RLSpoofer: A Sample-Efficient Black-Box Spoofing Attack for Stress-Testing LLM Watermarks

Hanbo Huang [1] **Xuan Gong** [1] **Yiran Zhang** [1] **Hao Zheng** [1] **Wenbin Dai** [1] **Jieren Kuang** [1] **Shiyu Liang** [1]

Abstract

Large language model (LLM) watermarking has emerged as a promising approach for detecting and attributing AI-generated text, yet its robustness to black-box spoofing remains insufficiently evaluated. Existing evaluation methods often demand extensive datasets and white-box access to algorithmic internals, limiting their practical applicability. In this paper, we study watermark resilience against spoofing from a distributional perspective. We first establish a *local capacity bottleneck*, which theoretically characterizes the probability mass that can be reallocated under KL-bounded local updates with semantic-fidelity constraints. Motivated by this, we propose RLSpoofer, a reinforcement learning-based black-box spoofing attack that requires only 100 human-watermarked paraphrase training pairs and zero access to the watermarking internals or detectors. Despite weak supervision, it empowers a 4B model to achieve a 62.0% spoof success rate with small semantic shift on PF-marked texts, far exceeding the 6% of baseline methods trained on up to 10,000 samples. Our findings expose the weaknesses in spoofing resistance of current LLM watermarking paradigms, providing a sample-efficient evaluation framework and underscoring the urgent need for more robust schemes.

1. Introduction

With the rapid advancement and increasing availability of large language models (LLMs), they are being widely applied across diverse applications to generate fluent and human-like content (Yang et al., 2025; Grattafiori et al., 2024). However, this widespread use raises major concerns about model misuse, including the generation of misinformation (Chen & Shu, 2023), copyright violations (Xu et al., 2025), data contamination (Shumailov et al., 2023), and academic dishonesty (Cotton et al., 2024). As a safeguard, text watermarking has become an important defense. By subtly embedding statistical signals into model outputs, watermarking enables reliable AI text detection and source tracking while maintaining the quality of the text (Kirchenbauer et al., 2023; Zhao et al., 2024).

Most existing watermarking schemes predominantly operate on a generate-and-detect paradigm, utilizing detection algorithms to identify hidden patterns that differentiate AI-generated text from human authorship (Lu et al., 2024; Lee et al., 2024). However, this paradigm inadvertently creates an avenue for watermark spoofing. Malicious actors can query a watermarked LLM to gather text samples, using this data to construct a surrogate model capable of approximating the underlying watermarking rules (Jovanović et al., 2024). By successfully forging the watermark, such attacks pose a fundamental threat to the reliability of existing watermarking systems.

To rigorously assess the spoofing resistance of LLM watermarking schemes, researchers have investigated various attack paradigms. However, existing methods typically suffer from at least one of three major limitations: (1) reliance on partial algorithmic knowledge or detector API access (Kirchenbauer et al., 2023; Chen et al., 2024); (2) the need for large training corpora (An et al., 2025; Gu et al., 2023) (up to 10k); or (3) an inability to generate semantically coherent text that seamlessly matches the original context (Pang et al., 2024; Gloaguen et al., 2024). These limitations reduce their practicality as evaluation tools for watermark spoofing resilience. As LLMs continue to advance, there is an urgent need for a practical, generalizable, and data-efficient methodology for benchmarking the robustness of watermarking schemes.

In this paper, we investigate the resilience of LLM watermarks to spoofing attacks. In the black-box setting, direct instance-level spoofing is intractable because the adversary has no access to the watermark details or the detector. Motivated by recent evidence that watermarked text exhibits systematic distributional discrepancy from human-written text (Huang et al., 2025), even for distortion-free watermarking schemes (Liu et al., 2024a), we take a *distributional view*

[1]Shanghai Jiao Tong University, Shanghai, China. Correspondence to: Shiyu Liang <lsy18602808513@sjtu.edu.cn>.

Accepted to Trustworthy AI for Good Workshop at ICML 2026, Seoul, South Korea.

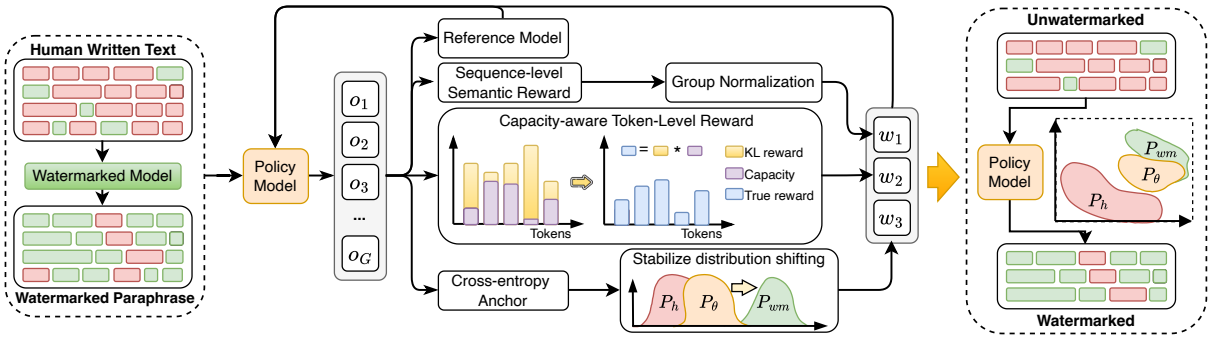


Figure 1. Overview of RLSpoofer. Given human–watermarked rewrite pairs, RLSpoofer jointly optimizes sequence-level semantic rewards, capacity-aware token-level rewards, and a cross-entropy anchor, shifting the policy distribution P_θ from the human-like distribution P_h toward the watermarked distribution P_{wm}, thereby effectively spoofing the watermark.

of watermark spoofing. Specifically, the attacker seeks to shift the paraphraser's output distribution away from the human-like distribution and toward the watermarked distribution.

Leveraging the autoregressive structure of LLM generation, we establish a *local capacity bottleneck* that characterizes how much token-level probability mass can be reallocated under KL-bounded local updates while preserving semantic fidelity. Building on this insight, we propose RLSpoofer, a reinforcement-learning-based black-box attack that requires no access to the watermarking internals or the detector. As illustrated in Figure 1, RLSpoofer jointly optimizes semantic fidelity, capacity-aware token-level reward, and a cross-entropy anchor, thereby shifting the model outputs toward the target watermarked distribution. Using only **100 training pairs**, RLSpoofer enables a compact 4B model to achieve a **62.0%** spoof success rate against PF-Watermark, substantially surpassing the *6.0%* of baseline trained on *10,000 examples*. Extensive experiments across five attacker models and six watermarking schemes show that RLSpoofer provides a practical, sample-efficient stress test for spoofing resilience, exposing critical vulnerabilities in current watermarking defenses. Our contributions are as follows:

- We establish a *local capacity bottleneck* that characterizes the token-level bottleneck on probability mass reallocation under KL-bounded local updates while preserving semantic integrity. (Sec. 3.1.)
- We propose RLSpoofer, an effective and sample-efficient black-box attack. It successfully spoofs watermarks with high semantic fidelity, requiring only limited human-watermarked rewrite pairs and zero access to the watermarking scheme or detector. (Sec. 3.2.)
- We conduct extensive experiments across five architectures and six watermarking schemes from both logit-based and sampling-based families. The results demonstrate the effectiveness of RLSpoofer, exposing critical vulnerabilities in current watermark defense. (Sec. 4.)

2. Preliminaries

LLM Paraphraser and Logit-based Watermark. Let $\mathcal{V}$ be a finite vocabulary and $\mathcal{V}^*$ denote the space of all finite token sequences. For a source input $\mathbf{X} \in \mathcal{V}^*$, an autoregressive paraphraser π_θ induces a probability distribution over generated output sequences $\mathbf{X}' = (x'_1, \ldots, x'_{|\mathbf{X}'|}) \in \mathcal{V}^*$: $P_\theta(\mathbf{X}' \mid \mathbf{X}) := \prod_{t=1}^{|\mathbf{X}'|} \pi_\theta(x'_t \mid x'_{<t}, \mathbf{X})$. Independent of the paraphrasing model, a watermark is embedded into generated text using a secret key s and a target green-token rate $g \in (0, 1)$. At each generation step, key s defines a prefix-dependent pseudorandom subset of the vocabulary with expected size $g|\mathcal{V}|$, whose tokens are softly upweighted during sampling. Given a sequence $\mathbf{X}'$, the detector f computes a score $f(\mathbf{X}', s)$ under the key s and flags the sequence watermarked whenever $f(\mathbf{X}', s) > \delta$, where δ is a fixed detection threshold.

Threat Model: Semantics-Preserving Paraphrase-based Spoofing. We consider a black-box adversary aiming to spoof the watermark into an unwatermarked sequence $\mathbf{X}$ via paraphrasing. The adversary is defined by: (1) ***Objective***: Generate an output $\mathbf{X}'$ that preserves the semantic meaning of $\mathbf{X}$ while triggering the watermark detector. (2) ***Knowledge***: The adversary lacks access to the secret key s, the detector function f, and the threshold δ, but can observe watermarked text by querying the generator. (3) ***Capability***: The adversary can construct a training set by querying the watermarked model for paraphrases of given texts, and can subsequently fine-tune a paraphrasing model π_θ without requiring access to the detector.

Adversarial Objective. Let $d : \mathcal{V}^* \times \mathcal{V}^* \to [0, \infty)$ be a distance metric where smaller values indicate higher semantic similarity. For a tolerance $\varepsilon > 0$, we define the instance-level spoof success rate as: $\mathrm{SSR}_{\mathbf{X}}(\theta) := P_\theta(\{\mathbf{X}' \in \mathcal{V}^* : d(\mathbf{X}', \mathbf{X}) < \varepsilon,\ f(\mathbf{X}', s) > \delta\} \mid \mathbf{X})$. The adversary seeks to maximize this rate. However, without access to f and s, direct optimization is intractable.

Distributional Surrogate. Prior work suggests a distribu-

tional discrepancy between watermarked and human-written text (Huang et al., 2025; Liu et al., 2024a). Since the detector function f and the secret key s are unavailable in the black-box setting, directly optimizing the instance-level spoof success rate is intractable. We therefore adopt a distributional surrogate objective that favors outputs relatively more likely under a watermarked distribution than under a human-like distribution, while enforcing semantic fidelity. Let $P_h(\cdot \mid \mathbf{X})$ and $P_{wm}(\cdot \mid \mathbf{X})$ denote the human-like and watermarked distributions, respectively. We consider the following surrogate objective based on the Kullback–Leibler (KL) divergences D_{KL}: $\max_\theta \; D_{\mathrm{KL}}\big(P_\theta(\cdot \mid \mathbf{X}) \,\|\, P_h(\cdot \mid \mathbf{X})\big) - D_{\mathrm{KL}}\big(P_\theta(\cdot \mid \mathbf{X}) \,\|\, P_{wm}(\cdot \mid \mathbf{X})\big)$.

Under the mild regularity conditions in Appendix A.1, the KL-difference objective admits an expected log-likelihood-ratio form. Imposing the semantic-fidelity constraint $P_\theta(d(X', X) < \varepsilon \mid X) \geq 1 - \rho$ for $\rho \in [0, 1)$ yields the following constrained surrogate objective:

$$\begin{aligned} \max_\theta \quad & \mathbb{E}_{\mathbf{X}' \sim P_\theta(\cdot|\mathbf{X})} \left[\log \frac{P_{wm}(\mathbf{X}' \mid \mathbf{X})}{P_h(\mathbf{X}' \mid \mathbf{X})} \right] \\ \text{s.t.} \quad & P_\theta\big(d(\mathbf{X}', \mathbf{X}) < \varepsilon \mid \mathbf{X}\big) \geq 1 - \rho. \end{aligned} \tag{1}$$

This formulation yields a tractable surrogate objective that prefers semantically faithful outputs with higher relative likelihood under the watermarked reference than under the human-like reference.

3. Methods

3.1. Local Capacity Bottleneck for Semantics-Preserving KL-Bounded Redistribution

To optimize the sequence-level objective in Eq. 1, we decompose it into token-level steps. Let the generation history at step t be $h_t := (x'_{<t}, \mathbf{X})$. Since both reference distributions are autoregressive, $\log \frac{P_{wm}(\mathbf{X}'|\mathbf{X})}{P_h(\mathbf{X}'|\mathbf{X})} = \sum_{t=1}^{|\mathbf{X}'|} \log \frac{P_{wm}(x'_t|h_t)}{P_h(x'_t|h_t)}$. Accordingly, the adversarial objective becomes

$$\begin{aligned} \max_\theta \quad & \mathbb{E}_{\mathbf{X}' \sim P_\theta(\cdot|\mathbf{X})} \sum_{t=1}^{|\mathbf{X}'|} \left[\log \frac{P_{wm}(x'_t \mid h_t)}{P_h(x'_t \mid h_t)} \right] \\ \text{s.t.} \quad & P_\theta\big(d(\mathbf{X}', \mathbf{X}) < \varepsilon \mid \mathbf{X}\big) \geq 1 - \rho. \end{aligned} \tag{2}$$

This decomposition identifies a token-level surrogate signal. To localize the sequence-level semantic constraint, we consider KL-bounded local updates around the human-like reference $P_h(\cdot \mid h_t)$, using such neighborhoods as a tractable proxy for compatible modifications. This leads to the following local capacity question: *under a fixed KL budget, how much probability mass can be reassigned?*

Formally, for a fixed history h_t, we consider a local update from $P_h(\cdot \mid h_t)$ to $\pi_\theta(\cdot \mid h_t)$ under a KL constraint, and bound the resulting increase of probability mass on an arbitrary subset of tokens. Let $d_{\mathrm{kl}}(p\|q) := p \log \frac{p}{q} + (1-p) \log \frac{1-p}{1-q}$, $p, q \in [0, 1]$, with the standard conventions $0 \log 0 := 0$ and $b \log(b/0) := +\infty$ for $b > 0$. We then establish the following theorem.

Theorem 3.1 (Local capacity characterization)**.** *Fix a history h_t. For any next-token distribution $\pi_\theta(\cdot \mid h_t)$ and any subset $A \subseteq \mathcal{V}$, we have*

$$\begin{aligned} &\pi_\theta(A \mid h_t) - P_h(A \mid h_t) \\ &\leq \sup\Big\{ \lambda \in [0, 1] : d_{\mathrm{kl}}\Big(\lambda \,\Big\|\, 1 - \max_{x \in \mathcal{V}} P_h(x \mid h_t)\Big) \\ &\qquad\qquad \leq D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid h_t) \,\|\, P_h(\cdot \mid h_t)\big) \Big\}. \end{aligned} \tag{3}$$

Moreover, for any fixed $C \geq 0$,

$$\lim_{1 - \max_{x \in \mathcal{V}} P_h(x|h_t) \to 0} \; \sup_{\pi_\theta :\, D_{\mathrm{KL}}(\pi_\theta(\cdot|h_t) \,\|\, P_h(\cdot|h_t)) \leq C} \; \sup_{A \subseteq \mathcal{V}} \big(\pi_\theta(A \mid h_t) - P_h(A \mid h_t)\big) = 0. \tag{4}$$

Remark 3.2. The proof is provided in Appendix A.2. Theorem 3.1 characterizes a local bottleneck for KL-bounded redistribution around the human-like reference: when the next-token distribution is highly concentrated, only limited probability mass can be reassigned under a fixed KL budget. While purely distributional, this provides a tractable local proxy for compatible modification. In particular, taking $A = \{x\}$ recovers the token-level case.

Let $x_t^* \in \arg\max_{x \in \mathcal{V}} P_h(x \mid h_t)$, and define $c_t := 1 - P_h(x_t^* \mid h_t)$. We refer to c_t as the *local capacity mass*. By Theorem 3.1, c_t quantifies the local bottleneck for semantics-consistent redistribution: a more concentrated human-like distribution leaves less probability mass available for reassignment under a fixed KL budget.

