A Reinforcement Learning Framework for Robust and Secure LLM Watermarking

Li An[1], Yujian Liu[1], Yepeng Liu[1], Yuheng Bu[1]*, Yang Zhang[2]*, Shiyu Chang[1]*
[1]UC Santa Barbara, [2]MIT-IBM Watson AI Lab
{li_an, yujianliu, yepengliu, buyuheng, chang87}@ucsb.edu
Yang.Zhang2@ibm.com

Abstract

Watermarking has emerged as a promising solution for tracing and authenticating text generated by large language models (LLMs). A common approach to LLM watermarking is to construct a green/red token list and assign higher or lower generation probabilities to the corresponding tokens, respectively. However, most existing watermarking algorithms rely on heuristic green/red token list designs, as directly optimizing the list design with techniques such as reinforcement learning (RL) comes with several challenges. First, desirable watermarking involves multiple criteria, i.e., detectability, text quality, robustness against removal attacks, and security against spoofing attacks. Directly optimizing for these criteria introduces many partially conflicting reward terms, leading to an unstable convergence process. Second, the vast action space of green/red token list choices is susceptible to reward hacking. In this paper, we propose an end-to-end RL framework for robust and secure LLM watermarking. Our approach adopts an anchoring mechanism for reward terms to ensure stable training and introduces additional regularization terms to prevent reward hacking. Experiments on standard benchmarks with two backbone LLMs show that our method achieves a state-of-the-art trade-off across all criteria, with notable improvements in resistance to spoofing attacks without degrading other criteria. Our code is available at https://github.com/UCSB-NLP-Chang/RL-watermark.

1 Introduction

Large language models (LLMs) now underlie many public-facing applications, producing text that is increasingly difficult to distinguish from human writing. As a result, watermarking, which embeds imperceptible yet algorithmically-detectable patterns into LLM-generated text, has become a key line of defense for provenance tracking and content authentication (Pan et al., 2024; Liu et al., 2024; Zhao et al., 2025).

A desired watermarking algorithm should satisfy four criteria: ① Detectability: Any watermarked text should be accurately detected, and any unwatermarked text should not be falsely detected; ② Text quality: The watermarked text should have similar quality to the unwatermarked text; ③ Robustness to removal attacks: The watermarks should remain detectable under paraphrasing; and ④ Security against spoofing attacks: The watermarks should be removed after malicious modifications, such as flips of sentiments and insertions of hate speech. To design effective watermarking algorithms, a common approach is based on a green/red token list, where the token vocabulary is divided into a green list and a red list. During generation, the probability of generating green tokens is increased, and that of generating red tokens is decreased. Consequently, by counting the frequency of green versus red tokens, one can effectively detect watermarked texts (Kirchenbauer et al., 2023; Zhao et al., 2023a; Kuditipudi et al., 2023a). Although watermarking performance heavily depends on the design of the green/red list, existing approaches typically determine it randomly, leading to a suboptimal trade-off across multiple criteria.

More recently, semantic-aware watermarking methods (Liu and Bu, 2024; Guo et al., 2024; Liu et al.) have been proposed, where a mapping model encodes the prior context, and the green/red token list is determined by the semantic embeddings. By contrastively training the mapping model to be insensitive to semantic-preserving operations and sensitive to semantic-distorting operations (An et al., 2025), the watermarking can be simultaneously robust to paraphrase removal attacks and secure against spoofing attacks. However, An et al. (2025) only trains the model to distinguish different operations in the embedding space, which does not

*Equal advising and contribution.

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics
Volume 1: Long Papers, pages 7181–7198

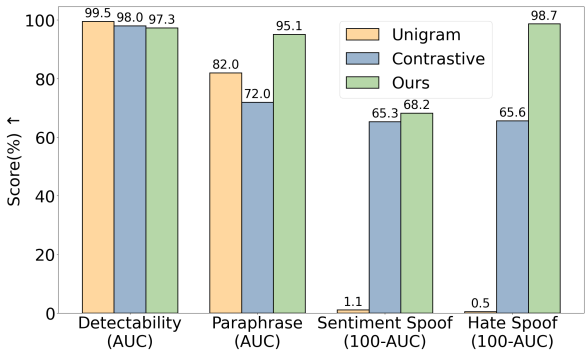

Figure 1: Performance comparison (higher is better) on detectability, robustness against paraphrase attacks, and security against sentiment and hate speech spoofing attacks. For detectability and paraphrase attack, AUCs are reported. For sentiment and hate attacks, scores are calculated using the complements (*i.e.*, 100-AUC) of the corresponding AUCs.

necessarily translate to improved end performance of watermarking. Figure 1 shows the performance of An et al. (2025), which illustrates that as security improves (sentiment and hate), performance on detectability and robustness (paraphrase) degrade, indicating a tradeoff among the criteria.

In this paper, we propose an end-to-end reinforcement learning (RL) framework to directly optimize the watermarking design. Building on semantic-aware approaches, we employ a mapping model as a policy to generate the green/red token list. The policy is then optimized through a reward that balances multiple criteria under a unified framework. Specifically, given a generated green/red token list, a watermarked text is sampled, and the policy is rewarded for correctly detecting the watermark in the generated text but not in its unwatermarked counterpart. The watermarked text is then modified in two ways: through semantic-preserving edits (e.g., paraphrasing) and through semantic-distorting edits (e.g., sentiment flips and insertions of hate speech). The policy earns additional rewards when the watermark survives the first type of edits but disappears after the second.

However, such an RL training setup faces two key challenges. The first challenge lies in the inherently conflicting nature of the watermarking criteria: detectability and robustness require watermarks to be invariant to paraphrasing, whereas security demands sensitivity to semantic-distorting modifications. Balancing these partially opposing objectives within a single reward function often leads to unstable training dynamics. The second challenge arises from the large action space of the mapping model–each token can be assigned as a green or red token. In this setting, the model may exploit shortcuts in the reward function to achieve high reward scores without genuinely improving watermarking performance, a phenomenon analogous to reward hacking.

To address these challenges, our method includes two key designs. First, we adopt an anchored mechanism that constrains the fraction of green tokens in unwatermarked texts toward 50% to address the partial conflicts among different criteria and stabilize training. Second, we augment the reward function with adversarial evaluations. In addition to applying various attacks to watermarked texts, we also generate attacks for unwatermarked texts and reward the policy if the attacked unwatermarked texts maintain a 50% of green tokens. This discourages degenerate strategies such as naive hate speech detection.

