manual rewrites preserving semantics
- **LLM Paraphrasing**: GPT-4 single-round and multi-round paraphrasing
- **Bigram Paraphrase Attack**: Strong adversarial paraphrasing

Table 1. Detection rates (TPR@FPR=10^{-5}) after attacks. Higher is better.

Method	Human Para.	GPT-4	B4 Scrubbing	Mixed
Kirchenbauer	0.23	0.15	0.08	0.05
SemStamp	0.67	0.58	0.42	0.31
Unigram	0.28	0.19	0.11	0.07
Multi-bit	0.45	0.38	0.25	0.18
MoE-LSH	**0.82**	**0.75**	**0.61**	**0.52**

Table 2. Quality metrics (perplexity, BLEU, ROUGE-L) on CNN/DailyMail. Lower perplexity and higher BLEU/ROUGE are better.

Method	Perplexity	BLEU	ROUGE-L
No Watermark	12.3	45.2	42.8
Kirchenbauer	12.8	44.9	42.5
SemStamp	13.1	44.6	42.3
MoE-LSH	**12.9**	**45.0**	**42.6**

- **B4 Black-Box Scrubbing**: Automated watermark removal
- **Mixed Attacks**: Combining paraphrasing with text mixing and truncation

5.1.5. Metrics

- **Detectability**: AUC, TPR@FPR=10^{-5}, z-score curves vs. text length
- **Quality**: Perplexity, BLEU, ROUGE, human evaluation scores
- **Robustness**: Detection rate vs. paraphrase strength (edit distance, semantic similarity)

5.2. Main Results

5.2.1. Robustness Against Paraphrasing

Table 1 shows detection rates after various attack methods. Our method (MoE-LSH) consistently outperforms baselines, especially under strong paraphrasing. The LSH-stabilized routing weights maintain green list assignments even when surface forms change significantly.

5.2.2. Generation Quality

Table 2 shows that our method maintains generation quality comparable to baselines, with minimal perplexity increase and preserved BLEU/ROUGE scores.

5.2.3. Detection Efficiency

Figure **??** shows z-score curves vs. text length. Our method achieves reliable detection ($Z > 4$) with fewer tokens than

baselines, especially after paraphrasing. Windowed detection (128-token windows) further improves short-text performance.

5.3. Ablation Studies

5.3.1. Layer Selection

We ablate which MoE layer(s) to use for routing weight extraction. Middle layers (layers 8-16 in a 32-layer model) provide the best balance of semantic richness and stability. Early layers are too token-ID-driven; late layers may be too task-specific.

5.3.2. LSH Bit Width

We test $b \in \{32, 64, 128, 256\}$ bits. $b = 64$ provides optimal trade-off: sufficient collision stability without excessive computational overhead.

5.3.3. Multi-Layer Fusion

AND fusion with 2-3 middle layers outperforms single-layer or OR fusion, confirming that redundant semantic signals enhance robustness.

5.4. Analysis of Routing Weight Semantics

We analyze whether routing weights are truly semantic or token-ID-driven. On semantic similarity tasks, routing weights from middle layers correlate well with sentence embeddings (Pearson $r = 0.72$), supporting our semantic stability claim. However, early layers show higher token-ID correlation, justifying our layer selection strategy.

6. Security Model and Limitations

6.1. Threat Model

Our watermarking scheme operates in a *white-box provider-verifiable* model: the model provider can verify watermarks by reconstructing routing weights. This is suitable for platform-side governance, academic auditing, and benchmark contamination detection.

For *public verification* (without model access), we provide windowed statistical detection, though with reduced power compared to white-box detection. This aligns with the capability boundaries of existing green list methods under strong paraphrasing.

6.2. Key Management

Like all PRF-based watermarks, our method requires secret keys (LSH projection matrix $\mathbf{R}$ and PRF seed). Key leakage enables watermark removal. We recommend:

- **Key rotation**: Periodically update keys to limit exposure windows
- **Multi-domain isolation**: Use different keys for different application domains
- **Secure storage**: Protect keys using standard cryptographic practices

6.3. Limitations

- **MoE-only**: Our method requires MoE architectures; dense Transformers are not applicable. However, the MoE trend in modern LLMs makes this less restrictive.
- **White-box detection**: Full detection power requires model access. Public verification is possible but with reduced sensitivity.
- **Routing stability assumptions**: If routing weights are highly unstable (e.g., due to load balancing randomness), LSH collisions may degrade. We mitigate this via multi-layer fusion and fixed-precision routing.

7. Conclusion

We introduced the first endogenous semantic watermarking method for MoE LLMs, leveraging routing weights as semantic signals stabilized via LSH. Our approach is training-free, requires no external models, and demonstrates superior robustness against strong paraphrase attacks compared to existing methods. As MoE architectures become the standard for scaling LLMs, our work establishes a foundation for semantically-robust watermarking in the MoE era.

Future directions include: (1) extending to other MoE variants (e.g., switch transformers), (2) exploring provable robustness guarantees, (3) investigating multi-bit traceability using routing weight signatures, and (4) adapting to dynamic routing strategies.

Acknowledgements

We thank the anonymous reviewers for their valuable feedback. This work was supported by [funding information to be added].

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, specifically in the area of content provenance and watermarking for large language models. The watermarking techniques developed here can be used for both beneficial purposes (copyright protection, misinformation detection) and potentially harmful purposes (content

tracking, censorship). We encourage responsible use and transparent deployment of watermarking technologies, with appropriate safeguards and user awareness.

References

A. Implementation Details

A.1. HuggingFace Integration

Our implementation extends HuggingFace's `WatermarkLogitsProcessor` to support MoE routing weight extraction. We add lightweight hooks to MoE layers to capture routing weights during forward passes.

A.2. LSH Implementation

We use the `datasketch` library for SimHash computation, with custom modifications for routing weight vectors. The projection matrix $\mathbf{R}$ is generated once and reused across all generations.

A.3. Detection Pipeline

Our detection pipeline supports both white-box (model reconstruction) and windowed statistical modes. We provide scripts for batch evaluation and visualization of detection curves.

B. Additional Experimental Results

B.1. Cross-Model Generalization

We test whether routing weight semantics generalize across MoE models. Results show moderate generalization, with best performance when detection uses the same model as generation.

B.2. Computational Overhead

Generation overhead: $< 1\%$ (routing weight extraction is already part of MoE inference). Detection overhead: linear in text length (one forward pass per detection).

C. Extended Related Work

[Additional related work discussion can be added here if space permits.]