Moreover, if a local update preserves the dominant human-like continuation, i.e., $\pi_\theta(x_t^* \mid h_t) = P_h(x_t^* \mid h_t)$, then for $c_t > 0$, the local surrogate gain factorizes as

$$\begin{aligned} &\sum_{x \in \mathcal{V}} \big(\pi_\theta(x \mid h_t) - P_h(x \mid h_t)\big) \log \frac{P_{wm}(x \mid h_t)}{P_h(x \mid h_t)} \\ &= c_t \sum_{x \neq x_t^*} \frac{\pi_\theta(x \mid h_t) - P_h(x \mid h_t)}{c_t} \log \frac{P_{wm}(x \mid h_t)}{P_h(x \mid h_t)}. \end{aligned} \tag{5}$$

Thus, c_t serves as a local indicator of both the feasible amount of redistribution and the scale of the token-level surrogate gain. This suggests that token positions with larger c_t may admit more flexible redistribution, thereby motivating a token-dependent weighting scheme.

3.2. RLSpoofer: Capacity-Aware RL Attack for Watermark Spoofing

From surrogate objective to policy optimization. Equation 1 defines an expected sequence-level objective under the

paraphraser-induced distribution over discrete generations, which naturally admits an episodic Markov decision process formulation with π_θ serving as the policy. Motivated by this view, we propose RLSpoofer, a GRPO-based (Shao et al., 2024) watermark spoofing attack that requires only limited sample pairs $(\mathbf{X}, \mathbf{X}'_{wm})$, where $\mathbf{X}$ is a human-written text and $\mathbf{X}'_{wm}$ is its semantic-preserving paraphrase generated by a watermarked model. Using such pairs, the attack steers the paraphrasing distribution $P_\theta(\cdot \mid \mathbf{X})$ toward the watermarked distribution $P_{wm}(\cdot \mid \mathbf{X})$ and away from the human-written distribution $P_h(\cdot \mid \mathbf{X})$, thereby encouraging watermark-spoofed generations.

Approximating the target distributions. Since the ground-truth watermarked and human-like distributions are not directly accessible, we approximate both using a *lightweight* reference model π_{ref}. As LLMs are trained to model human text (Grattafiori et al., 2024), we approximate P_h by querying the π_{ref} conditioned on the original input $\mathbf{X}$, namely $P_h(x'_t \mid x'_{<t}, \mathbf{X}) \approx \pi_{\text{ref}}(x'_t \mid x'_{<t}, \mathbf{X})$. Similarly, motivated by the observation that weak rewriting often preserves both semantics and watermark patterns (Kirchenbauer et al., 2023), we approximate the watermarked distribution by conditioning the same reference model on the watermarked instance $\mathbf{X}'_{wm}$, i.e., $P_{wm}(x'_t \mid x'_{<t}, \mathbf{X}) \approx \pi_{\text{ref}}(x'_t | x'_{<t}, \mathbf{X}'_{wm})$. Intuitively, conditioning on the semantics-preserving rewrite X'_{wm} provides a tractable proxy for the watermarked distribution's lexical and semantic preferences, yielding a tractable local approximation of P_{wm}.

Reward design. Based on Eq. 2, the token-level spoofing signal is naturally given by $\log \frac{P_{wm}(x'_t|h_t)}{P_h(x'_t|h_t)}$. Moreover, Theorem 3.1 together with Eq. 5 suggest that, under semantics-consistent local updates around $P_h(\cdot \mid h_t)$, the attainable improvement of this signal is modulated by the local capacity mass c_t. Motivated by this, we define the reward for a sampled token $x'_t \sim \pi_\theta(\cdot \mid h_t)$ as $r_t := c_t \log \frac{P_{wm}(x'_t|h_t)}{P_h(x'_t|h_t)}$, where $c_t = 1 - \max_{x \in \mathcal{V}} P_h(x \mid h_t) \approx 1 - \max_{x \in \mathcal{V}} \pi_{\text{ref}}(x \mid x'_{<t}, \mathbf{X})$. The quantity c_t represents the human-plausible mass outside the dominant continuation, thereby capturing the local room for lexical redistribution under semantic fidelity. As a result, r_t rewards tokens preferred by the surrogate watermarked distribution while attenuating updates at positions where the human-like distribution is already sharply concentrated.

However, token-level redistribution alone is insufficient, since a successful attack must also preserve semantics at the sequence level. We therefore introduce a sequence-level reward A based on the P-SP score (Wieting et al., 2022), followed by a sigmoid transformation to improve gradient sensitivity. Specifically, we compute the semantic similarity between the generated paraphrase and each of $\mathbf{X}$ and $\mathbf{X}'_{wm}$, and take the minimum as A. This conservative design preserves the meaning of the original text while keeping the policy close to the watermarked rewrite, thereby improving the quality of the surrogate target for watermark-favored generation. We further normalize A within each rollout group to obtain $\hat{A}_i = \frac{A_i - \text{mean}(A)}{\text{std}(A)}$, which stabilizes training and improves semantic consistency across samples.

Training objective. Recent work has shown that distillation-based objectives can be effective for watermark spoofing (Gu et al., 2023; An et al., 2025). Accordingly, in addition to the reward design above, RLSpoofer also incorporates a cross-entropy term to anchor optimization toward the target watermarked distribution. Concretely, the attack policy π_θ is iteratively updated by sampling a set of outputs $\{x'_1, \ldots, x'_G\}$ from the previous policy $\pi_{\theta_{\text{old}}}$, maximizing the following training objective: $\mathcal{J}(\theta) \approx$

$$\mathbb{E}_{\{x'_i\} \sim \pi_{\theta_{\text{old}}}} \frac{1}{G} \sum_{i=1}^{G} \frac{1}{|x'_i|} \sum_{t=1}^{|x'_i|} \left[\frac{\pi_\theta(x'_{i,t}|\mathbf{X}, x'_{i,<t})}{\pi_{\theta_{\text{old}}}(x'_{i,t}|\mathbf{X}, x'_{i,<t})} \left(w_1 \hat{A}_i + w_2 r_{i,t} \right) - \beta D_{\text{KL}} \left[\pi_\theta \,\|\, \pi_{\text{ref}} \right] \right] - w_3 \, \mathcal{L}_{\text{CE}} \left(\pi_\theta, \{(\mathbf{X}, \mathbf{X}'_{wm})\} \right),$$

where π_{ref} denotes the reference model, and w_1, w_2, w_3 control the contributions of the respective components. The KL term regularizes the policy against the reference model at the token level, while $\mathcal{L}_{\text{CE}}$ denotes the teacher-forced cross-entropy loss for predicting $\mathbf{X}'_{wm}$ conditioned on $\mathbf{X}$.

4. Experiments

4.1. Experimental Setup

We provide detailed experimental setups and configurations in Appendix B.

Victim Models and Attackers. We use Llama3.1-8B-Instruct (Grattafiori et al., 2024) as the victim model to generate watermarked paraphrases. For attackers, we consider five lightweight models of varying sizes from three well-known model families: Qwen3-0.6B, Qwen3-1.7B, Qwen3-4B (Yang et al., 2025), Qwen2.5-3B-Instruct (Qwen et al., 2025), and Llama3.2-3B-Instruct. We also include Qwen3-8B for surrogate-sensitivity analysis.

Watermarking Schemes. We evaluate six watermarking schemes drawn from both logit-based and sampling-based families. Specifically, the logit-based methods include EWD (Lu et al., 2024), SWEET (Lee et al., 2024), KGW (Kirchenbauer et al., 2023), and Unigram (Zhao et al., 2023). To capture recent advances, we additionally consider two sampling-based distortion-free methods: semantic watermark PMark (Huo et al., 2025) and the cryptographic PF-Watermark (Zhao et al., 2024). We implement PMark based on its official codebase, whereas the remaining schemes are implemented and detected using the widely adopted MarkLLM toolkit (Pan et al., 2024).

Datasets. We construct the training set from C4-RealNewslike subset (Raffel et al., 2020) and the test set from four datasets: Reddit WritingPrompts (Verma et al., 2024), LFQA (Krishna et al., 2023), and the BookReport and FakeNews subsets of MMW (Piet et al., 2025), following (Jovanović et al., 2024). Specifically, we prompt Qwen3-8B to generate human-like unwatermarked texts and use watermarked Llama3.1-8B-Instruct to rewrite them into semantically preserved watermarked responses. For each watermarking scheme, we use 100 unwatermarked–watermarked response pairs for training and 400 unwatermarked test samples drawn evenly from the four datasets. We standardize the length of all training and test samples to 500 tokens.

Spoof Methods. Under the black-box threat model, in addition to RLSpoofer, we consider three spoofing baselines to further reveal vulnerabilities in watermarking schemes. **Distill** (Gu et al., 2023) distills the distribution of the watermarked model using 10,000 human–watermarked rewritten pairs. **DITTO** (An et al., 2025) further exploits the distilled model by analyzing its output distribution and reproducing the statistical preferences induced by the target watermark. Additionally, **DPO** (Diaa et al., 2024) casts watermark spoofing as a preference alignment and optimizes the attacker on 7,000 samples preference pairs.

Implementation Details of RLSpoofer. RLSpoofer utilizes a reparameterized sigmoid scaling function for the semantic reward, with a threshold of 0.85 to separate positive and negative rewards, thereby encouraging higher-quality rephrasings. Training on 100 samples with a batch size of 48 and group size $G = 12$ converges efficiently, completing in approximately 1.5 hours for Qwen3-4B and 45 minutes for Qwen3-0.6B on four NVIDIA Pro 6000 GPUs. In the experiments, we use the attack model itself as the reference model to generate the surrogate watermark distribution.

Metric. We primarily evaluate spoofing effectiveness using Spoof Success Rate (SSR), defined as the proportion of rephrased texts detected as watermarked under a semantic similarity constraint (P-SP ¿ 0.7, following prior work (Jovanović et al., 2024)). We also report Spoof Rate (SR), which omits the semantic constraint. Rephrasing quality is further assessed using perplexity and GPT-as-a-Judge (GPTS) (Cheng et al., 2025).

4.2. Main Results

In this subsection, we evaluate the spoofing resilience of different watermarking schemes across four attacks, with results shown in Table 1. Details of rephrase quality are provided in Appendix C.1.

Logit-based watermarks are vulnerable to baseline attacks. We observe that across all attack settings, *Distill* consistently achieves strong spoofing performance against logit-based watermarks. As shown in Table 1, Distill enables Qwen3-0.6B to achieve 42.3% SSR and a P-SP score of 0.75 on EWD, significantly outperforming DITTO (7.5%) and DPO (0.25%), with similar trends observed across models. Furthermore, Table 2 shows that Distill and DITTO achieve spoof rates (SR) of 62.5% and 94.3%, respectively, on Qwen3-4B with EWD. This effectiveness arises because logit-based watermarking induces pronounced shifts in the output distribution (Liu et al., 2024a), which can be captured by distribution-matching approaches. In contrast, DPO yields consistently low SSR, spoof rates, and P-SP scores across most watermarks, suggesting that preference-based optimization is less suitable for watermark spoofing.

Table 2. Spoofing performance on EWD using Qwen3-4B.

Method	SSR	SR	P-SP
Distill	51.3	62.5	0.81
DITTO	56.0	**94.3**	0.68
DPO	0.25	0.50	0.78
RLSpoofer	**56.5**	87.0	0.73

Sampling-based distortion free watermarks are more challenging to spoof. We observe that all three baselines struggle to spoof sampling-based distortion-free watermarks effectively. Distill achieves only ∼7% SSR on PF-Watermark and at most 23.3% on PMark, with the other baselines showing similarly limited performance. Furthermore, as shown in Table 3, all three methods obtain spoof rates below 9% on PF-Watermark, indicating that they fail to replicate the watermark distribution. We hypothesize this resilience stems from the fact that PF and PMark watermarks are distortion-free in expectation (Zhao et al., 2024), rendering their latent signals exceptionally difficult to model from large training corpora.

Table 3. Spoofing performance on PF using Qwen3-4B.

Method	SSR	SR	P-SP
Distill	6.00	6.50	0.96
DITTO	3.50	6.25	0.87
DPO	5.25	8.75	0.88
RLSpoofer	**62.0**	**82.8**	0.77

RLSpoofer consistently compromises the resilience of watermarking schemes. We observe that, with only 100 training samples, RLSpoofer matches or even surpasses baselines requiring 10,000 samples across evaluated watermarks. Specifically, on SWEET, it enables Qwen3-0.6B to achieve a 50.5% spoofing success rate (SSR), significantly exceeding Distill (20.0%) and DITTO (6.75%). Furthermore, RLSpoofer empowers Qwen3-4B to effectively spoof the PF watermark with a 62.0% SSR, far surpassing the best baseline of 8.75%. This efficacy stems from its ability to align with the distribution of watermarked text, successfully exploiting the sample-level distribution shifts induced even by distortion-free watermarks under a fixed key (Liu et al., 2024a). Our analysis of EWD and SWEET detection scores (400 samples, 500 tokens per input) confirms this alignment: as shown in Figure 2(a), RLSpoofer shifts the

Table 1. Spoof Success Rate (SSR, %) and P-SP scores across models and watermarking schemes. **Bold** indicates the *best* SSR for each model. Higher SSR indicates stronger spoofing performance.