We evaluate our method on two widely used benchmarks for LLM watermarking with two popular base LLMs. Experiments show that our method outperforms strong baselines by better satisfying the four criteria. Additional analyses also demonstrate the importance of our two key designs. We summarize our contributions as follows:

- We propose an end-to-end RL framework to optimize the green/red token list generation policy, which jointly considers detectability, robustness, and security within a unified framework.
- We introduce an anchored reward mechanism to mitigate conflicts among reward components and stabilize training.
- We incorporate adversarial regularization in the reward function to prevent reward hacking.
- Experiments on standard benchmarks demonstrate that our method achieves a better trade-off among the desired criteria.

2 Related Works

2.1 LLM Watermark

Watermarking AI-generated text (Pan et al., 2024; Liu et al., 2024; Zhao et al., 2025) is crucial for ensuring transparency, accountability, and the ability to distinguish between human and machine-generated content. An in-process LLM watermark usually embeds a hidden signal directly into AI-generated text during generation by manipulating the decoding process (Kirchenbauer et al., 2023; Zhao et al., 2023b; Liu et al., 2023a,b; Zhu et al., 2024; Dathathri et al., 2024; Lee et al., 2023; He

et al., 2025, 2024; Liu et al., 2025b; Hu et al., 2023; Christ et al., 2024; Chen et al., 2025; Kuditipudi et al., 2023b), fine-tuning model weights (Xu et al., 2024; Zhao et al., 2023c; Xu et al., 2025; Block et al., 2025), or using a watermarking instruction (Liu et al., 2025a). A post-hoc LLM watermark (Qiang et al., 2023; Zhang et al., 2024; Yang et al., 2022) does not require direct access to the original LLM. Instead, it embeds a watermark into the generated text using a separate LLM, such as by paraphrasing the unwatermarked text with a watermarked LLM (An et al., 2025) or incorporating selected keywords using LLMs (Chang et al., 2024). Specifically, Kirchenbauer et al. (2023) uses the previous token as a hash to partition the LLMs' vocabulary into green and red lists, and softly encourages the selection of green tokens during text generation to embed the watermark. Guo et al. (2025) proposes a policy-driven approach for code watermarking by optimizing token selections during the next-token prediction. Xu et al. (2024) embeds a watermark into LLM weights by co-training the LLM and detector based on RL. Our watermarking method also employs RL, but it differs significantly from Xu et al. (2024) in two key aspects. First, Xu et al. (2024) fully fine-tunes the LLM, whereas our approach leaves the original LLM weights untouched and instead trains a lightweight semantic mapping model, thereby maintaining the integrity of the base model. Second, rather than training a dedicated detector, we embed the watermark by perturbing the sampling process and use statistical methods for detection.

2.2 LLM Watermark against Spoofing Attack

The spoofing attack (Sadasivan et al., 2023; Jovanović et al., 2024; Pang et al., 2024) poses a severe threat to the security of LLM watermarks, especially the reputation of the LLM owners. Specifically, Sadasivan et al. (2023) proposes a watermark forgery spoofing attack that forges watermarked text by reverse-engineering the green and red token lists used in the KGW watermarking method (Kirchenbauer et al., 2023). Pang et al. (2024) introduces the piggyback spoofing attack, which subtly modifies a few words to change the overall meaning or inserts harmful content, such as hate speech, into watermarked text. Although the semantic intent is significantly altered, the watermark remains detectable, potentially leading to false attribution of the harmful content to the LLM owner. To enhance the security against spoofing attacks, several semantic-aware watermarking methods have been proposed (Liu and Bu, 2024; An et al., 2025; Yi et al., 2025; Cai et al., 2025; Hou et al., 2023; Fu et al., 2024; Ren et al., 2023). Specifically, Liu and Bu (2024) uses a pre-trained sentence embedding model as a semantic mapping model to extract the semantic meaning of watermarked text, thereby improving resistance to forgery spoofing attacks. However, this approach struggles to defend against piggyback spoofing attacks, as pre-trained sentence embedding models often fail to capture significant semantic shifts caused by minor textual modifications. An et al. (2025) enhances watermark security against piggyback spoofing attacks by contrastively training a semantic mapping model to be sensitive to semantic-distorting changes while insensitive to semantic-preserving changes. This approach improves the trade-off between robustness and security. In this paper, we adopt an end-to-end training strategy to achieve a more favorable trade-off.

3 Background

3.1 Problem Formulation

In this work, our goal is to develop a watermarking algorithm that satisfies the following four criteria:

- **Detectability:** The system should detect the presence of watermarks in watermarked text with a high success rate while avoiding false detection on unwatermarked text.
- **Text quality:** The embedded watermarks should not disturb the quality of the generated text.
- **Robustness against removal attacks:** The watermarks should remain detectable after semantic-preserving edits such as paraphrasing, so that they cannot be easily removed.
- **Security against spoofing attacks:** The watermarks should be removed after malicious modifications like hate speech insertion and sentiment flipping, so that attackers cannot forge watermarked texts containing malicious content.

We adopt the post-hoc watermarking setting in An et al. (2025), where given an unwatermarked text, which can be generated by LLMs or humans, we generate a watermarked version of it by paraphrasing the unwatermarked input and inserting watermarks into the paraphrase.

3.2 Semantic-aware Watermarking Algorithm

Our method builds upon the semantic-aware watermarking algorithm (Liu and Bu, 2024; An et al., 2025). Particularly, the watermarking system com-

prises two components: a mapping model $M_{\boldsymbol{\theta}}$ and a backbone LLM. To generate a watermarked text, the following two steps are conducted:

Step 1: Green/red token list construction. Given an unwatermarked input text $\boldsymbol{x}^{\text{uw}}$, the mapping model maps it to a vocabulary-size vector $\boldsymbol{z} = M_{\boldsymbol{\theta}}(\boldsymbol{x}^{\text{uw}}) \in \mathbb{R}^{|\mathcal{V}|}$, where $\mathcal{V}$ is the vocabulary. Tokens with positive values are labeled as green tokens, forming the green token list $\mathcal{G} = \{v : \boldsymbol{z}[v] > 0\}$, while the remaining tokens constitute the red token list. Since the mapping model builds on the hidden representations of the input text, the constructed green/red token list will depend on the semantic information of $\boldsymbol{x}^{\text{uw}}$.