		EWD		SWEET		KGW		Unigram		PF		PMark	
Models	**Methods**	**SSR**	**P-SP**	**SSR**	**P-SP**	**SSR**	**P-SP**	**SSR**	**P-SP**	**SSR**	**P-SP**	**SSR**	**P-SP**
Qwen3-0.6B	Distill	42.3	0.75	20.0	0.76	35.8	0.68	13.8	0.84	6.50	0.97	20.0	0.90
	DITTO	7.50	0.43	6.75	0.41	1.00	0.33	0.25	0.32	5.50	0.79	11.8	0.63
	DPO	0.25	0.57	0.00	0.76	1.00	0.66	0.25	0.32	2.50	0.57	6.25	0.63
	RLSpoofer	**54.3**	0.73	**50.5**	0.79	**52.0**	0.72	**49.5**	0.70	**33.3**	0.66	**29.5**	0.92
Qwen3-1.7B	Distill	43.8	0.80	26.8	0.79	44.5	0.75	19.5	0.81	7.00	0.96	20.3	0.90
	DITTO	21.5	0.53	29.0	0.56	13.8	0.52	1.50	0.36	7.50	0.86	16.5	0.68
	DPO	0.25	0.94	0.00	0.84	1.00	0.96	0.50	0.87	4.25	0.58	22.5	0.93
	RLSpoofer	**53.5**	0.76	**52.0**	0.71	**52.0**	0.71	**54.8**	0.73	**29.0**	0.73	**29.5**	0.90
Qwen3-4B	Distill	51.3	0.81	37.3	0.82	57.0	0.79	28.0	0.80	6.00	0.96	21.5	0.92
	DITTO	56.0	0.68	43.3	0.66	36.5	0.61	2.25	0.39	3.50	0.88	16.5	0.71
	DPO	0.25	0.78	0.00	0.78	1.00	0.93	0.75	0.59	5.25	0.88	17.5	0.68
	RLSpoofer	**56.5**	0.73	**52.3**	0.75	**58.0**	0.75	**54.8**	0.74	**62.0**	0.77	**36.3**	0.91
Qwen2.5-3B -Instruct	Distill	**55.5**	0.76	49.8	0.77	**60.3**	0.74	25.8	0.80	6.50	0.93	22.3	0.91
	DITTO	14.0	0.50	22.3	0.54	9.50	0.48	0.25	0.34	5.25	0.72	11.3	0.56
	DPO	0.00	0.88	0.00	0.87	1.25	0.87	2.50	0.78	6.25	0.83	24.3	0.95
	RLSpoofer	53.5	0.70	**54.5**	0.75	57.3	0.72	**54.5**	0.77	**50.3**	0.68	**30.3**	0.89
Llama3.2-3B -Instruct	Distill	53.8	0.77	45.5	0.76	**56.3**	0.75	26.0	0.77	8.75	0.93	23.3	0.89
	DITTO	19.3	0.54	24.3	0.56	14.0	0.51	1.00	0.35	6.50	0.79	18.0	0.65
	DPO	2.50	0.49	0.50	0.53	0.75	0.36	7.75	0.60	6.25	0.67	25.0	0.87
	RLSpoofer	**54.5**	0.70	**54.5**	0.74	55.3	0.76	**52.0**	0.72	**49.8**	0.85	**33.3**	0.92

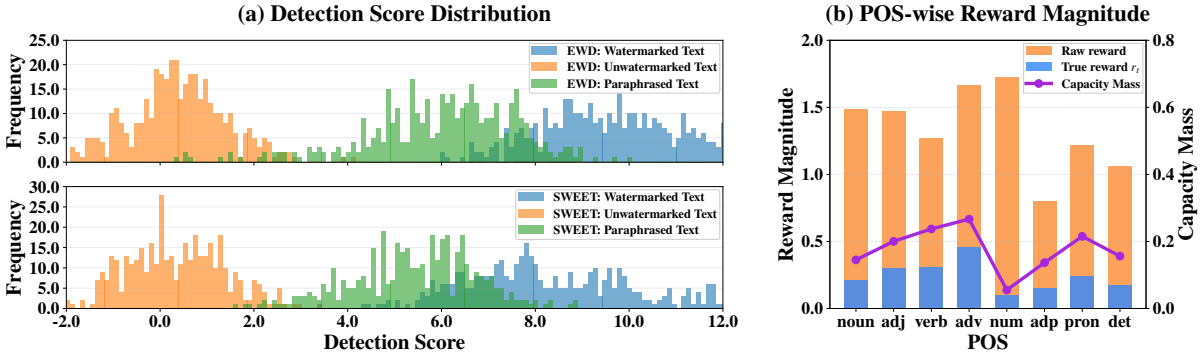


Figure 2. (a) illustrates the detection score distributions for EWD and SWEET watermarks across unwatermarked outputs, watermarked outputs, and paraphrased texts generated by Qwen3-4B trained with RLSpoofer; (b) shows the POS-wise reward magnitudes for SWEET watermark on Qwen3-4B.

z-score distribution of its generated rephrasings toward that of watermarked text, maintaining clear separation from the unwatermarked distribution. Details are in Appendix C.2.

4.3. Empirical Analysis of RLSpoofer

In this subsection, we analyze the effectiveness of RLSpoofer on three complementary perspectives: feasible token-level watermark injection, sequence-level semantic preservation, and optimization stability. Additional details can be found in Appendix C.3 - C.5.

Local capacity mass identifies feasible room for watermark injection. Theorem 3.1 suggests that the local capacity mass c_t captures the local bottleneck of semantics-consistent redistribution, and thus favors positions with greater semantics-preserving substitutability. Since c_t directly modulates token-level rewards, its effect should be reflected in the POS-wise distribution of r_t. We therefore plot POS-wise token-level rewards of the Qwen3-4B RLSpoofer on the SWEET training set. Figure 2 (b) shows that the induced reward is small on numerals, which are typically critical for semantic fidelity, but larger on adjectives, verbs, and adverbs, which allow greater lexical flexibility. Thus, local capacity mass suppresses overly sharp rewards on semantically rigid tokens while amplifying rewards on more substitutable ones. This suggests that RLSpoofer exploits genuinely flexible positions rather than uniformly favoring all watermark-preferred tokens.

We next verify that this weighting is not only intuitively aligned with semantic flexibility but also critical to spoofing

Table 4. SSR on reward weights.

Model	Weight	EWD	SWEET
Qwen3 (0.6B)	1 - $p_{\max}$	**54.3**	**50.5**
	uniform	42.8	39.0
	$p_{\max}$	40.5	29.8
Qwen3 (4B)	1 - $p_{\max}$	**56.5**	**52.3**
	uniform	35.5	28.8
	$p_{\max}$	28.0	25.0

Table 5. SSR across semantic rewards.

Model	Scheme	Min.	Avg.	Hum.	W.M.
Qwen3 (0.6B)	EWD	**54.3**	44.8	42.3	48.5
	SWEET	**50.5**	48.8	43.0	46.8
	PF	**33.3**	26.3	24.5	25.5
Qwen3 (4B)	EWD	**56.5**	47.3	47.5	45.5
	SWEET	**52.3**	34.3	47.0	48.5
	PF	**62.0**	50.3	48.5	51.0

Table 6. CE-anchor ablation (SSR vs. RLS.).

Model	Scheme	W.O.	Dis.(100)
Qwen3 (0.6B)	EWD	29.3 (-25.0)	0.25 (-54.0)
	SWEET	27.3 (-23.2)	0.25 (-50.3)
	PF	12.3 (-19.0)	0.00 (-31.3)
Qwen3 (4B)	EWD	33.5 (-23.0)	0.00 (-56.5)
	SWEET	26.8 (-25.5)	0.25 (-52.0)
	PF	25.5 (-36.5)	0.00 (-62.0)

performance. Specifically, we replace 1 - $p_{\max}$ with either a uniform weight of 1 or the reverse weight $p_{\max}$. Table 4 shows that both replacements consistently degrade performance. On EWD, uniform weighting and $p_{\max}$ reduce the SSR of Qwen3-4B from 56.5% to 35.5% and 28.0%, respectively; similar trends hold on SWEET and for Qwen3-0.6B. These results show that spoofing does not benefit from uniformly enlarging token-level rewards. Instead, it requires identifying contexts with sufficient semantically permissible redistribution room, where probability mass can be shifted toward watermark-preferred continuations without compromising semantic fidelity.

A conservative semantic reward improves surrogate quality. Beyond identifying feasible token positions, successful spoofing must preserve the meaning of the original text while staying close to the watermarked rewrite, which serves as a surrogate for the watermark-favored distribution. We therefore use a conservative semantic reward (Min.), defined by the weaker of the two semantic matches. Compared with averaging the two matches (Avg.) or using only one reference (Hum./W.M.), our design avoids one-sided alignment and produces a better surrogate target for optimization. As shown in Table 5, we observe Min. consistently achieves the best SSR across models and watermarking schemes, suggesting that successful spoofing benefits from preserving the original meaning while maintaining a better surrogate target for optimization.

Cross-entropy (CE) anchoring stabilizes optimization. We observe that even with carefully designed token-level and semantic rewards, unconstrained policy optimization can still drift away from the base model and exploit brittle reward shortcuts. Cross-entropy anchoring mitigates this by keeping optimization aligned with the base-model distribution while still allowing watermark-favored redistribution. As shown in Table 6, removing the anchor (W.O.) causes large SSR drops of 23.0 - 36.5 points, while replacing RL with supervised distillation on the same 100 training pairs (Dis.(100)) leads to near-complete failure. These results indicate that the gain comes from stable distributional alignment rather than limited-pair imitation alone.

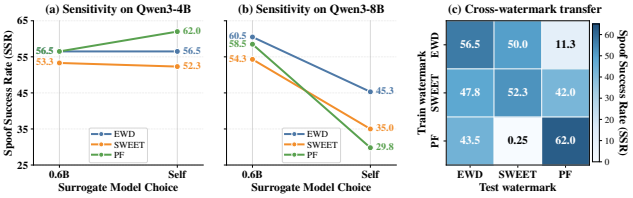


Figure 3. (a) and (b) compare SSR of RLSpoofer under two surrogate choices, Qwen3-0.6B (0.6B) and attack model (Self). (c) shows cross watermark transferability on Qwen3-4B.

4.4. Ablation Study

In this subsection, we study the sensitivity of RLSpoofer to the training data, the choice of surrogate distribution generated model, and its ability to generalize to OOD test data.

Table 7. SSR across Training set.

Models	Scheme	Samples 50	100	200
Qwen3 (0.6B)	EWD	44.3	54.3	56.8
	SWEET	46.5	50.5	51.3
	PF	9.25	31.3	20.5
Qwen3 (4B)	EWD	46.5	56.5	58.5
	SWEET	49.3	52.3	52.3
	PF	20.8	60.8	40.3

Small training sets suffice for strong RLSpoofer performance. We examine the impact of training set size on RLSpoofer under three watermarking schemes, EWD, SWEET, and PF, using Qwen3-0.6B and Qwen3-4B. Table 7 shows that RLSpoofer can achieve strong spoofing performance with only 50 training samples. For instance, on EWD, Qwen3-4B attains an SSR of 46.5%, which increases to 58.5% with 200 samples. A similar pattern holds for SWEET. However, for the PF watermark, enlarging the training set does not yield consistent gains in SSR. This is likely because PF's distortion-free design creates a distribution gap that becomes harder to capture with more data. Overall, these findings confirm that relatively small training sets are sufficient for RLSpoofer to obtain strong spoofing performance. We provide details in Appendix C.6.

RLSpoofer exhibits increased sensitivity to surrogate selection as attacker capability grows. We evaluate how the surrogate model used to approximate the target watermarked distribution impacts RLSpoofer's efficacy. Specifi-

cally, we evaluate two surrogates, Qwen3-0.6B (0.6B) and the base attacker (Self), against EWD, SWEET, and PF watermarks across the 4B and 8B Qwen3 attackers. Fig. 3 demonstrates that surrogate selection minimally affects the weaker Qwen3-4B, which yields comparable SSR across all settings. Conversely, the stronger Qwen3-8B is highly sensitive: the 0.6B surrogate consistently outperforms the Self surrogate. For instance, on the PF watermark, utilizing 0.6B instead of Self elevates the SSR from 29.8% to 58.5%, with substantial gains similarly observed for EWD and SWEET. We attribute this disparity to Qwen3-8B's superior watermark-removal capabilities (Huang et al., 2025); its rewrites heavily weaken the embedded signal, rendering the model itself an ineffective proxy for the target watermark distribution. Details are in Appendix C.7.

RLSpoofer demonstrates strongly asymmetric and directional cross-watermark transfer. We evaluate zero-shot transferability with Qwen3-4B by training RLSpoofer on one watermarking scheme and testing it on the others. As shown in Fig. 3 (c), transferability is not strictly constrained by watermark family. Although the KGW-style, logit-based watermarks EWD and SWEET (Lu et al., 2024; Lee et al., 2024) exhibit substantial bidirectional transfer (SSR $\sim$ 47.8% - 50%), interactions involving the sampling-based PF watermark are notably directional. Specifically, RLSpoofer transfers effectively from SWEET to PF (SSR 47%) and from PF to EWD (52.5%), whereas the reverse directions are far weaker: PF to SWEET achieves only 0.25% SSR, and EWD to PF only 6%. This pronounced asymmetry suggests that transferability is not determined solely by mechanism-level similarity, but also by more intricate and directional overlaps in the vulnerabilities induced by different watermarking schemes. We provide details in Appendix C.8.

5. Related Works

LLM watermarks. As LLMs become increasingly widespread, the human-written and machine-generated text grows harder to distinguish. By embedding imperceptible yet algorithmically detectable signals into generated text, watermarking has become important tool for content attribution, copyright protection, and the mitigation of malicious misuse (Wu et al., 2025; Liu et al., 2024b; Zhang et al., 2024). Existing methods mainly follow two paradigms. *Logit-based* approaches (Kirchenbauer et al., 2023; Lu et al., 2024) use a secret pseudo-random hash key to partition the vocabulary into *green* and *red* token lists at each generation step. They then bias the logits of green-list tokens to increase their sampling probability, and detection is performed through statistical tests for green-token overrepresentation (Jovanović et al., 2024). Conversely, *sampling-based* methods (Zhao et al., 2024; Huo et al., 2025; Hou et al., 2024) embed watermark signals directly into the sampling process, rather than modifying logits explicitly, offering finer-grained control while reducing quality degradation. Across both paradigms, recent advances prioritize minimizing statistical distortion, ensuring watermarked outputs remain indistinguishable from natural text (Kuditipudi et al., 2023). Beyond such distributional concerns, recent work further shows that LLM watermarking can inadvertently degrade model alignment and safety behavior (Verma et al., 2025).

LLM watermark spoofing attack. LLM watermark *spoofing attacks* seek to forge a target watermark signal and thereby falsely attribute arbitrary text to a specific model provider. Existing spoofing methods can generally be grouped into three categories. The first, *piggyback spoofing* (Pang et al., 2024), makes subtle edits to authentically watermarked text in order to inject toxic content while preserving detector confidence. However, this strategy substantially limits the attacker's semantic flexibility. The second, *feedback-guided spoofing* attacks (Pang et al., 2024; Zhou et al., 2024), rely on repeated interaction with the watermark detector during generation to search for text that remains watermarked. Such attacks are typically query-intensive and depend on detector access, limiting their practical applicability. The third, *learning-based spoofing* (Jovanović et al., 2024; An et al., 2025; Gu et al., 2023), queries the target model to construct a training set for a surrogate model that internalizes the watermark signal. However, these approaches typically require either the knowledge of the watermarking mechanism (Jovanović et al., 2024) or a large amount of training data (e.g., up to 10K samples (An et al., 2025)). Consequently, the restrictive assumptions and inefficiencies of these approaches make them unsuitable for practically evaluating the spoofing resilience of watermarking schemes.

6. Conclusion

In this paper, we study LLM watermark spoofing from a distributional perspective. We identify a local capacity bottleneck that limits how token-level probability mass can be redistributed under KL-bounded, semantics-consistent local updates. Motivated by this, we propose RLSpoofer, a black-box RL-based spoofing attack requiring only 100 human–watermarked paraphrase pairs, with no access to watermark internals or detectors. Across five attacker models and six watermarking schemes, RLSpoofer achieves strong spoofing performance against both logit-based and sampling-based watermarks. These results suggest that current LLM watermarking designs remain vulnerable to spoofing, and that RLSpoofer can serve as a useful stress test for developing more spoofing-resistant schemes. We discuss limitations of RLSpoofer in Appendix C.9.

Impact Statement

RLSpoofer provides a lightweight and practical stress test for evaluating whether current watermarking schemes remain reliable under realistic black-box attacks, which may help the community design more robust provenance and detection mechanisms. At the same time, because our method can expose concrete weaknesses in existing watermarking systems, it also carries dual-use risk if deployed irresponsibly. To mitigate this concern, we focus on evaluation rather than misuse, and provide sufficient implementation details for scientific assessment.

References

An, H., Park, S., Woo, S., and Han, Y.-S. Ditto: A spoofing attack framework on watermarked llms via knowledge distillation. *arXiv preprint arXiv:2510.10987*, 2025.

Chen, C. and Shu, K. Can llm-generated misinformation be detected? *arXiv preprint arXiv:2309.13788*, 2023.

Chen, R., Wu, Y., Guo, J., and Huang, H. De-mark: Watermark removal in large language models. *arXiv preprint arXiv:2410.13808*, 2024.

Cheng, Y., Guo, H., Li, Y., and Sigal, L. Revealing weaknesses in text watermarking through self-information rewrite attacks. *arXiv preprint arXiv:2505.05190*, 2025.

Cotton, D. R., Cotton, P. A., and Shipway, J. R. Chatting and cheating: Ensuring academic integrity in the era of chatgpt. *Innovations in education and teaching international*, 61(2):228–239, 2024.

Diaa, A., Aremu, T., and Lukas, N. Optimizing adaptive attacks against watermarks for language models. *arXiv preprint arXiv:2410.02440*, 2024.

Gloaguen, T., Jovanović, N., Staab, R., and Vechev, M. Discovering spoofing attempts on language model watermarks. *arXiv preprint arXiv:2410.02693*, 2024.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Gu, C., Li, X. L., Liang, P., and Hashimoto, T. On the learnability of watermarks for language models. *arXiv preprint arXiv:2312.04469*, 2023.

Hou, A., Zhang, J., Wang, Y., Khashabi, D., and He, T. k-semstamp: A clustering-based semantic watermark for detection of machine-generated text. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 1706–1715, 2024.

Huang, H., Zhang, Y., Zheng, H., Gong, X., Li, Y., Liu, L., and Liang, S. Rlcracker: Exposing the vulnerability of llm watermarks with adaptive rl attacks. *arXiv preprint arXiv:2509.20924*, 2025.

Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025. URL https://github.com/huggingface/open-r1.

Huo, J., Liu, S., Wang, B., Zhang, J., Yan, Y., Liu, A., Hu, X., and Zhou, M. Pmark: Towards robust and distortion-free semantic-level watermarking with channel constraints. *arXiv preprint arXiv:2509.21057*, 2025.

Jovanović, N., Staab, R., and Vechev, M. Watermark stealing in large language models. *arXiv preprint arXiv:2402.19361*, 2024.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models. In *International Conference on Machine Learning*, pp. 17061–17084. PMLR, 2023.

Krishna, K., Song, Y., Karpinska, M., Wieting, J., and Iyyer, M. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in neural information processing systems*, 36:27469–27500, 2023.

Kuditipudi, R., Thickstun, J., Hashimoto, T., and Liang, P. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

Lee, T., Hong, S., Ahn, J., Hong, I., Lee, H., Yun, S., Shin, J., and Kim, G. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4890–4911, 2024.

Liu, A., Guan, S., Liu, Y., Pan, L., Zhang, Y., Fang, L., Wen, L., Yu, P. S., and Hu, X. Can watermarked llms be identified by users via crafted prompts? *arXiv preprint arXiv:2410.03168*, 2024a.

Liu, A., Sheng, Q., and Hu, X. Preventing and detecting misinformation generated by large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 3001–3004, 2024b.

Lu, Y., Liu, A., Yu, D., Li, J., and King, I. An entropy-based text watermarking detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11724–11735, 2024.

Pan, L., Liu, A., He, Z., Gao, Z., Zhao, X., Lu, Y., Zhou, B., Liu, S., Hu, X., Wen, L., et al. Markllm: An open-source toolkit for llm watermarking. *arXiv preprint arXiv:2405.10051*, 2024.

Pang, Q., Hu, S., Zheng, W., and Smith, V. Attacking llm watermarks by exploiting their strengths. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*, 2024.

Piet, J., Sitawarin, C., Fang, V., Mu, N., and Wagner, D. Markmywords: Analyzing and evaluating language model watermarks. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 68–91. IEEE, 2025.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shumailov, I., Shumaylov, Z., Zhao, Y., Gal, Y., Papernot, N., and Anderson, R. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*, 2023.

Verma, A., Phan, N., and Trivedi, S. Watermarking degrades alignment in language models: Analysis and mitigation. *arXiv preprint arXiv:2506.04462*, 2025.

Verma, V., Fleisig, E., Tomlin, N., and Klein, D. Ghostbuster: Detecting text ghostwritten by large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1702–1717, 2024.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., and Lambert, N. Trl: Transformer reinforcement learning. https://github.com/lvwerra/trl, 2020.

Wieting, J., Gimpel, K., Neubig, G., and Berg-Kirkpatrick, T. Paraphrastic representations at scale. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 379–388, 2022.

Wu, J., Yang, S., Zhan, R., Yuan, Y., Chao, L. S., and Wong, D. F. A survey on llm-generated text detection: Necessity, methods, and future directions. *Computational Linguistics*, 51(1):275–338, 2025.

Xu, Z., Yue, X., Wang, Z., Liu, Q., Zhao, X., Zhang, J., Zeng, W., Xing, W., Kong, D., Lin, C., et al. Copyright protection for large language models: A survey of methods, challenges, and trends. *arXiv preprint arXiv:2508.11548*, 2025.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Zhang, R., Hussain, S. S., Neekhara, P., and Koushanfar, F. {REMARK-LLM}: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pp. 1813–1830, 2024.

Zhao, X., Ananth, P., Li, L., and Wang, Y.-X. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*, 2023.

Zhao, X., Li, L., and Wang, Y.-X. Permute-and-flip: An optimally stable and watermarkable decoder for llms. *arXiv preprint arXiv:2402.05864*, 2024.

Zheng, Y., Zhang, R., Zhang, J., Ye, Y., Luo, Z., Feng, Z., and Ma, Y. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL http://arxiv.org/abs/2403.13372.

Zhou, T., Zhao, X., Xu, X., and Ren, S. Bileve: Securing text provenance in large language models against spoofing with bi-level signature. *Advances in Neural Information Processing Systems*, 37:56054–56075, 2024.

A. Details for the distributional surrogate

A.1. Absolute continuity and finite KL divergences

In this subsection, we state the regularity conditions under which the distributional surrogate in Section 2 is well defined and can be expanded into the expected log-likelihood-ratio objective in Eq. 1. These conditions ensure that the relevant KL divergences are finite and that the log-likelihood ratio is integrable under the attack distribution.

Assumption A.1 (Absolute continuity and finite KL divergences)**.** For a fixed input $\mathbf{X}$, $P_\theta(\cdot \mid \mathbf{X})$ is absolutely continuous with respect to both reference distributions ($P_\theta \ll P_h$ and $P_\theta \ll P_{wm}$). Furthermore, the log-likelihood ratio is integrable: $\mathbb{E}_{\mathbf{X}'\sim P_\theta}[|\log(P_{wm}(\mathbf{X}' \mid \mathbf{X})/P_h(\mathbf{X}' \mid \mathbf{X}))|] < \infty$. Moreover, $D_{\mathrm{KL}}\big(P_\theta(\cdot \mid \mathbf{X}) \,\|\, P_h(\cdot \mid \mathbf{X})\big) < \infty$ and $D_{\mathrm{KL}}\big(P_\theta(\cdot \mid \mathbf{X}) \,\|\, P_{wm}(\cdot \mid \mathbf{X})\big) < \infty$.

Under Assumption A.1, both KL terms in the surrogate objective are well defined, and their difference can be expanded as

$$D_{\mathrm{KL}}\big(P_\theta \,\|\, P_h\big) - D_{\mathrm{KL}}\big(P_\theta \,\|\, P_{wm}\big) = \mathbb{E}_{\mathbf{X}'\sim P_\theta(\cdot|\mathbf{X})}\left[\log \frac{P_{wm}(\mathbf{X}' \mid \mathbf{X})}{P_h(\mathbf{X}' \mid \mathbf{X})}\right],$$

where we suppress the conditioning on $\mathbf{X}$ for readability.

A.2. Proof of Theorem 3.1

In this subsection, we use the binary KL divergence

$$d_{\mathrm{kl}}(p\|q) := p\log\frac{p}{q} + (1-p)\log\frac{1-p}{1-q}, \qquad p, q \in [0,1],$$

with the standard conventions $0\log(0/q) := 0$ for $q \in [0,1]$, $0\log(0/0) := 0$, and $b\log(b/0) := +\infty$ for $b > 0$.

For convenience, we restate Theorem 3.1.

Theorem 1 (Local capacity characterization). Fix a history h_t. For any next-token distribution $\pi_\theta(\cdot \mid h_t)$ and any subset $A \subseteq \mathcal{V}$, we have

$$\pi_\theta(A \mid h_t) - P_h(A \mid h_t) \le \sup\Big\{\lambda \in [0,1] : d_{\mathrm{kl}}\Big(\lambda \,\Big\|\, 1 - \max_{x\in\mathcal{V}} P_h(x \mid h_t)\Big) \le D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid h_t) \,\|\, P_h(\cdot \mid h_t)\big)\Big\}. \quad (6)$$

Moreover, for any fixed $C \ge 0$,

$$\lim_{1-\max_{x\in\mathcal{V}} P_h(x|h_t)\to 0} \sup_{\pi_\theta:\, D_{\mathrm{KL}}(\pi_\theta(\cdot|h_t)\,\|\,P_h(\cdot|h_t))\le C} \sup_{A\subseteq\mathcal{V}} \big(\pi_\theta(A \mid h_t) - P_h(A \mid h_t)\big) = 0. \quad (7)$$

Proof. Fix a history h_t, and let $x_t^* \in \arg\max_{x\in\mathcal{V}} P_h(x \mid h_t)$ and $q := 1 - P_h(x_t^* \mid h_t) = 1 - \max_{x\in\mathcal{V}} P_h(x \mid h_t)$. Thus, q is the total mass assigned by $P_h(\cdot \mid h_t)$ to all tokens other than the dominant human-like continuation x_t^*.

We first prove Eq. equation 6. Consider the measurable map $T : \mathcal{V} \to \{0,1\}$ defined by $T(x) = \mathbf{1}\{x \ne x_t^*\}$, which induces the binary partition $\{x_t^*\}$ and $\mathcal{V}\setminus\{x_t^*\}$. Under $P_h(\cdot \mid h_t)$, the pushforward distribution of T is $(1-q, q)$, whereas under $\pi_\theta(\cdot \mid h_t)$ it is $\big(\pi_\theta(x_t^* \mid h_t),\, 1 - \pi_\theta(x_t^* \mid h_t)\big)$. By the data processing inequality for KL divergence under the map T,

$$d_{\mathrm{kl}}\big(1 - \pi_\theta(x_t^* \mid h_t) \,\|\, q\big) \le D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid h_t) \,\|\, P_h(\cdot \mid h_t)\big). \quad (8)$$

Hence,

$$1 - \pi_\theta(x_t^* \mid h_t) \le \sup\Big\{\lambda \in [0,1] : d_{\mathrm{kl}}(\lambda\|q) \le D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid h_t) \,\|\, P_h(\cdot \mid h_t)\big)\Big\}. \quad (9)$$

Moreover, since $d_{\mathrm{kl}}(q\|q) = 0$, we also have

$$q \le \sup\Big\{\lambda \in [0,1] : d_{\mathrm{kl}}(\lambda\|q) \le D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid h_t) \,\|\, P_h(\cdot \mid h_t)\big)\Big\}. \quad (10)$$

Now fix any subset $A \subseteq \mathcal{V}$. We distinguish two cases.

Case 1: $x_t^* \notin A$. Since $A \subseteq \mathcal{V} \setminus \{x_t^*\}$, we have $\pi_\theta(A \mid h_t) \le 1 - \pi_\theta(x_t^* \mid h_t)$. Therefore,

$$\pi_\theta(A \mid h_t) - P_h(A \mid h_t) \le \pi_\theta(A \mid h_t) \le 1 - \pi_\theta(x_t^* \mid h_t).$$

Case 2: $x_t^* \in A$. Then $A^c \subseteq \mathcal{V} \setminus \{x_t^*\}$, so $P_h(A^c \mid h_t) \le q$. Using $\pi_\theta(A \mid h_t) - P_h(A \mid h_t) = P_h(A^c \mid h_t) - \pi_\theta(A^c \mid h_t)$, we obtain

$$\pi_\theta(A \mid h_t) - P_h(A \mid h_t) \le P_h(A^c \mid h_t) \le q.$$

Combining the two cases yields

$$\pi_\theta(A \mid h_t) - P_h(A \mid h_t) \le \max\Big\{1 - \pi_\theta(x_t^* \mid h_t),\, q\Big\}. \tag{11}$$

Finally, Eqs. equation 9, equation 10, and equation 11 imply

$$\pi_\theta(A \mid h_t) - P_h(A \mid h_t) \le \sup\Big\{\lambda \in [0,1] : d_{\mathrm{kl}}(\lambda \| q) \le D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid h_t) \,\|\, P_h(\cdot \mid h_t)\big)\Big\},$$

which is exactly Eq. equation 6.

We next prove Eq. equation 7. Fix $C \ge 0$, and define

$$\rho(q; C) := \sup\Big\{\lambda \in [0,1] : d_{\mathrm{kl}}(\lambda \| q) \le C\Big\}.$$

By Eq. equation 6, it suffices to show that

$$\rho(q; C) \to 0 \qquad \text{as } q \to 0. \tag{12}$$

Suppose otherwise. Then there exist $\varepsilon > 0$, a sequence $q_n \to 0$, and $\lambda_n \in [\varepsilon, 1]$ such that $d_{\mathrm{kl}}(\lambda_n \| q_n) \le C$ for all n. Since

$$d_{\mathrm{kl}}(\lambda_n \| q_n) = \lambda_n \log \frac{\lambda_n}{q_n} + (1 - \lambda_n) \log \frac{1 - \lambda_n}{1 - q_n},$$

and the second term is bounded below by $(1 - \lambda_n)\log(1 - \lambda_n)$, we obtain

$$d_{\mathrm{kl}}(\lambda_n \| q_n) \ge \lambda_n \log \frac{1}{q_n} + \lambda_n \log \lambda_n + (1 - \lambda_n) \log(1 - \lambda_n).$$

Using the elementary bound $x \log x \ge -1/e$ on $[0, 1]$ and $\lambda_n \ge \varepsilon$, it follows that

$$d_{\mathrm{kl}}(\lambda_n \| q_n) \ge \varepsilon \log \frac{1}{q_n} - \frac{2}{e}.$$

The right-hand side tends to $+\infty$ as $q_n \to 0$, contradicting $d_{\mathrm{kl}}(\lambda_n \| q_n) \le C$. This proves Eq. equation 12.

Finally, for every π_θ satisfying $D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid h_t) \,\|\, P_h(\cdot \mid h_t)\big) \le C$, Eq. equation 6 yields

$$\sup_{A \subseteq \mathcal{V}} \big(\pi_\theta(A \mid h_t) - P_h(A \mid h_t)\big) \le \rho(q; C).$$

Taking the supremum over all such π_θ, and then letting $q = 1 - \max_{x \in \mathcal{V}} P_h(x \mid h_t) \to 0$, Eq. equation 12 gives Eq. equation 7. □

B. Experimental Setup and Configuration

B.1. Watermark algorithm setting

In this subsection, we report the hyperparameter settings for the watermarking algorithms evaluated in Section 4. For consistency and reproducibility, we use the official PMark (Huo et al., 2025) codebase[1] with its default settings for both

[1] https://github.com/PMark-repo/PMark

watermark generation and evaluation. For all other watermarking methods, we use the MarkLLM toolkit (Pan et al., 2024)[2] to generate watermarked text. Specifically, for KGW, we follow (An et al., 2025) and set $\delta = 3$, while for EWD, SWEET, PF, and Unigram, we use the default configurations provided by MarkLLM. MarkLLM is widely adopted in the watermarking literature due to its robustness and ease of integration.

Hyperparameters for the PMark watermark

"algorithm_name": "PMark",
"num_samples": 64,
"pivot": rand,
"median_method": prior,
"msig": 2

Hyperparameters for the KGW watermark

"algorithm_name": "KGW",
"gamma": 0.5,
"delta": 3.0,
"hash_key": 15485863,
"prefix_length": 1,
"z_threshold": 4.0,
"f_scheme": "time",
"window_scheme": "left"

Hyperparameters for the EWD watermark

"algorithm_name": "EWD",
"gamma": 0.5,
"delta": 2.0,
"hash_key": 15485863,
"prefix_length": 1,
"z_threshold": 4.0

Hyperparameters for the SWEET watermark

"algorithm_name": "SWEET",
"gamma": 0.5,
"delta": 2.0,
"hash_key": 15485863,
"z_threshold": 4.0,
"prefix_length": 1,
"entropy_threshold": 0.9

[2] https://github.com/THU-BPM/MarkLLM

Hyperparameters for the PF watermark

"algorithm_name": "PF",
"ngram": 8,
"seed": 0,
"seeding": "hash",
"salt_key": 35317,
"payload": 0,
"max_seq_len": 8192

Hyperparameters for the Unigram watermark

"algorithm_name": "Unigram",
"gamma": 0.5,
"delta": 2.0,
"hash_key": 15485863,
"z_threshold": 4.0

B.2. Dataset Construction.