Step 2: Watermark injection. Given a paraphrasing prompt $\boldsymbol{q}$ and the input text $\boldsymbol{x}^{\text{uw}}$, the backbone LLM generates the watermarked output $\boldsymbol{x}^{\text{wm}}$. Specifically, at each decoding step t, the model produces logits $\boldsymbol{l}_t = \text{LLM}(\boldsymbol{q}, \boldsymbol{x}^{\text{uw}}, \boldsymbol{x}^{\text{wm}}_{<t})$. The watermark is then embedded by perturbing and increasing the logits of the green tokens: $\hat{\boldsymbol{l}}[v] = \boldsymbol{l}[v] \cdot (1 + \delta \mathbb{1}(v \in \mathcal{G}))$, where δ is the watermarking strength. Finally, the next token $\boldsymbol{x}^{\text{wm}}_t$ is sampled according to the logits $\hat{\boldsymbol{l}}$. In this way, the green tokens are more likely to be sampled, thereby embedding the watermark signal in generated text.

To detect the presence of watermarks in an unknown text, the same mapping model $M_{\boldsymbol{\theta}}$ is used to construct the green/red token list. The text is marked as watermarked if the percentage of green tokens in the text is above a pre-defined threshold.

4 Methodology

In this section, we formulate the construction of the green/red token list as an RL problem and optimize the policy directly with the criteria *detectability*, *robustness*, and *security* within a unified framework.

4.1 RL Framework for Watermarking

We cast the construction of the green/red token list as an RL problem in which the mapping model $M_{\boldsymbol{\theta}}$ serves as the actor. Given an unwatermarked input $\boldsymbol{x}^{\text{uw}}$, the state is defined as $s = \boldsymbol{x}^{\text{uw}}$. An action is a green/red assignment over the vocabulary,

$$\boldsymbol{g} \in \{0,1\}^{|\mathcal{V}|},\ \boldsymbol{g}[v] = 1 \text{ iff token } v \text{ is in green list.}$$

The actor induces a stochastic policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{g}|s)$ that outputs a distribution over such assignments conditioned on the state. Taking an action corresponds to fixing the green/red token list according to $\boldsymbol{g}$. Then, a watermarked text $\boldsymbol{x}^{\text{wm}}$ is sampled from a frozen backbone LLM following the procedure in Section 3.2, using the green/red token list $\boldsymbol{g}$. The environment applies a suite of attacks to $\boldsymbol{x}^{\text{wm}}$ and returns a scalar reward $R(s, \boldsymbol{g})$ aggregating detectability, robustness, and security. The detailed reward design is in Section 4.2.

Formally, our objective is to maximize the expected reward

$$\max_{\boldsymbol{\theta}} \ \mathbb{E}_{\boldsymbol{g} \sim \pi_{\boldsymbol{\theta}}(\cdot|s)}\big[R(s, \boldsymbol{g})\big],$$

which encourages policies that satisfy the desired watermarking criteria.

Instantiating the actor. To obtain a valid stochastic policy, the actor maps $\boldsymbol{x}^{\text{uw}}$ to a vocabulary-sized vector $\boldsymbol{z} = M_{\boldsymbol{\theta}}(\boldsymbol{x}^{\text{uw}})$ and converts it to per-token green-list probabilities $\boldsymbol{p} = \sigma(\boldsymbol{z})$, where $\sigma(\cdot)$ is the sigmoid function. An action $\boldsymbol{g}$ is then sampled independently across tokens,

$$\boldsymbol{g}[v] \sim \text{Bernoulli}(\boldsymbol{p}[v]).$$

Therefore, for a particular action $\boldsymbol{g}$, its probability can be calculated as:

$$\pi_{\boldsymbol{\theta}}(\boldsymbol{g}|s) = \prod_{v \in \mathcal{V}} \boldsymbol{p}[v]^{\boldsymbol{g}[v]} (1 - \boldsymbol{p}[v])^{1 - \boldsymbol{g}[v]}.$$

This stochasticity makes the list-generation process amenable to policy-gradient optimization.

Training procedure. For each training instance, we perform the following three steps, as illustrated in Figure 2:

Step 1: Generation. From state $s = \boldsymbol{x}^{\text{uw}}$, sample $\boldsymbol{g} \sim \pi_{\boldsymbol{\theta}}(\cdot|s)$, construct the green list $\mathcal{G}$, and generate $\boldsymbol{x}^{\text{wm}}$ with the frozen backbone LLM under the corresponding watermark injection.

Step 2: Reward calculation. Compute $R(s, \boldsymbol{g})$ by evaluating the detectability on $\boldsymbol{x}^{\text{wm}}$ and its attacked variants, as well as on unwatermarked counterparts (details in Section 4.2).

Step 3: Optimization. Update $\boldsymbol{\theta}$ to maximize the expected reward using GRPO (Shao et al., 2024), while keeping the backbone LLM frozen.

4.2 Reward Function

We construct the reward function by jointly optimizing multiple watermarking criteria–*detectability*, *robustness*, and *security*–within a unified formulation. Before describing each reward component, we first define the detection score, which quantifies the likelihood that a text is watermarked.

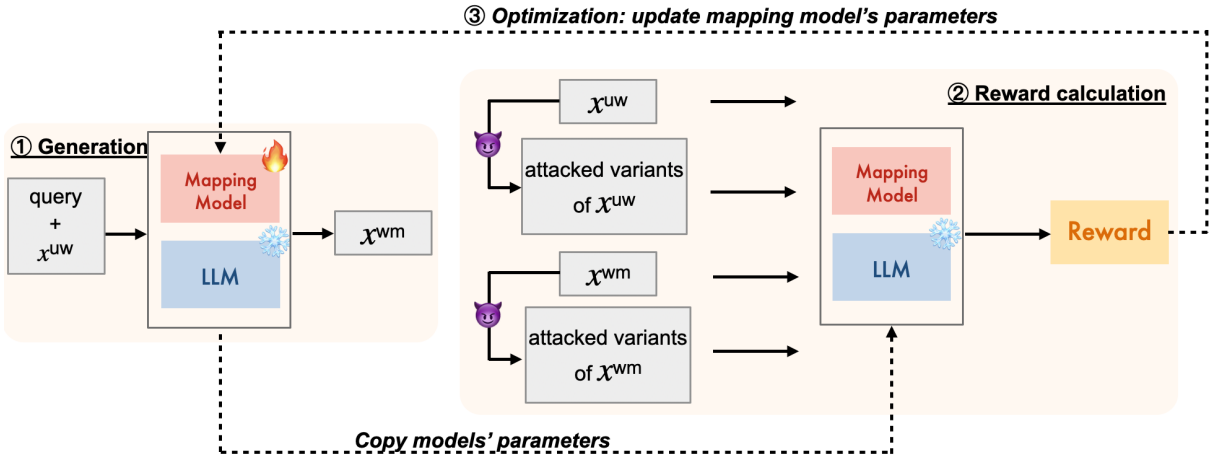

Figure 2: The RL training framework of our method. For each training instance, we perform the following three steps: ① Given an unwatermarked text $\boldsymbol{x}^{\text{uw}}$, sample and construct the green/red token assignment $\boldsymbol{g}$, and generate a watermarked text $\boldsymbol{x}^{\text{wm}}$. ② Compute the reward of $\boldsymbol{g}$ by evaluating the detection score on $\boldsymbol{x}^{\text{wm}}$ and its attacked variants, as well as on unwatermarked counterparts. ③ Update the mapping model to maximize the expected reward using GRPO, while keeping the backbone LLM frozen.