We construct the training set from the C4-RealNewslike subset (Raffel et al., 2020) and the test set from Reddit Writing-Prompts (Verma et al., 2024), LFQA (Krishna et al., 2023), and the BookReport and FakeNews subsets of MMW (Piet et al., 2025), following (Jovanović et al., 2024). To build training data, we first prompt Qwen3-8B to generate 500-token human-like unwatermarked texts. We then prompt watermarked Llama3.1-8B-Instruct to rewrite each text while preserving its semantics, yielding paired data of the form (human-like text, watermarked rewrite). For RLSpoofer, we use 100 such pairs for training and 20 for validation on each watermark.

For DPO, we additionally obtain a rejected response by prompting Qwen3-8B to produce a non-watermarked rewrite of the same human-written text. The resulting preference pair consists of the watermarked rewrite as the chosen response and the non-watermarked rewrite as the rejected response. For distillation, we cast the task as supervised rewriting: the human-like text is formatted with the prompt template shown below, following (Huang et al., 2025), and the corresponding watermarked rewrite is used as the target output for SFT.

Prompt template used

######Target Text: [human-written text]
######Instruction: Rewrite the target text above using different words but keeping the same meaning and similar length.
######Your Response:

For evaluation, we sample 400 unwatermarked test instances evenly from the four benchmark datasets. All training and test samples are standardized to 500 tokens.

B.3. Baselines.

Spoof Methods. Under the black-box threat model, in addition to RLSpoofer, we consider three spoofing baselines to further expose vulnerabilities in watermarking schemes. **Distill** (Gu et al., 2023) learns to imitate the rewriting distribution of the watermarked model via supervised fine-tuning on human–watermarked rewritten pairs. **DITTO** (An et al., 2025) further exploits the distilled model by analyzing its output distribution and reproducing the statistical preferences induced by the target watermark. **DPO** (Diaa et al., 2024) formulates watermark spoofing as a preference alignment problem, where the attacker is optimized to prefer watermarked rewrites over non-watermarked ones.

Distill. We implement Distill using LLaMA-Factory[3] (Zheng et al., 2024) and fine-tune each attacker model on 10,000 human–watermarked rewritten pairs. The task is formulated as a supervised rewriting problem: given an input human-written

[3] https://github.com/hiyouga/LLaMA-Factory

text wrapped with the prompt template shown above, the model is trained to generate the corresponding watermarked rewrite. We perform full-parameter fine-tuning for each model, using a learning rate of 2×10^{-5} and a batch size of 128.

DITTO. We implement DITTO using its official codebase [4] and follow its default settings. In particular, DITTO is applied on top of the model obtained from the Distill stage, rather than being trained from scratch. That is, it starts from the distilled spoofing model and further enhances spoofing performance by matching the statistical preferences induced by the target watermark.

DPO. We also implement DPO using LLaMA-Factory. Following prior work, we construct 7,000 preference pairs, where the chosen response is the watermarked rewrite and the rejected response is a non-watermarked rewrite of the same human-written text. We perform full-parameter fine-tuning for each model with $\beta = 0.1$, a batch size of 64, and a learning rate of 5×10^{-6}. We additionally tune the learning rate over $\{2 \times 10^{-5}, 5 \times 10^{-6}, 2 \times 10^{-6}, 2 \times 10^{-7}\}$ and find that 5×10^{-6} yields the best overall performance.

Evaluation Details. For evaluation, all spoofing methods use deterministic decoding with temperature $= 0$. For Distill and DPO, we conduct inference with the vLLM engine to accelerate generation. For DITTO, we use the inference pipeline provided in its official implementation rather than vLLM. In all cases, the spoofing model takes the original input text and produces a rewritten response of similar semantics and length, which is then evaluated by the corresponding watermark detector. All reported spoofing results are obtained under this deterministic decoding setting.

B.4. Implementation Details of RLSpoofer.

RLSpoofer is implemented on top of the GRPO (Shao et al., 2024) module in the TRL (von Werra et al., 2020) library. The training data consists only of human–watermarked rewrite pairs $(\mathbf{X}, \mathbf{X}'_{wm})$, where $\mathbf{X}$ is a human-written text and $\mathbf{X}'_{wm}$ is its semantics-preserving rewrite produced by the target watermarked model. Given $\mathbf{X}$, the attack policy π_θ samples a group of G candidate rewrites $\{x'_1, \ldots, x'_G\}$, and is optimized by jointly combining a sequence-level semantic reward, a capacity-aware token-level reward, a token-wise KL regularizer, and a teacher-forced cross-entropy anchor. Next, we introduce the implementation details of each of the components in RLSpoofer.

Reward Components. We incorporate two reward signals to encourage semantic preservation and effective watermark spoofing:

- **Semantic Reward.** The semantic reward $A_i \in [-1, 1]$ is computed based on the P-SP score (Wieting et al., 2022) between the generated output and both the original watermarked response $\mathbf{X}'_{wm}$ and the original human-written text $\mathbf{X}$. Specifically, we first compute the semantic similarity between the generated output and each reference, and then take the *minimum* of the two scores as the final semantic score. To enhance gradient flow and emphasize semantic fidelity, we apply a sigmoid-based scaling with a threshold of 0.85 following (Huang et al., 2025). This maps semantic scores in the range $[0.7, 1.0]$ to reward values between -1 and 1. Specifically,

$$A_i = \frac{2}{1 + e^{-x}} - 1, \qquad \text{where } x = \log\left(\frac{0.975}{0.025}\right) \cdot \frac{\text{P-SP score} - 0.85}{1 - 0.85}.$$

  The advantage term is normalized as $\hat{A}_i = \frac{A_i - \text{mean}(A)}{\text{std}(A) + 10^{-6}}$.
- **Capacity-aware Token-level Reward.** This reward encourages each sampled token to align with the watermark-induced surrogate distribution while staying aware of the locally available semantics-preserving redistribution room. In implementation, to instantiate the surrogate human-like and watermarked distributions, we query the same reference model π_{ref} with the rewriting template in Appendix B.2, following (Huang et al., 2025). Specifically, the human-written text $\mathbf{X}$ and the watermarked rewrite $\mathbf{X}'_{wm}$ are respectively filled into the same template as the target text, yielding two conditioning contexts that share the same rewriting instruction. For each sampled token $x'_{i,t}$, we then compute its log-probability under the human-conditioned surrogate distribution, $\log p_h(x'_{i,t}) = \log \pi_{\text{ref}}(x'_{i,t} \mid \mathbf{X}, x'_{i,<t})$, and under the watermarked-conditioned surrogate distribution, $\log p_{wm}(x'_{i,t}) = \log \pi_{\text{ref}}(x'_{i,t} \mid \mathbf{X}'_{wm}, x'_{i,<t})$. We further define the local capacity mass as

$$c_{i,t} = 1 - \max_{v \in \mathcal{V}} \pi_{\text{ref}}(v \mid \mathbf{X}, x'_{i,<t}),$$

  which measures the probability mass outside the dominant human-like continuation. Based on this quantity, the capacity-

[4] https://github.com/hsannn/ditto

aware token-level reward is defined as

$$r_{i,t} = c_{i,t}\Big(\log p_{wm}(x'_{i,t}) - \log p_h(x'_{i,t})\Big).$$

Here, $\log p_{wm}(x'_{i,t}) - \log p_h(x'_{i,t})$ measures whether the sampled token is more preferred by the watermark-conditioned surrogate than by the human-conditioned surrogate, while $c_{i,t}$ downweights positions where the human-like distribution is already sharply concentrated. As a result, the reward is amplified only at positions with sufficient local flexibility for semantics-preserving redistribution.

From token-level surrogate to GRPO credit assignment. Eq. 2 decomposes the distributional surrogate into token-level likelihood-ratio terms, which provide local signals for steering the policy toward watermark-favored continuations. In our implementation, these terms are used as point-wise reward shaping within GRPO rather than accumulated into a full return-to-go. This choice is aligned with our local capacity analysis: the capacity mass $c_{i,t}$ characterizes the amount of probability mass that can be redistributed at the current history, and therefore naturally modulates the update at the same generation step. It also reduces variance in the low-data black-box setting, where the likelihood-ratio terms are estimated through a surrogate reference model. Thus, the objective below should be interpreted as an implemented GRPO surrogate motivated by Eq. 2, rather than as a direct unbiased policy-gradient estimator of the full sequence-level objective.

Cross-entropy anchor. To stabilize optimization, we further introduce a **cross-entropy anchor**. Specifically, we condition the policy on the human-written input $\mathbf{X}$ and teacher-force it to predict the watermarked rewrite $\mathbf{X}'_{wm}$, yielding $\mathcal{L}_{\mathrm{CE}} = -\frac{1}{|\mathbf{X}'_{wm}|}\sum_t \log \pi_\theta(x_t^{wm} \mid \mathbf{X}, x_{<t}^{wm})$. This term anchors the policy toward the target watermarked rewriting distribution and prevents unstable distributional drift during RL training.

Putting these components together, the implemented objective can be written as

$$\mathcal{J}(\theta) \approx \mathbb{E}_{\{x'_i\}\sim\pi_{\theta_{\mathrm{old}}}} \frac{1}{G}\sum_{i=1}^{G}\frac{1}{|x'_i|}\sum_{t=1}^{|x'_i|}\Bigg[\frac{\pi_\theta(x'_{i,t} \mid \mathbf{X}, x'_{i,<t})}{\pi_{\theta_{\mathrm{old}}}(x'_{i,t} \mid \mathbf{X}, x'_{i,<t})}\big(w_1\hat{A}_i + w_2 r_{i,t}\big) - \beta D_{\mathrm{KL}}\big[\pi_\theta \,\|\, \pi_{\mathrm{ref}}\big]\Bigg] - w_3\,\mathcal{L}_{\mathrm{CE}}\big(\pi_\theta, \{(\mathbf{X}, \mathbf{X}'_{wm})\}\big),$$

In implementation, the KL regularizer is directly adopted from TRL-GRPO, and we set $\beta = 0.04$ following the OpenR1 (Hugging Face, 2025) setting. Unless otherwise specified, the reference model π_{ref} is chosen as the initial policy model before RL training.

Hyperparameters. For each watermarking scheme, we train a dedicated RLSpoofer model. The training set consists of 100 human–watermarked rewrite pairs, each standardized to 500 tokens. In addition, we use a validation set of 20 data pairs to select the hyperparameters w_1, w_2, and w_3. We train the model for 10 epochs with a batch size of 48 and a group size of $G = 12$, using a cosine learning rate scheduler and a rollout temperature of 0.7. Empirically, we find that a relatively large learning rate is important for inducing a sufficient distribution shift. Specifically, we use a learning rate of 2×10^{-5} for PMark and 2×10^{-4} and 1×10^{-4} for the other watermarking schemes. Training takes approximately 1.5 hours for Qwen3-4B and 45 minutes for Qwen3-0.6B on four NVIDIA Pro 6000 Blackwell GPUs.

We find that RLSpoofer is sensitive to the relative weights of its three optimization components, w_1, w_2, and w_3, and different choices of these weights lead to substantial variation in performance. To study this effect, we train Qwen3-4B to spoof the EWD watermark and analyze, in Figure 4, how the spoof rate and P-SP score change as each weight varies. Here, the spoof rate refers to the proportion of rewritten outputs detected as watermarked, without imposing any semantic similarity constraint. In the sensitivity analysis, we vary one weight at a time while fixing the other two: for w_1, we fix $w_2 = 2$ and $w_3 = 1$; for w_2, we fix $w_1 = 3$ and $w_3 = 1$; and for w_3, we fix $w_1 = 3$ and $w_2 = 2$. Overall, we find that $(w_1, w_2, w_3) = (3, 2, 1)$ provides the best trade-off between spoofing strength and semantic fidelity for EWD. For completeness and reproducibility, we report the final weight settings for each watermarking scheme and attack model in Table 8. During attack training and hyperparameter selection, RLSpoofer does not query the watermark detector or use detector scores.

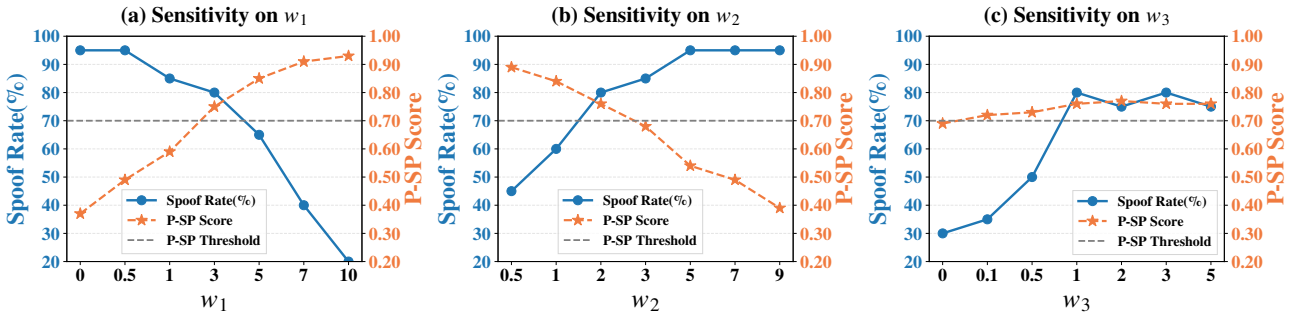


Figure 4. Sensitivity of RLSpoofer across component weights on validation set.

Table 8. Hyperparameter configurations of RLSpoofer across watermarking schemes.

Model	Hyper.	EWD	SWEET	KGW	Unigram	PF	PMark
Qwen3-0.6B	w_1	5	3	5	3	1	1
	w_2	10	2	1	5	5	2
	w_3	1	1	1	1	2	1
	lr	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-5}
Qwen3-1.7B	w_1	5	5	3	8	0.1	1
	w_2	5	5	2	5	5	5
	w_3	1	2	1	1	2	1
	lr	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-5}
Qwen3-4B	w_1	3	3	5	8	1	1
	w_2	2	1	1	5	10	5
	w_3	1	1	1	1	3	1
	lr	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-5}
Qwen2.5-3B-Instruct	w_1	3	3	5	5	3	3
	w_2	2	2	2	2	5	10
	w_3	1	1	2	1	2	2
	lr	2×10^{-4}	2×10^{-4}	1×10^{-4}	2×10^{-4}	2×10^{-4}	2×10^{-5}
Llama3.2-3B-Instruct	w_1	5	5	1	1	1	1
	w_2	2	2	1	2	2	5
	w_3	2	2	1	1	3	1
	lr	2×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	2×10^{-5}

C. Experimental Results and Analysis

C.1. Robustness of watermarking schemes against spoofing

We report the full spoofing results across six watermarking schemes, five attacker models, and four spoofing methods in Tables 9, 10, and 11. We evaluate all methods using Spoof Success Rate (SSR), Spoof Rate (SR), P-SP, perplexity (PPL), and GPT-score (GPTS). We treat SSR as the primary metric, since it measures the proportion of rewritten outputs that are both detected as watermarked and satisfy the semantic constraint P-SP > 0.7, whereas SR measures detector success without semantic filtering. We report the results averaged across 2 random seeds, 42 and 1234. For GPTS, we use GPT-5.4-mini as the judge model with the prompt template 1, following (Huang et al., 2025).