Detection score. Given a text $\boldsymbol{x} = (x_1, \ldots, x_L)$, of length L, we obtain the per-token green-list probabilities following Section 4.1: $\boldsymbol{p} = \sigma(M_{\boldsymbol{\theta}}(\boldsymbol{x}))$, where $\boldsymbol{p}[v]$ represents the probability of token v being assigned as a green token. The detection score $D(\boldsymbol{x})$ is computed as the average probability of its tokens being green:[1]

$$D(\boldsymbol{x}) = \frac{1}{L} \sum_{t=1}^{L} \log \boldsymbol{p}[x_t]. \tag{1}$$

A higher $D(\boldsymbol{x})$ indicates a higher likelihood that the text contains watermarks.

Reward formulation. Let $\boldsymbol{x}^{\text{uw}}$ be the original unwatermarked input, $\boldsymbol{x}^{\text{wm}}$ its watermarked version, $\boldsymbol{x}^{\text{para-wm}}$ a semantic-preserving paraphrase of $\boldsymbol{x}^{\text{wm}}$, and $\boldsymbol{x}^{\text{sent-wm}}$, $\boldsymbol{x}^{\text{hate-wm}}$ the spoofed variants produced by flipping the sentiments (*e.g.,* positive to negative) and inserting hate speech, while preserving the main meaning of the original text (detailed procedure in Appendix A.4). We instantiate three reward terms that align with the desired watermarking criteria:

- **Detectability:** We encourage a large margin between watermarked and unwatermarked scores, $D(\boldsymbol{x}^{\text{wm}}) - D(\boldsymbol{x}^{\text{uw}})$, so that watermarks can be accurately detected.
- **Robustness:** We favor high detection scores on paraphrases $D(\boldsymbol{x}^{\text{para-wm}})$, so that watermarks are resilient to removal attacks.
- **Security:** We favor low detection scores for the spoofing-attacked versions $D(\boldsymbol{x}^{\text{sent-wm}})$, $D(\boldsymbol{x}^{\text{hate-wm}})$, so that watermarks can be removed after malicious edits.

However, naively training an RL framework with the above reward faces two key challenges.

Challenge 1: Partial conflicts among criteria. Due to the complex interplay and partial conflicts among the criteria, simply combining multiple reward terms can lead to training instability. In particular, improving detectability and robustness requires high detection scores for $D(\boldsymbol{x}^{\text{wm}})$, $D(\boldsymbol{x}^{\text{para-wm}})$, and low scores for $D(\boldsymbol{x}^{\text{uw}})$. At the same time, achieving strong security requires low detection scores for $D(\boldsymbol{x}^{\text{sent-wm}})$ and $D(\boldsymbol{x}^{\text{hate-wm}})$, ideally lower than $D(\boldsymbol{x}^{\text{uw}})$. Therefore, simply pushing down $D(\boldsymbol{x}^{\text{uw}})$ is insufficient, and naively summing these terms causes instability.

Challenge 2: Reward hacking. The large action space (per-token green/red assignments) enables possible shortcuts, which the model may exploit to achieve high reward scores without genuinely improving watermarking performance. For example, to achieve low detection scores for $D(\boldsymbol{x}^{\text{hate-wm}})$, the policy could degenerate to a hate speech detector, so that as long as an input text is classified as containing hate speech, the model could assign near-zero green-list probabilities to all tokens, regardless of whether it is watermarked or not. To address these challenges, we introduce the following two key designs in the reward function.

[1] No entropy filtering (Liu and Bu, 2024) for simplicity.

Anchored reward mechanism. To address the first challenge, the detection score of unwatermarked text $D(\boldsymbol{x}^{\text{uw}})$ should approach a neutral midpoint so that spoofed variants can be reliably driven below it, and the watermarked texts' scores can remain above it. Therefore, we penalize deviations of the unwatermarked detection score from 0.5 by replacing the raw $D(\boldsymbol{x}^{\text{uw}})$ with its absolute deviation term $|D(\boldsymbol{x}^{\text{uw}}) - 0.5|$. We further discuss the choice of the anchored value and conduct a sensitivity analysis in Appendix B.3.

Anti-reward-hacking regularization. To discourage degenerated policies such as naive hate speech detection, we apply the same anchoring mechanism to attacked *unwatermarked* texts–a paraphrase $\boldsymbol{x}^{\text{para-uw}}$, sentiment-flipped $\boldsymbol{x}^{\text{sent-uw}}$, and hate-speech-inserted $\boldsymbol{x}^{\text{hate-uw}}$–so that the policy cannot merely learn to capture the characteristics of the attacks. This forces unwatermarked content (even after attacks) to remain neutral, preventing behaviors like hate-speech detection.

Final reward. The complete reward function combines multiple detection scores, with the regularized terms for the unwatermarked text:

$$\begin{aligned} R = {} & D(\boldsymbol{x}^{\text{wm}}) + D(\boldsymbol{x}^{\text{para-wm}}) - D(\boldsymbol{x}^{\text{sent-wm}}) - D(\boldsymbol{x}^{\text{hate-wm}}) \\ & - |D(\boldsymbol{x}^{\text{uw}}) - 0.5| - |D(\boldsymbol{x}^{\text{para-uw}}) - 0.5| \\ & - |D(\boldsymbol{x}^{\text{sent-uw}}) - 0.5| - |D(\boldsymbol{x}^{\text{hate-uw}}) - 0.5|. \end{aligned} \tag{2}$$

5 Experiments

5.1 Experiment Settings

Evaluation datasets. We evaluate our method on two datasets commonly used in LLM watermarking: the `realnewslike` subset of C4 (Raffel et al., 2020) and the LFQA dataset (Krishna et al., 2023). In our main results, we sample 200 texts from each dataset as the original unwatermarked texts. The evaluation results on a larger scale of 500 texts can be found in Appendix B.4. Specifically, for C4, we use the original document, and for LFQA, we use the annotated gold completion.

Metrics. Following An et al. (2025), we report ROC-AUC scores for detecting watermarked text under four conditions: the original watermarked text, and its variants under paraphrasing, sentiment spoofing, and hate speech spoofing attacks. Higher ROC-AUC scores on the original watermarked and paraphrased texts indicate better detectability and robustness, respectively. By contrast, lower AUC scores on the sentiment and hate speech spoofed texts are preferred, as they suggest that watermarks are successfully removed by malicious edits, reflecting stronger security. We further compute an overall AUC score following An et al. (2025), by averaging the AUCs for detectability and robustness, along with the complements (*i.e.*, 100-AUC) of the two security-related AUCs. We also report perplexity to assess the text quality of watermarked generations. In addition to perplexity, we evaluate text quality and semantic relevance using an LLM-as-judge protocol; the detailed results are reported in Appendix B.5.

Baselines. We compare our approach with five baseline methods: KGW(Kirchenbauer et al., 2023), UNIGRAM(Zhao et al., 2023a), ADAPTIVE (Liu and Bu, 2024), POSTMARK (Chang et al., 2024), and CONTRASTIVE (An et al., 2025).

The first two methods, KGW and UNIGRAM, rely on token-level information to determine the green/red token split. KGW determines the green/red token split using the previous token and a random hash function, while UNIGRAM improves robustness by using a fixed split. Neither method considers the semantic content of the text. ADAPTIVE introduces semantic awareness by leveraging prefix embeddings to guide the green/red split. It also utilizes the entropy of the LLM's output to adaptively choose tokens on which to inject watermarks. POSTMARK (Chang et al., 2024), a recent post-hoc method that inserts input-conditioned tokens directly into the generated text. Finally, we include CONTRASTIVE(An et al., 2025), which uses contrastive training to obtain a mapping model that is both sensitive to semantic-distorting operations and insensitive to semantic-preserving operations.

Although KGW, UNIGRAM, and ADAPTIVE were not originally designed for post-hoc watermarking, we adapt them to our setting by prepending a paraphrasing instruction to the target text (see Figure 6 for the prompt format). For fair comparison, we tune all methods to produce watermarked texts with similar perplexity levels, ensuring comparable text quality. The parameter details are included in Appendix A.2.

Training details. We initialize the mapping model with the contrastively trained semantic mapping model released by An et al. (2025) and further fine-tune it using our RL framework. We apply Group Relative Policy Optimization (GRPO) (Shao et al., 2024), an efficient and effective RL algorithm, and adopt an unbiased gradient formulation specific

Method	Dataset	ROC-AUC (%)				Overall AUC ↑	PPL↓
		Detectability ↑	Robustness ↑	Security ↓			
			Paraphrase	Sentiment Spoof	Hate Speech Spoof		
Llama-3.1-8B-Instruct							
KGW	C4	100.00	72.68	98.85	100.00	43.46	8.27
	LFQA	100.00	78.03	99.32	100.00	44.68	9.04
UNIGRAM	C4	99.54	81.96	98.44	99.54	45.88	8.23
	LFQA	99.98	86.12	98.94	99.98	46.80	8.81
ADAPTIVE	C4	99.78	72.18	96.50	99.35	44.03	8.77
	LFQA	99.97	70.45	97.14	99.91	43.34	9.90
POSTMARK	C4	99.99	89.03	94.07	99.87	48.77	9.21
	LFQA	99.93	87.20	95.54	99.47	48.03	9.21
CONTRASTIVE	C4	98.02	71.97	34.68	34.38	75.23	8.80
	LFQA	99.16	80.99	29.23	29.89	80.26	9.57
OURS	C4	97.30	95.11	31.81	1.31	**89.82**	9.01
	LFQA	98.08	100.00	37.31	3.00	**89.44**	9.83
Qwen2.5-7B-Instruct							
KGW	C4	99.12	67.92	94.04	99.08	43.48	8.97
	LFQA	99.58	67.38	95.51	99.56	42.97	9.23
UNIGRAM	C4	97.34	66.99	93.03	96.13	43.79	10.03
	LFQA	99.62	62.50	97.07	99.32	41.43	9.98
ADAPTIVE	C4	99.17	66.08	91.75	98.49	43.75	9.77
	LFQA	99.26	61.37	89.96	98.86	42.95	10.74
POSTMARK	C4	99.99	89.03	94.07	99.87	48.77	9.21
	LFQA	99.93	87.20	95.54	99.47	48.03	9.21
CONTRASTIVE	C4	95.80	67.09	32.54	18.58	77.94	9.57
	LFQA	98.94	81.27	37.58	29.04	78.40	10.99
OURS	C4	95.12	98.27	34.16	3.45	**88.95**	9.29
	LFQA	95.24	98.38	25.22	4.84	**90.89**	11.19

Table 1: Performance of watermarking methods. The Overall AUC is the average of the four AUC scores.

to our setting (see details in Appendix B.1). For training data, we use the dataset released by An et al. (2025), where only the original texts is used as unwatermarked inputs. During reward computation, we use `Qwen3-14B` (Team, 2025) as the attacker model to generate paraphrased and sentiment spoofed variants, and details of the attack process and prompt templates are provided in Appendix A.4. We evaluate the model on the validation set during training, and select the checkpoint with the highest overall AUC as the final model. We further analyze the computational overhead of the RL training procedure and the inference-time cost for practical deployment in Appendix A.5.

5.2 Main Results

Table 1 compares our method with baseline watermarking algorithms across the four criteria: *detectability*, *robustness*, *security*, and *text quality*. For each backbone–`Llama-3.1-8B-Instruct` and `Qwen2.5-7B-Instruct`–we train a separate mapping model while freezing the backbone.

As can be observed, our method achieves the highest overall AUC on both datasets and backbones, while maintaining comparable perplexity to baselines. Traditional methods such as KGW, UNIGRAM, ADAPTIVE, and POSTMARK all struggle with removing watermarks in spoofing-attacked texts, yielding AUCs above 90 even when the

text is maliciously modified. Although the CONTRASTIVE baseline improves spoofing resilience, the trade-off on other criteria is still suboptimal. By contrast, our end-to-end RL framework achieves a better trade-off across all three key dimensions–detectability, robustness, and security–without sacrificing perplexity. In particular, our approach not only improves robustness against paraphrasing attacks, but also shows strong security under spoofing attacks, especially hate-speech spoofing. The ablation study in Section 5.3 further shows how each component of our framework contributes to the overall performance improvement.

We further evaluate robustness under a broader set of attacks unseen during training, including learnable spoofing, alternative paraphrasing strategies, and translation-based attacks; detailed implementation details and results are reported in Appendix B.6.