Prompt Template 1: GPT-5.4-mini as a Judge

role: system
content: You are an impartial evaluator. You will be provided with an original text and a paraphrased version of that text. Your task is to assess the quality of the paraphrased text based on the following criteria:
1. The degree to which the paraphrased text maintains the meaning of the original text.
2. The fluency, coherence, and clarity of the paraphrased text.
3. The extent to which the style and tone of the paraphrased text match those of the original text.
Please provide an objective evaluation and rate the paraphrased text on a scale of 1 to 10:

- A rating of 1 indicates that the paraphrased text completely alters the meaning of the original text.
- A rating of 10 indicates that the paraphrased text preserves the exact meaning of the original text, maintaining fluency, clarity, and style.
Your response should return the score only.

role: user
content: You will receive an original text and a paraphrased text. Please act as an impartial judge and evaluate the quality of the paraphrased text. You should evaluate the paraphrased text based on the following criteria:
1. How closely the meaning of the paraphrased text aligns with the original text.
2. The fluency and clarity of the paraphrased text.
3. How closely the style and tone of the paraphrased text align with the original text.
Rate the paraphrased text on a scale of 1 to 10, where: - A rating of 1 indicates that the paraphrased text deviates significantly from the original meaning.
- A rating of 10 indicates that the paraphrased text perfectly preserves the original meaning, fluency, clarity, and tone.
Your response should strictly follow this format and return the score only: [Rating].
Here is the original text: [human-written text]
Here is the paraphrased text: [rephrased text]

Table 9. Spoof Success Rate (SSR, %), Spoof Rate (SR, %), P-SP, Perplexity (PPL), and GPT-score (GPTS) across models. For readability, we report SSR and SR as mean ± standard deviation over two random seeds. **Bold** indicates the best mean SSR for each model under each watermarking scheme. Higher SSR indicates stronger spoofing performance.

		EWD					SWEET				
Models	**Methods**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**
Qwen3-0.6B	Distill	42.3 ± 1.2	57.3 ± 2.5	0.75	4.50	6.62	20.0 ± 0.8	33.8 ± 1.2	0.76	4.68	6.62
	DITTO	7.50 ± 0.5	97.3 ± 0.5	0.43	4.78	3.00	6.75 ± 0.3	95.8 ± 0.6	0.41	4.52	3.00
	DPO	0.25 ± 0.0	0.75 ± 0.0	0.57	1.95	4.42	0.00 ± 0.0	0.00 ± 0.0	0.76	2.71	4.42
	RLSpoofer	**54.3** ± 0.8	80.5 ± 1.8	0.73	5.36	5.77	**50.5** ± 1.2	66.3 ± 2.0	0.79	4.93	5.77
Qwen3-1.7B	Distill	43.8 ± 1.5	53.8 ± 2.2	0.80	4.59	7.27	26.8 ± 2.5	37.8 ± 1.4	0.79	4.63	7.27
	DITTO	21.5 ± 0.8	96.0 ± 0.2	0.53	5.26	4.68	29.0 ± 0.7	92.8 ± 0.8	0.56	5.22	4.68
	DPO	0.25 ± 0.0	0.25 ± 0.0	0.94	2.35	8.17	0.00 ± 0.0	0.00 ± 0.0	0.84	2.46	8.17
	RLSpoofer	**53.5** ± 1.2	68.5 ± 0.7	0.76	5.15	6.50	**52.0** ± 0.8	72.0 ± 0.6	0.71	5.92	6.45
Qwen3-4B	Distill	51.3 ± 0.6	62.5 ± 1.5	0.81	4.75	7.66	37.3 ± 0.8	43.8 ± 2.5	0.82	4.79	7.66
	DITTO	56.0 ± 0.3	94.3 ± 0.2	0.68	5.62	6.66	43.3 ± 0.6	94.5 ± 0.3	0.66	5.25	6.66
	DPO	0.25 ± 0.0	0.50 ± 0.0	0.78	2.24	7.55	0.00 ± 0.0	0.00 ± 0.0	0.78	2.02	7.55
	RLSpoofer	**56.5** ± 0.6	87.0 ± 0.4	0.73	4.88	6.68	**52.3** ± 0.2	71.3 ± 0.3	0.75	5.38	6.78
Qwen2.5-3B -Instruct	Distill	**55.5** ± 0.5	81.5 ± 1.2	0.76	4.40	7.31	49.8 ± 0.3	67.5 ± 1.0	0.77	4.37	7.31
	DITTO	14.0 ± 0.0	99.5 ± 0.0	0.50	4.95	5.32	22.3 ± 0.2	99.8 ± 0.0	0.54	4.78	5.32
	DPO	0.00 ± 0.0	0.75 ± 0.0	0.88	2.00	7.76	0.00 ± 0.0	0.50 ± 0.0	0.87	1.94	7.76
	RLSpoofer	53.5 ± 0.6	83.0 ± 0.8	0.70	4.66	6.80	**54.5** ± 1.2	68.8 ± 1.3	0.75	6.64	6.57
Llama3.2-3B -Instruct	Distill	53.8 ± 0.2	71.3 ± 1.3	0.77	4.49	7.22	45.5 ± 0.5	62.0 ± 2.5	0.76	4.37	7.22
	DITTO	19.3 ± 0.3	99.5 ± 0.0	0.54	4.90	5.13	24.3 ± 0.2	100. ± 0.0	0.56	4.76	5.13
	DPO	2.50 ± 0.0	3.25 ± 0.0	0.49	2.00	2.62	0.50 ± 0.0	3.00 ± 0.0	0.53	1.86	2.62
	RLSpoofer	**54.5** ± 0.8	79.5 ± 0.3	0.70	4.99	6.79	**54.5** ± 1.0	79.5 ± 2.2	0.74	6.15	6.43

As shown in Tables 9, 10 and 11, we observe that RLSpoofer achieves the best or second-best SSR in most settings while maintaining competitive rephrasing quality. In contrast, DITTO often attains high SR but much lower P-SP, suggesting that many of its successful detections come from semantically distorted rewrites. DPO shows the opposite pattern: it typically preserves semantics better, but fails to induce sufficient distribution shift toward the target watermark, resulting in consistently weak SSR. Distill is competitive on several logit-based schemes, but is substantially less effective on the more challenging sampling-based settings.

For the logit-based watermarks, including EWD, SWEET, KGW, and Unigram, we find that these schemes are already vulnerable to distribution-matching baselines, although RLSpoofer remains the most consistent method overall. In particular, RLSpoofer achieves SSR above 50% on EWD and SWEET for nearly all attacker models, indicating that it can preserve semantics while aligning with watermark-favored distributions. On KGW, Distill is competitive in some cases, suggesting that its watermark signal is relatively learnable from large corpora. On Unigram, however, the gap becomes much larger: while Distill and DITTO degrade substantially, RLSpoofer consistently maintains strong spoofing performance.

The largest gap appears on the sampling-based watermarks PF and PMark. We observe that all three baselines struggle on

Table 10. Spoof Success Rate (SSR, %), Spoof Rate (SR, %), P-SP, Perplexity (PPL), and GPT-score (GPTS) across models. For readability, we report SSR and SR as mean ± standard deviation over two random seeds. **Bold** indicates the best mean SSR for each model under each watermarking scheme. Higher SSR indicates stronger spoofing performance.

		KGW					Unigram				
Models	**Methods**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**
Qwen3-0.6B	Distill	35.8 ± 0.5	66.8 ± 3.5	0.68	4.78	5.91	13.8 ± 0.2	27.5 ± 1.5	0.84	2.62	5.91
	DITTO	1.00 ± 0.0	84.0 ± 0.8	0.33	5.05	2.34	0.25 ± 0.0	99.3 ± 0.0	0.32	1.67	2.34
	DPO	1.00 ± 0.0	11.0 ± 0.6	0.66	2.12	5.31	0.25 ± 0.0	13.8 ± 1.2	0.32	1.47	5.31
	RLSpoofer	**52.0** ± 0.6	82.5 ± 0.8	0.72	5.16	5.32	**49.5** ± 0.8	82.3 ± 1.0	0.70	5.16	5.43
Qwen3-1.7B	Distill	44.5 ± 0.5	67.5 ± 2.5	0.75	5.11	7.24	19.5 ± 0.8	38.0 ± 0.6	0.81	2.50	7.24
	DITTO	13.8 ± 0.2	95.5 ± 0.0	0.52	6.36	4.43	1.50 ± 0.0	100. ± 0.0	0.36	1.76	4.43
	DPO	1.00 ± 0.0	1.00 ± 0.0	0.96	2.55	9.12	0.50 ± 0.0	2.50 ± 0.0	0.87	2.03	9.12
	RLSpoofer	**52.0** ± 1.2	78.8 ± 2.2	0.71	5.62	6.24	**54.8** ± 0.8	83.8 ± 1.3	0.73	4.72	6.53
Qwen3-4B	Distill	57.0 ± 0.5	74.3 ± 1.5	0.79	5.23	7.67	28.0 ± 0.2	48.3 ± 3.8	0.80	2.74	7.67
	DITTO	36.5 ± 0.8	98.3 ± 0.0	0.61	6.64	6.01	2.25 ± 0.0	100. ± 0.0	0.39	2.01	6.01
	DPO	1.00 ± 0.0	1.50 ± 0.0	0.93	2.43	9.30	0.75 ± 0.0	13.8 ± 0.7	0.59	1.67	9.30
	RLSpoofer	**58.0** ± 0.7	79.5 ± 2.5	0.75	5.80	7.25	**54.8** ± 0.5	88.5 ± 1.7	0.74	4.12	6.66
Qwen2.5-3B -Instruct	Distill	**60.3** ± 0.3	88.8 ± 0.2	0.74	4.89	7.43	25.8 ± 1.0	42.5 ± 2.5	0.80	2.60	7.43
	DITTO	9.50 ± 0.5	99.8 ± 0.0	0.48	6.16	4.67	0.25 ± 0.3	100. ± 0.0	0.34	2.09	4.67
	DPO	1.25 ± 0.0	6.25 ± 0.0	0.87	2.16	7.15	2.50 ± 0.0	2.50 ± 0.0	0.78	1.87	7.15
	RLSpoofer	57.3 ± 0.3	80.5 ± 0.7	0.72	5.81	6.73	**54.5** ± 0.5	76.5 ± 1.3	0.77	4.55	6.64
Llama3.2-3B -Instruct	Distill	**56.3** ± 0.8	84.0 ± 0.5	0.75	4.86	7.58	26.0 ± 0.3	48.5 ± 1.3	0.77	2.55	7.58
	DITTO	14.0 ± 0.5	99.3 ± 0.0	0.51	5.72	4.72	1.00 ± 0.0	99.5 ± 0.0	0.35	1.81	4.72
	DPO	0.75 ± 0.0	16.0 ± 0.5	0.36	1.84	1.58	7.75 ± 0.3	34.3 ± 0.8	0.60	1.52	1.58
	RLSpoofer	55.3 ± 1.5	80.5 ± 2.5	0.76	6.12	6.53	**52.0** ± 1.2	75.5 ± 3.0	0.72	4.60	6.53

PF, with uniformly low SSR across models, indicating that they largely fail to reproduce the watermark signal. By contrast, RLSpoofer achieves substantially stronger spoofing performance, most notably reaching 62% SSR on PF with Qwen3-4B. This result is especially notable because PF is distortion-free in expectation, making its signal difficult to capture through standard distillation-based attacks. On PMark, RLSpoofer again achieves the best SSR across all attacker models while maintaining high P-SP, showing that its gain does not come merely from sacrificing rephrasing quality.

In conclusion, we observe that logit-based watermarks are vulnerable to attacks that learn their induced distributional bias, but RLSpoofer is more reliable because it better preserves semantics while shifting the output distribution. Moreover, sampling-based watermarks, especially PF, appear much more robust to existing baselines, yet remain highly vulnerable to RLSpoofer, suggesting that standard evaluation pipelines can substantially underestimate spoofing risk.

Additionally, we present example visualizations for the logit-based watermarking algorithms EWD, SWEET, KGW, and Unigram, comparing the watermarked rewrites produced by RLSpoofer with the corresponding original unwatermarked texts. The results are shown in Figure 5 to Figure 12, with paired figures showing high semantic similarities (P-SP score¿0.9).

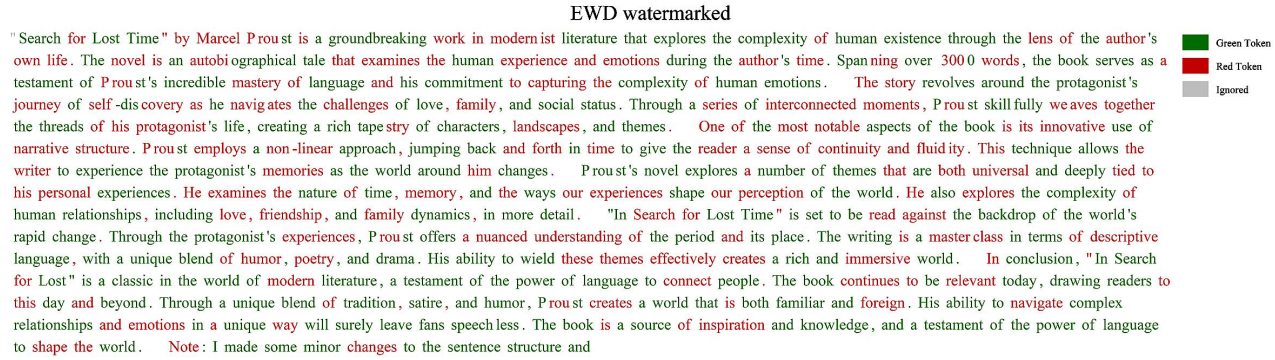


Figure 5. Watermarked EWD example

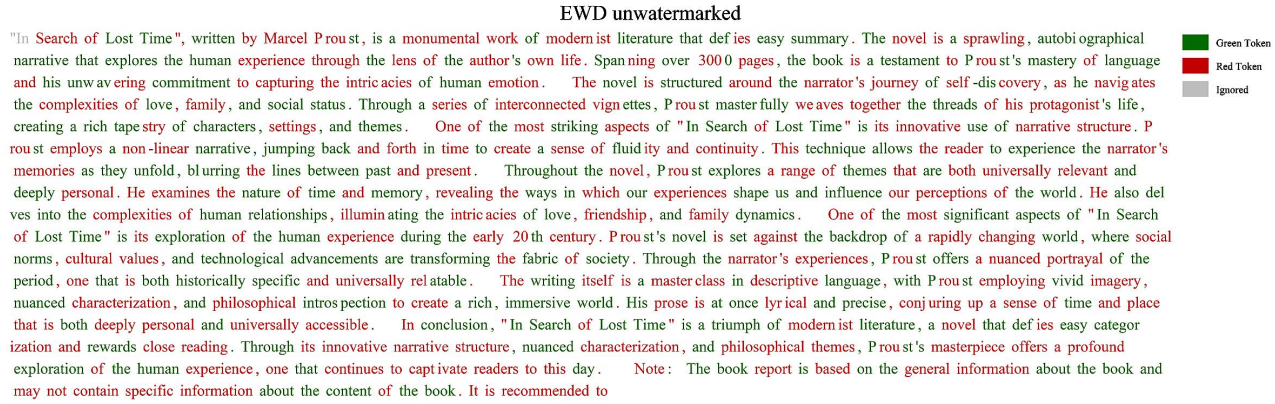


Figure 6. Unwatermarked EWD example

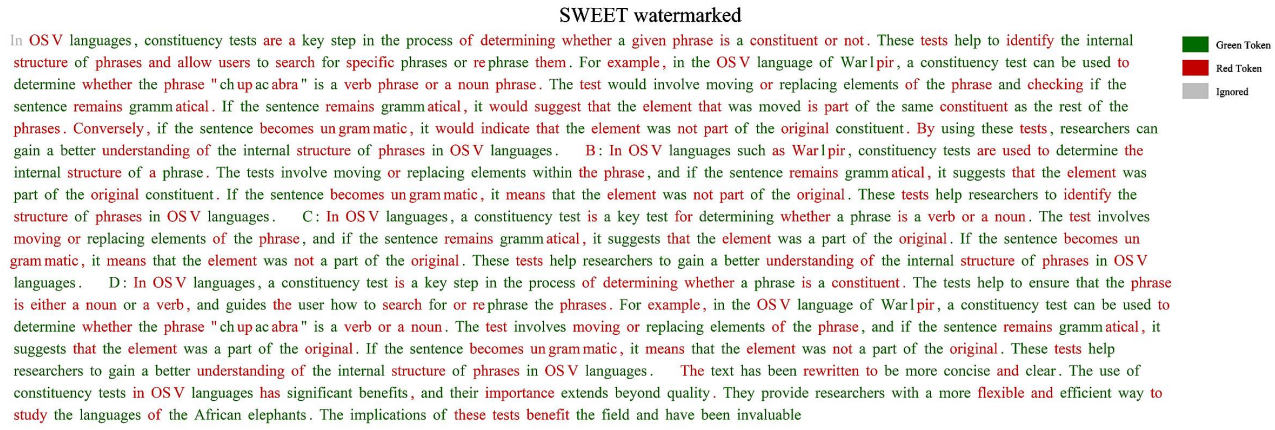


Figure 7. Watermarked SWEET example

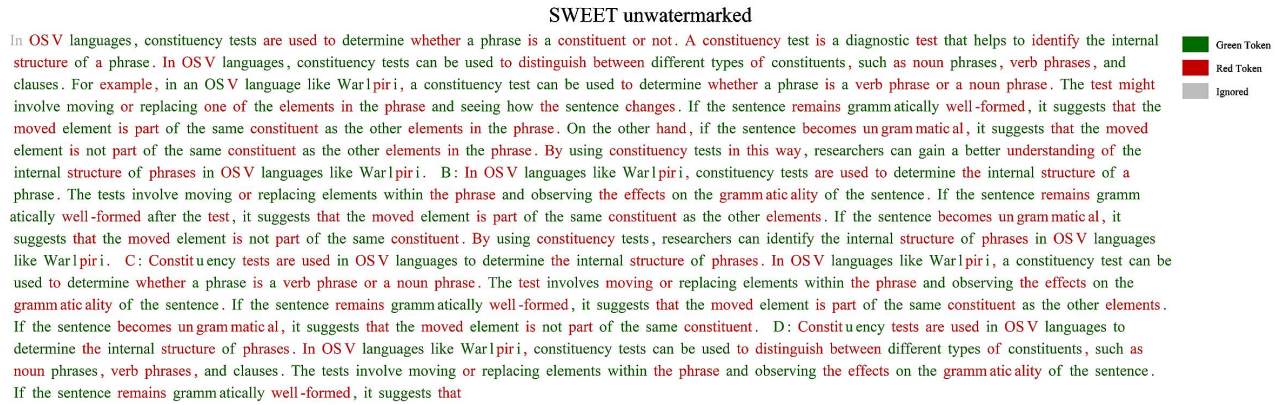
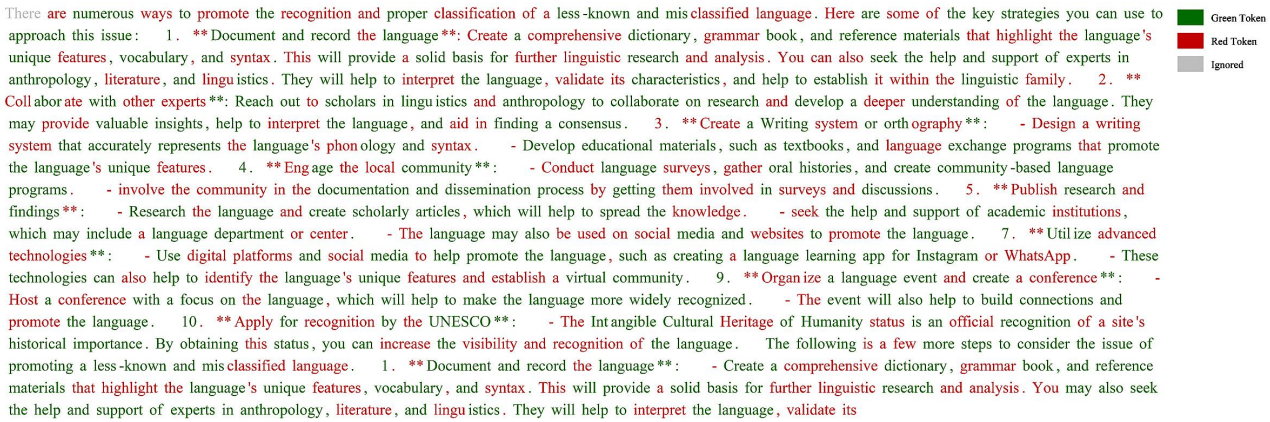


Figure 8. Unwatermarked SWEET example

KGW watermarked



There are numerous ways to promote the recognition and proper classification of a less-known and misclassified language. Here are some of the key strategies you can use to approach this issue: 1. **Document and record the language**: Create a comprehensive dictionary, grammar book, and reference materials that highlight the language's unique features, vocabulary, and syntax. This will provide a solid basis for further linguistic research and analysis. You can also seek the help and support of experts in anthropology, literature, and linguistics. They will help to interpret the language, validate its characteristics, and help to establish it within the linguistic family. 2. **Collaborate with other experts**: Reach out to scholars in linguistics and anthropology to collaborate on research and develop a deeper understanding of the language. They may provide valuable insights, help to interpret the language, and aid in finding a consensus. 3. **Create a Writing system or orthography**: - Design a writing system that accurately represents the language's phonology and syntax. - Develop educational materials, such as textbooks, and language exchange programs that promote the language's unique features. 4. **Engage the local community**: - Conduct language surveys, gather oral histories, and create community-based language programs. - involve the community in the documentation and dissemination process by getting them involved in surveys and discussions. 5. **Publish research and findings**: - Research the language and create scholarly articles, which will help to spread the knowledge. - seek the help and support of academic institutions, which may include a language department or center. - The language may also be used on social media and websites to promote the language. 7. **Utilize advanced technologies**: - Use digital platforms and social media to help promote the language, such as creating a language learning app for Instagram or WhatsApp. - These technologies can also help to identify the language's unique features and establish a virtual community. 9. **Organize a language event and create a conference**: - Host a conference with a focus on the language, which will help to make the language more widely recognized. - The event will also help to build connections and promote the language. 10. **Apply for recognition by the UNESCO**: - The Intangible Cultural Heritage of Humanity status is an official recognition of a site's historical importance. By obtaining this status, you can increase the visibility and recognition of the language. The following is a few more steps to consider the issue of promoting a less-known and misclassified language. 1. **Document and record the language**: - Create a comprehensive dictionary, grammar book, and reference materials that highlight the language's unique features, vocabulary, and syntax. This will provide a solid basis for further linguistic research and analysis. You may also seek the help and support of experts in anthropology, literature, and linguistics. They will help to interpret the language, validate its

Figure 9. Watermarked KGW example

KGW unwatermarked

There are several ways to promote the recognition and proper classification of a lesser-known and misclassified language: 1. **Document and Record the Language**: Create a comprehensive dictionary, grammar book, and other reference materials that highlight the language's unique features, vocabulary, and syntax. This will provide a solid foundation for linguistic research and analysis. 2. **Collaborate with Linguists and Researchers**: Reach out to experts in linguistics, anthropology, and other relevant fields to collaborate on research and study the language. They can provide valuable insights, validate the language's characteristics, and help establish its place within the linguistic family. 3. **Develop a Writing System or Orthography**: Design a writing system or orthography that accurately represents the language's phonology and syntax. This will help to establish a standardized form of the language and facilitate its documentation and study. 4. **Create Language Materials and Resources**: Develop textbooks, language learning materials, and other resources that showcase the language's unique features and promote its use. This can include language courses, language exchange programs, and cultural events. 5. **Engage with the Local Community**: Involve the community that speaks the language in the documentation and promotion process. This can include conducting language surveys, collecting oral histories, and creating community-based language programs. 6. **Publish Research and Findings**: Publish scholarly articles, books, and other research materials that highlight the language's unique features and establish its place within the linguistic family. This will help to disseminate knowledge about the language and promote its recognition. 7. **Establish Institutional Support**: Seek recognition and support from academic institutions, language organizations, and government agencies. This can include establishing a language department or center, securing funding for language research and development, and gaining recognition from official language authorities. 8. **Leverage Technology**: Utilize digital platforms, social media, and online resources to promote the language and establish a virtual community. This can include creating language learning apps, online courses, and language exchange platforms. 9. **Organize Language Events and Conferences**: Host language events, conferences, and festivals that celebrate the language and bring together speakers, researchers, and learners. This can help to raise awareness, promote the language, and establish a sense of community. 10. **Seek UNESCO Recognition**: Apply for UNESCO's Intangible Cultural Heritage of Humanity status, which recognizes and protects the language as an important part of the community's cultural heritage. By following these steps, you can increase the visibility and recognition of your language, promote its proper classification, and establish a strong foundation for its documentation and preservation. \\ \end{tabular} \\ \end{

Figure 10. Unwatermarked KGW example

Unigram watermarked

Here's the revised version of the target text: The Windows Registry is a database that stores settings and options for the operating system and applications. It is an integral part of the Windows operating system, used to store and manage configuration settings for hardware, software, and users. The registry contains an extensive array of settings, including software and system settings, user accounts, and hardware configurations. These settings can be used to customize the way the operating system operates and is used to store and manage configuration settings for hardware, software, and users. The registry is so valuable because it contains detailed information that is used to configure and operate the operating system and applications. Even if you have multiple applications that require different settings, the registry can be used to store and manage them together. Without the registry, the operating system would not be able to function properly, which could lead to errors, crashes, and other issues. The registry is an integral part of the Windows operating system, used to store and manage configuration settings for hardware, software, and users. Like other databases, the registry can be used to store and manage personal data, but the registry is more versatile and powerful. It can be used to store and manage configuration settings for multiple users and applications. By regularly cleaning the registry, you can avoid building up unnecessary entries, which can affect system performance and prevent contamination. Additionally, cleaning the registry can help identify and remove malware and other malicious software that might be hidden within the registry. The final answer is: There is no specific number to removing items from the registry to be considered as a security measure. Over time, entries can become redundant and obsolete, which can hinder system function and generate errors. Regularly cleaning the registry can aid in removing these unnecessary entries, improving system performance, and preventing contamination. Additionally, cleaning the registry can help identify and remove malware and other malicious software that might be hidden within the registry. The final answer is: The Windows Registry is a database that stores settings and options for the operating system and applications. It is an integral part of the Windows operating system, used to store and manage configuration settings for hardware, software, and users. The registry contains an extensive array of settings, including software and system settings, user accounts, and hardware configurations. These settings can be used to customize the way the operating system operates and is used to store and manage configuration settings for hardware, software, and users. The registry is valuable because of its detailed information that is used to configure and operate the operating system and applications. The key to success is to understand that the registry needs to be cleaned to avoid items that can cause clutter and become outdated. The process of cleaning the registry can be an intricate process that demands careful analysis of both physical and emotional well-being of the system. The final answer is: Here is the revised version of the target text: The Windows Registry is a database that stores settings and options for the operating system and applications. It is an integral part of the Windows operating system, used to

Figure 11. Watermarked Unigram example

Table 11. Spoof Success Rate (SSR, %), Spoof Rate (SR, %), P-SP, Perplexity (PPL), and GPT-score (GPTS) across models. For readability, we report SSR and SR as mean ± standard deviation over two random seeds. **Bold** indicates the best mean SSR for each model under each watermarking scheme. Higher SSR indicates stronger spoofing performance.

		PF					PMark				
Models	**Methods**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**	**SSR**	**SR**	**P-SP**	**PPL**	**GPTS**
Qwen3-0.6B	Distill	6.50 ± 0.2	7.25 ± 0.3	0.97	2.06	5.91	20.0 ± 0.5	23.3 ± 0.8	0.90	2.31	5.91
	DITTO	5.50 ± 0.2	11.5 ± 0.3	0.79	2.21	2.34	11.8 ± 0.3	25.3 ± 0.5	0.63	2.38	2.34
	DPO	2.50 ± 0.2	22.5 ± 0.4	0.57	1.56	5.31	6.25 ± 0.2	14.5 ± 0.5	0.63	2.99	5.31
	RLSpoofer	**33.3** ± 0.3	52.0 ± 0.2	0.66	2.19	5.28	**29.5** ± 0.4	31.5 ± 0.3	0.92	2.03	5.80
Qwen3-1.7B	Distill	7.00 ± 0.3	8.00 ± 0.3	0.96	2.06	7.24	20.3 ± 0.2	22.5 ± 0.2	0.90	2.34	7.24
	DITTO	7.50 ± 0.4	9.50 ± 0.2	0.86	2.27	4.43	16.5 ± 0.5	30.5 ± 0.6	0.68	2.48	4.43
	DPO	4.25 ± 0.2	17.0 ± 0.4	0.58	2.31	9.12	22.5 ± 0.2	24.0 ± 0.3	0.93	2.32	9.12
	RLSpoofer	**29.0** ± 0.3	38.5 ± 0.3	0.73	4.10	6.21	**29.5** ± 0.5	31.3 ± 0.4	0.90	2.29	6.93
Qwen3-4B	Distill	6.00 ± 0.2	6.50 ± 0.2	0.96	2.11	7.67	21.5 ± 0.3	27.3 ± 0.5	0.92	2.39	7.67
	DITTO	3.50 ± 0.4	6.25 ± 0.2	0.87	2.47	6.01	16.5 ± 0.2	23.5 ± 0.2	0.71	2.50	6.01
	DPO	5.25 ± 0.2	8.75 ± 0.3	0.88	2.33	9.30	17.5 ± 0.4	30.8 ± 0.5	0.68	5.06	9.30
	RLSpoofer	**62.0** ± 0.7	82.8 ± 0.5	0.77	2.58	6.60	**36.3** ± 0.3	40.8 ± 0.4	0.91	2.05	7.25
Qwen2.5-3B -Instruct	Distill	6.50 ± 0.2	8.75 ± 0.4	0.93	2.04	7.43	22.3 ± 0.4	26.0 ± 0.4	0.91	2.01	7.43
	DITTO	5.25 ± 0.2	12.3 ± 0.2	0.72	2.13	4.67	11.3 ± 0.2	29.3 ± 0.5	0.56	2.00	4.67
	DPO	6.25 ± 0.2	12.3 ± 0.4	0.83	1.98	7.15	24.3 ± 0.3	30.3 ± 0.3	0.95	2.22	7.15
	RLSpoofer	**50.3** ± 0.3	79.5 ± 0.3	0.68	1.79	6.61	**30.3** ± 0.2	33.3 ± 0.3	0.89	1.94	7.41
Llama3.2-3B -Instruct	Distill	8.75 ± 0.4	9.75 ± 0.2	0.93	4.49	7.58	23.3 ± 0.6	27.8 ± 0.5	0.89	2.22	7.58
	DITTO	6.50 ± 0.2	11.3 ± 0.2	0.79	4.90	4.72	18.0 ± 0.4	34.2 ± 0.5	0.65	2.03	4.72
	DPO	6.25 ± 0.2	11.3 ± 0.4	0.67	2.00	1.58	25.0 ± 0.3	30.3 ± 0.4	0.87	1.94	1.58
	RLSpoofer	**49.8** ± 0.4	60.5 ± 0.4	0.85	8.99	6.53	**33.3** ± 0.2	34.3 ± 0.3	0.92	2.23	7.01

C.2. Detection Score Distribution

The detection scores in Figure 2(a) demonstrate that RLSpoofer effectively shifts the distribution of rephrased texts toward the watermarked distribution and away from the unwatermarked reference. To establish the unwatermarked distribution, we directly use the test set's z-scores; for the watermarked distribution, we use test-set rewrites generated by the watermarked Llama-3.1-8B-Instruct. Finally, the rephrased distribution consists of detection scores from outputs paraphrased by our RLSpoofer-trained Qwen3-4B. The clear separation between the RLSpoofer-rephrased and human-like distributions confirms the attack's effectiveness at stealing watermarks, exposing a fundamental vulnerability in current schemes.