5.3 Ablation Study

Unwatermarked score	ROC-AUC (%)			
	Detectability ↑	Paraphrased ↑	Sentiment Spoof↓	Hate Speech Spoof ↓
Raw	98.46	99.04	52.00	15.88
Anchored	97.30	95.11	31.81	1.31

Table 2: Performance of models trained using rewards with or without the anchored reward mechanism.

We now investigate the impacts of the two key designs in Section 4.2.

Anchored reward mechanism. To address the partial conflicts among different criteria, we introduce an anchored reward mechanism that stabilizes training by using the absolute difference between the unwatermarked text detection score and 0.5. This design prevents the model from excessively reducing or inflating the unwatermarked detection score, maintaining it near the neutral midpoint.

To evaluate the effectiveness of this design, we compare it with a naive variant of the reward function in which the anchored term is replaced by the raw unwatermarked detection score, *i.e.,* $D(\boldsymbol{x}^{\text{wm}}) - D(\boldsymbol{x}^{\text{uw}})$. Both models are trained under identical settings using the `Llama-3.1-8B-Instruct` backbone. As shown in Table 2, removing the anchored reward leads to noticeably higher AUCs on spoofing attacks, indicating worse security. Although removing the anchor brings a slight increase in detectability and robustness, the overall trade-off across all criteria is worse. This result is consistent with our observation that, without anchoring, the unwatermarked detection scores decrease excessively, making it harder to distinguish spoofing-attacked watermarked text from unwatermarked text.

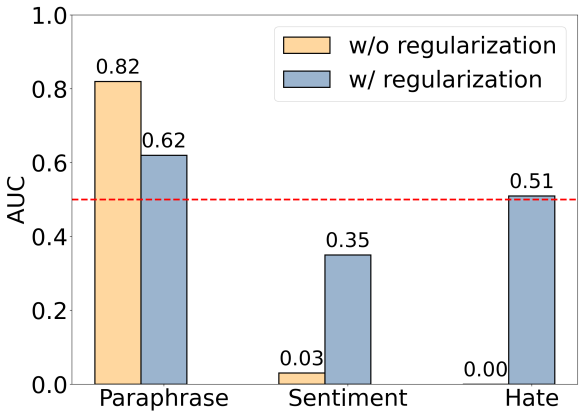

Figure 3: Effect of anti-reward-hacking regularization on AUCs of attacked unwatermarked text.

Anti-reward-hacking regularization. To evaluate the effectiveness of the proposed anti-reward-hacking regularization, we train a variant of the mapping model that does not contain the regularization terms, *i.e.,* we only restrict the detection score of the original unwatermarked text ($D(\boldsymbol{x}^{\text{uw}})$) around 0.5, but not its attacked versions ($D(\boldsymbol{x}^{\text{para-uw}}), D(\boldsymbol{x}^{\text{sent-uw}}), D(\boldsymbol{x}^{\text{hate-uw}})$).

For evaluation, we use the same set of 200 unwatermarked samples from the `C4` dataset as in Table 1. We apply the same paraphrasing, sentiment-spoofing, and hate-speech-spoofing prompts directly to the original unwatermarked text. We then calculate the ROC-AUC scores for distinguishing the attacked texts from the original ones. Since the whole process does not add any watermarks, a watermarking detector should not be able to distinguish the attacked texts and original texts, so the AUC values should be close to 0.5.

As shown in Figure 3, adding regularization brings the AUCs of attacked unwatermarked variants closer to 0.5 (indicated as the red dashed line). Specifically, the model without regularization yields a high AUC for paraphrased text and near-zero AUCs for sentiment and hate-speech variants. This suggests that the model is functioning as a paraphrase, sentiment-spoofing, and hate-speech-spoofing detector, as it accurately differentiates the attacked variants from the original texts, even if no watermarks are added. By contrast, the regularized model is less accurate in identifying the

attacked variants, indicating that the regularization effectively mitigates reward hacking and leads to a more reliable and interpretable reward signal.

We further provide an additional ablation study that removes individual reward components to assess their respective contributions. Detailed results are reported in Appendix B.2.

6 Conclusion

This paper proposes an end-to-end reinforcement learning framework for robust and secure LLM watermarking. By directly optimizing the green–red token list generation policy under a unified reward objective, the method achieves a more balanced trade-off across multiple criteria. An anchored reward mechanism and adversarial regularization stabilizes training and prevents reward hacking. Experiments on multiple benchmarks and backbones show that it achieves superior resistance to removal and spoofing attacks without compromising text quality or detectability.

Limitations

Despite its promising results, the proposed approach has several limitations that can be further explore in future works. First, the mapping model needs to be retrained for every new backbone LLM, which increases computational cost and limits scalability across architectures. Second, the training objective does not explicitly incorporate text quality metrics, such as fluency or coherence, which may cause the watermarking process to degrade generation quality even if detectability and robustness are improved.

Finally, similar to most policy-gradient-based reinforcement learning methods, our approach does not come with formal convergence guarantees. Existing theoretical analyses typically rely on strong assumptions, such as linear-quadratic settings or highly restricted policy classes, which do not directly apply to our setting with neural policies and complex, attack-driven reward functions. Developing theoretical guarantees for such general settings remains an important open problem and is beyond the scope of this work.

Acknowledgment

The work of Li An, Yujian Liu, and Shiyu Chang was partially supported by National Science Foundation (NSF) Grant IIS-2338252, NSF Grant IIS-2207052, NSF Grant IIS-2302730, and the Open Philanthropy Research Award. The work of Yuheng Bu was partially supported by NSF Grant OAC-2410693. The team acknowledge UFIT Research Computing for providing computational resources and support that contributed to the research results reported in this publication.

References

Li An, Yujian Liu, Yepeng Liu, Yang Zhang, Yuheng Bu, and Shiyu Chang. 2025. Defending llm watermarking against spoofing attacks with contrastive representation learning. *arXiv preprint arXiv:2504.06575*.

Adam Block, Ayush Sekhari, and Alexander Rakhlin. 2025. Gaussmark: A practical approach for structural watermarking of language models. *arXiv preprint arXiv:2501.13941*.

Yuhang Cai, Yaofei Wang, Donghui Hu, and Chen Gu. 2025. Machine never said that: Defending spoofing attacks by diverse fragile watermark. In *The 1st Workshop on GenAI Watermarking*.

Yapei Chang, Kalpesh Krishna, Amir Houmansadr, John Wieting, and Mohit Iyyer. 2024. Postmark: A robust blackbox watermark for large language models. *arXiv preprint arXiv:2406.14517*.

Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. 2025. Improved unbiased watermark for large language models. *arXiv preprint arXiv:2502.11268*.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139. PMLR.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, and 1 others. 2024. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823.

Yu Fu, Deyi Xiong, and Yue Dong. 2024. Watermarking conditional text generation for ai detection: Unveiling challenges and a semantic-aware watermark remedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18003–18011.

Yuxuan Guo, Zhiliang Tian, Yiping Song, Tianlun Liu, Liang Ding, and Dongsheng Li. 2024. Context-aware watermark with semantic balanced green-red lists for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22633–22646.

Zhimeng Guo, Huaisheng Zhu, Siyuan Xu, Hangfan Zhang, Teng Xiao, and Minhao Cheng. 2025. Optimizing token choice for code watermarking: A rl approach. *arXiv preprint arXiv:2508.11925*.

Haiyun He, Yepeng Liu, Ziqiao Wang, Yongyi Mao, and Yuheng Bu. 2024. Theoretically grounded framework for llm watermarking: A distribution-adaptive approach. *arXiv preprint arXiv:2410.02890*.

Haiyun He, Yepeng Liu, Ziqiao Wang, Yongyi Mao, and Yuheng Bu. 2025. Distributional information embedding: A framework for multi-bit watermarking. *arXiv preprint arXiv:2501.16558*.

Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2023. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv preprint arXiv:2310.03991*.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2023. Unbiased watermark for large language models. *arXiv preprint arXiv:2310.10669*.

Nikola Jovanović, Robin Staab, and Martin Vechev. 2024. Watermark stealing in large language models. *arXiv preprint arXiv:2402.19361*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Advances in Neural Information Processing Systems*, 36:27469–27500.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023a. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023b. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2023. Who wrote this code? watermarking for code generation. *arXiv preprint arXiv:2305.15060*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shu'ang Li, Lijie Wen, Irwin King, and Philip S Yu. 2023a. An unforgeable publicly verifiable watermark for large language models. *arXiv preprint arXiv:2307.16230*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models, 2024. *URL https://arxiv.org/abs/2310.06356*.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2023b. A semantic invariant robust watermark for large language models. *arXiv preprint arXiv:2310.06356*.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36.

Yepeng Liu and Yuheng Bu. 2024. Adaptive text watermark for large language models. *arXiv preprint arXiv:2401.13927*.

Yepeng Liu, Xuandong Zhao, Christopher Kruegel, Dawn Song, and Yuheng Bu. 2025a. In-context watermarks for large language models. *arXiv preprint arXiv:2505.16934*.

Yepeng Liu, Xuandong Zhao, Dawn Song, and Yuheng Bu. 2025b. Dataset protection via watermarked canaries in retrieval-augmented llms. *arXiv preprint arXiv:2502.10673*.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, and 1 others. 2024. Markllm: An open-source toolkit for llm watermarking. *arXiv preprint arXiv:2405.10051*.

Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. 2024. No free lunch in llm watermarking: Trade-offs in watermarking design choices. *Advances in Neural Information Processing Systems*, 37:138756–138788.

Jipeng Qiang, Shiyu Zhu, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. 2023. Natural language watermarking via paraphraser-based lexical substitution. *Artificial Intelligence*, 317:103859.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2023. A robust semantics-based watermark for large language model against paraphrasing. *arXiv preprint arXiv:2311.08721*.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. 2023. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Qwen Team. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Xiaojun Xu, Yuanshun Yao, and Yang Liu. 2024. Learning to watermark llm-generated text via reinforcement learning. *arXiv preprint arXiv:2403.10553*.

Yijie Xu, Aiwei Liu, Xuming Hu, Lijie Wen, and Hui Xiong. 2025. Mark your llm: Detecting the misuse of open-source large language models via watermarking. *arXiv preprint arXiv:2503.04636*.

Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. 2022. Tracing text provenance via context-aware lexical substitution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11613–11621.

Xin Yi, Yue Li, Shunfan Zheng, Linlin Wang, Xiaoling Wang, and Liang He. 2025. Unified attacks to large language model watermarks: spoofing and scrubbing in unauthorized knowledge distillation. *arXiv preprint arXiv:2504.17480*.

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2024. {REMARK-LLM}: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1813–1830.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. 2023a. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. 2023b. Provable robust watermarking for ai-generated text. *arXiv preprint arXiv:2306.17439*.

Xuandong Zhao, Sam Gunn, Miranda Christ, Jaiden Fairoze, Andres Fabrega, Nicholas Carlini, Sanjam Garg, Sanghyun Hong, Milad Nasr, Florian Tramer, Somesh Jha, Lei Li, Yu-Xiang Wang, and Dawn Song. 2025. Sok: Watermarking for ai-generated content. *Preprint*, arXiv:2411.18479.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. 2023c. Protecting language generation models via invisible watermarking. In *International Conference on Machine Learning*, pages 42187–42199. PMLR.

Chaoyi Zhu, Jeroen Galjaard, Pin-Yu Chen, and Lydia Y Chen. 2024. Duwak: Dual watermarks in large language models. *arXiv preprint arXiv:2403.13000*.

A Implementation Details

A.1 Derivation of the Unbiased Gradient

In Section 4.1, we formulate the construction of the green/red token list as an RL problem. We notice that the standard policy gradient is unsufficient for our watermark setting due to the inherent duel role of the mapping model.

Specifically, we denote state s as the unwatermarked text $\boldsymbol{x}^{\text{uw}}$, and a rollout $\boldsymbol{g}$, *i.e.,* a way of green/red token assignment, is sampled from the policy defined by the mapping model M_θ. The reward is defined as: $R_\theta(s, \boldsymbol{g})$. Our goal is to find θ that maximizes the following objective: $J(\theta) = \mathbb{E}_{\boldsymbol{g}\sim\pi_{\boldsymbol{\theta}}(\cdot|s)}\big[R(s, \boldsymbol{g})\big]$. The gradient $\nabla_\theta J(\theta)$ is:

$$\begin{aligned}\nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{\boldsymbol{g}\sim\pi_{\boldsymbol{\theta}}(\cdot|s)}\big[R(s, \boldsymbol{g})\big] \\ &= \nabla_\theta \int \pi_{\boldsymbol{\theta}}(\boldsymbol{g}) R_\theta(s, \boldsymbol{g})\, d\boldsymbol{g} \\ &= \int R_\theta(s, \boldsymbol{g}) \nabla_\theta \pi_{\boldsymbol{\theta}}(\boldsymbol{g})\, d\boldsymbol{g} + \int \pi_{\boldsymbol{\theta}}(\boldsymbol{g}) \nabla_\theta R_\theta(s, \boldsymbol{g})\, d\boldsymbol{g}\end{aligned} \tag{3}$$

In the last line of Eq.(3), the first term is the standard policy gradient. The second term, $\int \pi_{\boldsymbol{\theta}}(\boldsymbol{g}) \nabla_\theta R_\theta(s, \boldsymbol{g})\, d\boldsymbol{g} = \mathbb{E}\left[\nabla_\theta R_\theta(s, \boldsymbol{g})\right]$, is specific to our setting. The reward R is composed of multiple detection scores. And as shown in Eq.(1), detection score has a non-zero gradient and thus the second term must be accounted for during optimization.