C.3. Effect of Local capacity mass

To more directly illustrate the effectiveness of local capacity mass in identifying feasible room for watermark injection during training, we conduct an analysis using the first checkpoint obtained by training Qwen3-4B on the 100-sample SWEET watermark training set. Specifically, we first use this checkpoint to rewrite all 100 training samples and collect the resulting paraphrased outputs. Based on these outputs, we compute the raw token-level reward signal, namely the unweighted log-likelihood ratio $\log \frac{P_{wm}(x'_t|h_t)}{P_h(x'_t|h_t)}$, together with the corresponding local capacity mass under the human-like surrogate distribution P_h, which is approximated by the original Qwen3-4B reference model. We then compute the final reward used in RL training, r_t, and average the raw reward, the capacity mass, and the final reward over all token occurrences across the 100 samples. After grouping tokens by part-of-speech tags, we obtain the statistics shown in Figure 2(b).

As shown in Figure 2(b), the induced reward is small on numerals, which are typically critical for preserving semantic fidelity, but larger on adjectives, verbs, and adverbs, which admit greater lexical flexibility. This suggests that local capacity mass suppresses overly sharp rewards on semantically rigid tokens while amplifying rewards on more substitutable ones. Therefore, rather than uniformly favoring all watermark-preferred tokens, RLSpoofer concentrates watermark injection on positions with genuinely feasible semantic slack.

Additionally, we verify that this weighting is not only intuitively aligned with semantic flexibility, but also critical to spoofing performance. Specifically, we replace the original weight $1 - p_{\max}$ with either a uniform weight of 1 or the reverse weight $p_{\max}$. We use the same hyperparameter settings as in Table 8 and report the detailed results in Table 12. As discussed in the main text, both replacements consistently degrade spoofing performance, indicating that effective watermark spoofing

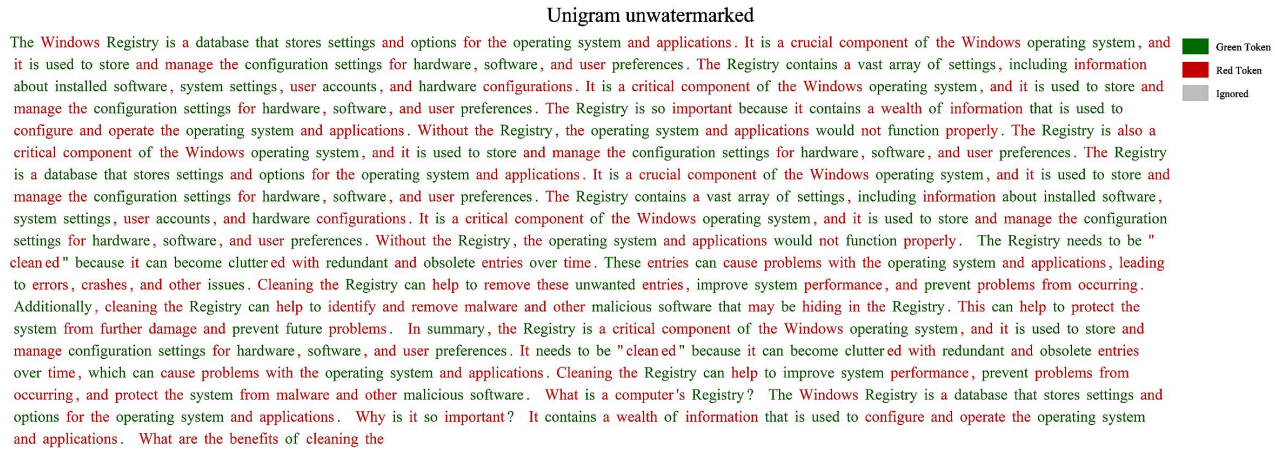


Figure 12. Unwatermarked Unigram example

requires identifying positions with sufficient semantics-preserving redistribution room, rather than uniformly enlarging token-level rewards.

Table 12. Results across different reward weighting.

		EWD			SWEET		
Model	**Weight**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
Qwen3-0.6B	$1 - p_{\max}$	54.3	80.5	0.73	50.5	66.3	0.79
	1	42.8	67.5	0.71	39.0	68.8	0.71
	$p_{\max}$	40.5	73.0	0.67	29.8	41.0	0.75
Qwen3-4B	$1 - p_{\max}$	56.5	87.0	0.73	52.3	71.3	0.75
	1	35.5	64.3	0.68	28.8	38.3	0.78
	$p_{\max}$	28.0	47.3	0.70	25.0	43.0	0.68

C.4. Effectiveness of conservative semantic rewards

We further study how the design of the sequence-level semantic reward affects spoofing performance. Specifically, we compare four variants: Min., Avg., Hum., and W.M., where Min. denotes the minimum of the semantic similarities to the human-written text and the watermarked rewrite, Avg. denotes their average, and Hum. and W.M. use only the human-written text or the watermarked rewrite as the semantic reference, respectively. We use the same training hyperparameter settings reported in Table 8, the results are shown in Table 13.

We observe that Min. consistently achieves the best SSR across both attacker models and all three watermarking schemes. For Qwen3-0.6B, Min. yields the highest SSR and SR on EWD, SWEET, and PF, outperforming the other reward variants by a clear margin. A similar trend holds for Qwen3-4B, where Min. again gives the best SSR and SSR, most notably improving PF from 50.3% and 51% under Avg. and W.M. to 62%. These results suggest that a conservative semantic reward provides a better optimization target for watermark spoofing. We attribute this improvement to the fact that successful spoofing must simultaneously preserve the meaning of the original text and remain close to the watermarked rewrite, which serves as the surrogate target distribution. Using only one reference may encourage one-sided alignment, while averaging the two scores can still mask weak alignment to one side. In contrast, Min. explicitly enforces both constraints and therefore yields more reliable semantic control during training.

C.5. Guidance of cross-entropy anchor

We examine the role of the cross-entropy (CE) anchor in stabilizing RLSpoofer training. We compare the RLSpoofer method (With Anchor) with two alternatives: removing the CE anchor (Without Anchor) and replacing RL training with supervised distillation on the same 100 training pairs (Distillation (100)). We adopt the training hyperparameters reported in

Table 13. Ablation on different semantic reward designs.

Model	Sem.R.	EWD			SWEET			PF		
		SSR	SR	P-SP	SSR	SR	P-SP	SSR	SR	P-SP
Qwen3-0.6B	Min.	54.3	80.5	0.73	50.5	66.3	0.79	33.3	52.0	0.66
	Avg.	44.8	78.3	0.70	48.8	60.5	0.77	26.3	42.3	0.72
	Hum.	42.3	69.0	0.70	43.0	59.5	0.75	24.5	40.0	0.76
	W.M.	48.5	71.3	0.70	46.8	61.3	0.76	25.5	51.3	0.64
Qwen3-4B	Min.	56.5	87.0	0.73	52.3	71.3	0.75	62.0	82.8	0.77
	Avg.	47.3	78.0	0.70	34.3	49.0	0.76	50.3	76.3	0.78
	Hum.	47.5	80.0	0.68	47.0	59.3	0.76	48.5	69.3	0.76
	W.M.	45.5	62.3	0.75	48.5	61.8	0.78	51.0	70.8	0.75

Appendix B. Specifically, for *Without Anchor* setting, we set the learning rate to 1×10^{-4} to stabilize the training process. The results are reported in Table 14.

We find that the CE anchor is critical for stable and effective spoofing. Removing it leads to substantial drops in SSR across all settings. For example, on Qwen3-4B, SSR decreases from 56.5% to 33.5% on EWD, and we observe similar degradations across all other settings. This shows that, even with carefully designed token-level and semantic rewards, unconstrained policy optimization can drift away from the desired watermarked rewriting distribution. Moreover, supervised distillation on the same 100 training pairs performs extremely poorly, with nearly zero SSR in all settings despite relatively high P-SP. This indicates that the gain of RLSpoofer does not come merely from imitation on limited data, but from the combination of RL-based distributional alignment and CE-based stabilization. Overall, these results confirm that the CE anchor is essential for maintaining a useful optimization trajectory while still allowing the policy to shift toward watermark-favored generations.

Table 14. Ablation on the effect of the cross-entropy anchor.

Model	Scheme	With Anchor			Without Anchor			Distillation (100)		
		SSR	SR	P-SP	SSR	SR	P-SP	SSR	SR	P-SP
Qwen3-0.6B	EWD	54.3	80.5	0.73	29.3	45.5	0.69	0.25	0.50	0.86
	SWEET	50.5	66.3	0.79	27.3	50.0	0.70	0.25	0.25	0.84
	PF	33.3	52.0	0.66	12.3	28.3	0.72	0.00	0.25	0.87
Qwen3-4B	EWD	56.5	87.0	0.73	33.5	50.3	0.73	0.00	0.25	0.93
	SWEET	52.3	71.3	0.75	26.8	44.3	0.78	0.25	0.25	0.94
	PF	62.0	82.8	0.77	25.5	45.3	0.70	0.00	0.25	0.84

C.6. Training set size ablation

We conduct an empirical study to examine the sensitivity of RLSpoofer to training set size. Specifically, we train the models on data generated by watermarked Llama3.1-8B-Instruct under three watermarking schemes, EWD, SWEET, and PF, using training sets containing 50, 100, and 200 samples, where each sample is standardized to 500 tokens. To ensure a comparable number of update steps across different training set sizes, we train for 20 epochs when using 50 samples and for 5 epochs when using 200 samples. For the 100-sample setting, we use the standard training schedule described in Table 8. We adopt the same remaining hyperparameters and report the results in Table 15.

We observe that RLSpoofer achieves strong spoofing performance with only 50 training samples. On the logit-based watermarks EWD and SWEET, increasing the training set generally improves SSR for both attacker models, with the best performance typically obtained using 200 samples. In contrast, on the sampling-based distortion-free PF watermark, enlarging the training set does not lead to consistent gains. For both Qwen3-0.6B and Qwen3-4B, the best PF performance is achieved with 100 samples, while using 200 samples leads to a noticeable drop in SSR. These results suggest that RLSpoofer is sample-efficient and that relatively small training sets are already sufficient to expose strong spoofing vulnerabilities, especially for the more challenging distortion-free setting.

Table 15. Effect of training set size on spoofing performance.

		EWD			SWEET			PF		
Model	**Samples**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
Qwen3-0.6B	50	44.3	77.0	0.72	42.5	65.5	0.74	12.3	19.3	0.72
	100	54.3	80.5	0.73	50.5	66.3	0.79	33.3	52.0	0.66
	200	56.8	85.0	0.74	51.3	71.5	0.79	20.5	37.0	0.76
Qwen3-4B	50	46.5	79.0	0.73	43.0	72.0	0.70	20.8	44.0	0.68
	100	56.5	87.0	0.73	52.3	71.3	0.75	62.0	82.8	0.77
	200	58.5	88.0	0.74	54.3	65.3	0.77	25.3	36.3	0.81

C.7. Sensitivity to surrogate model selection

We examine the sensitivity of RLSpoofer to the choice of surrogate reference model. Specifically, for each attacker, we compare two surrogate choices for approximating the target distributions: Qwen3-0.6B and the attacker itself (Self). The results are reported in Table 16. We find that surrogate choice has a limited impact on Qwen3-4B, where the two settings yield similar performance on EWD and SWEET, and Self performs slightly better on PF. In contrast, Qwen3-8B is much more sensitive: using Qwen3-0.6B consistently gives substantially higher SSR than Self across all three watermarking schemes. In particular, SSR improves from 45.3% to 60.5% on EWD, from 35% to 54.3% on SWEET, and from 29.8% to 58.5% on PF. These results suggest that surrogate selection becomes increasingly important as attacker capacity grows.

Table 16. Sensitivity of RLSpoofer to surrogate model choice.

		Qwen3-0.6B			Self		
Attacker	**Scheme**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
Qwen3-4B	EWD	56.5	85.8	0.73	56.5	87.0	0.73
	SWEET	53.3	73.0	0.75	52.3	71.3	0.75
	PF	56.5	83.8	0.76	62.0	82.8	0.77
Qwen3-8B	EWD	60.5	78.0	0.75	45.3	67.8	0.71
	SWEET	54.3	75.5	0.75	35.0	65.5	0.68
	PF	58.5	79.3	0.78	29.8	33.8	0.86

C.8. Cross watermark transferability of RLSpoofer

We further evaluate the zero-shot cross-watermark transferability of RLSpoofer. Specifically, we directly reuse the RLSpoofer-trained Qwen3-4B models obtained in Section 4.2, and test each model on the other watermarking schemes without any further tuning. The results are reported in Table 17.

We observe that transferability is not strictly constrained by the watermark family. In particular, the KGW-style logit-based watermarks EWD and SWEET exhibit substantial bidirectional transfer: training on EWD achieves 50.0% SSR on SWEET, while training on SWEET achieves 47.8% SSR on EWD. By contrast, interactions involving the sampling-based PF watermark are notably directional. Training on PF transfers effectively to EWD, achieving 52.5% SSR, whereas the reverse direction is much weaker, with EWD-to-PF achieving only 6.0% SSR. Similarly, PF-to-SWEET almost completely fails, yielding only 0.25% SSR. These results suggest that transferability is determined not only by broad watermark family similarity, but also by more intricate and directional overlaps in the vulnerabilities induced by different watermarking schemes.

Table 17. Cross-watermark transferability of RLSpoofer on Qwen3-4B.

Test	EWD			SWEET			PF		
Train	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**	**SSR**	**SR**	**P-SP**
EWD	56.5	87.0	0.73	50.0	64.8	0.76	6.00	10.0	0.78
SWEET	47.8	58.8	0.78	52.3	71.3	0.75	47.0	70.5	0.72
PF	52.5	80.5	0.75	0.25	1.00	0.61	62.0	82.8	0.77

C.9. Limitations

Limitations Although RLSpoofer demonstrates efficacy in evaluating the resilience of watermarks against spoofing attacks, it has several limitations. First, our method does not directly optimize the true detector-level spoofing objective. Instead, it relies on a tractable distributional surrogate together with reference-model-based local preference proxies for human-like and watermark-conditioned generation. Accordingly, our theory should be interpreted as motivating the reward design rather than providing a detector-agnostic guarantee of spoof success. Second, the practical effectiveness of RLSpoofer depends on the quality of these surrogate signals and can be sensitive to design choices such as reward weights, cross-entropy anchoring, and the choice of reference model. This dependency is inherent to the strict black-box setting: since the attacker has no access to the detector, secret key, or target model internals, RLSpoofer must estimate watermark-favored local preferences from observable watermarked examples. The approximation may be less accurate when the target system is fully closed-source and differs substantially from the surrogate in tokenization, vocabulary, decoding rule, or post-processing pipeline. This does not affect our controlled evaluation, where all methods are compared under consistent open-model implementations, but it may matter when extending RLSpoofer to highly opaque commercial deployments. In such settings, additional surrogate validation and detector-free stability criteria may be needed. Overall, RLSpoofer should be viewed as a practical stress-testing framework for assessing spoofing resilience, rather than as a universal guarantee of spoof success across all possible watermark deployments.