In our implementation, we take this gradient term into account. We also apply a ablation study in Appendix B.1 to figure out the effectiveness of adding this gradient term.

A.2 Baselines

To ensure a fair comparison, we tune the hyperparameters of all baseline methods such that the generated watermarked texts exhibit comparable perplexity levels, reflecting similar text quality. Specifically, we set $\delta = 3.0$ for KGW and UNIGRAM; $\alpha = 2.0$, $\delta_0 = 0.1$, and $\delta = 0.13$ for ADAPTIVE; and $ratio = 0.06$ for POSTMARK. All other hyperparameters not listed above are kept at their default values as specified in the respective original implementations.

A.3 RL Training Parameters

We optimize the parameters of the mapping model using the GRPO algorithm. The hyperparameter settings related to RL training are summarized in Table 3. We run our experiments on four B200 GPUs.

	Value
# of training steps	1000
Learning rate	5e−5
Batch size	16
# of mini batches	2
# of rollouts	8
Clipping coefficient	0.2
Max gradient norm	0.5
beta	0.04

Table 3: RL training related parameters.

A.4 Attack Prompts and Detailed Implementation

Paraphrase attack. Given an input text, we follow the same paraphrasing prompt as in An et al. (2025) to generate paraphrased variants that preserve the semantics of the original text.

Sentiment spoofing attack. The sentiment spoofing attack consists of three steps: ① Detect the original sentiment of the input text using the prompt shown in Fig. 4. ② Determine the target sentiment: if the detected sentiment is non-neutral, we flip it; otherwise, we randomly select either positive or negative. ③ Update the prompt shown in Fig. 5 with the target sentiment and generate the sentiment-flipped variant.

Hate-speech spoofing attack. Because modern LLMs are trained with strong safety alignment, directly prompting them to produce hate speech is challenging. Following the procedure in An et al. (2025), we first generate a list of entity names associated with discriminatory language. During training, to create a hate-speech–attacked variant, we prompt the LLM to produce 2–5 short hate-speech sentences containing placeholders, randomly select one entity name from the list to replace each placeholder, concatenate the sentences, and then randomly insert them into the original text. The prompts used to generate hate-speech entities and short sentences are identical to those in An et al. (2025).

A.5 Computational Overhead Analysis

We provide an analysis of the computational overhead in the RL training stage and practical adoption, respectively.

First, in the RL training stage, the most time-consuming step is applying different attacks to both

original and watermarked texts. Notice that this only happens in training, and it's not part of deployment. However, when there are enough computing resources, the attacks can be parallelized to further reduce training time. With our current setup (batch size 16, rollout group size 8, max-length 500), one training step takes about 5 minutes on 4xB200 GPUs.

In deployment, most time will be spent during inference. To inject watermarks into a text, our method requires (1) a single forward pass through the mapping model to construct the green/red list, and (2) standard next-token generation of the backbone LLM with perturbed logits. This cost is comparable to existing semantic-aware watermarking methods such as Adaptive(Liu and Bu, 2024) and Contrastive(An et al., 2025), which also perform a forward pass of a mapping model before generation. Because our method inherits the global green/red list from Contrastive, the number of forward passes needed is reduced to one time for each input text.

B Additional Results

B.1 Ablation on Reward Gradient

Reward gradient	ROC-AUC (%)			
	Detectability ↑	Paraphrased ↑	Sentiment Spoof↓	Hate Speech Spoof ↓
w/o	96.35	78.64	80.07	59.85
w/	97.30	95.11	31.81	1.31

Table 4: Performance comparison between models train using unbiased gradient or not.

To evaluate the effectiveness of incorporating the reward gradient into the optimization, we train two models under identical settings—one with and one without the reward gradient term.

Table 4 reports the performance comparison between the two models. The model trained with the unbiased gradient (which explicitly includes the reward gradient term) consistently outperforms the one without it across all four criteria. These results demonstrate that incorporating the reward gradient yields more stable optimization and leads to a better overall balance among detectability, robustness, and security.

B.2 Ablation on Reward Components

To study the contribution of individual reward components, we remove each component, $i.e.$, detectability, robustness, or security, from the complete reward while keeping all other settings unchanged.

Anchored value	ROC-AUC (%)			
	Detectability ↑	Paraphrased ↑	Sentiment Spoof↓	Hate Speech Spoof ↓
0.3	96.94	96.41	37.01	0.57
0.4	97.08	97.34	37.65	0.49
0.5	97.30	95.11	31.81	1.31
0.6	97.50	97.64	17.11	1.46
0.7	97.98	97.71	13.89	0.94

Table 5: Sensitivity analysis on the anchored value.

Table 6 reports the results. Removing the detectability term slightly increases the overall AUC (by approximately 0.3%), but leads to a noticeable drop in detectability. Removing the robustness term substantially degrades both detectability and paraphrase robustness. Removing the security term results in significantly worse performance under spoofing attacks. Overall, the complete reward achieves the best balance across all criteria, confirming that all three components are necessary for the desired trade-off.

B.3 Sensitivity Analyssis on the Anchored Value

We conduct a sensitivity analysis on the anchored value, which is the hyperparameter controlling the desired detection score of unwatermarked text. We vary the anchor value from 0.3 to 0.7, retrain the model under each setting while keeping all other configurations fixed, and then evaluate the resulting policies.

As shown in Table 5, our method is robust to the choice of anchor value. Detectability, paraphrase robustness, and hate-speech-spoof security remain at similar levels across all settings. The AUC for the sentiment-spoofed metric decreases as the anchor value increases, since a larger anchor value encourages higher detection scores for unwatermarked texts, thereby making sentiment-spoofed texts easier to differentiate from unwatermarked ones. This leads to improved security performance.

In conclusion, these results show that our method is stable across a wide range of anchor values, and reasonable choices within 0.3–0.7 all yield strong and consistent performance. However, extremely high or low anchor values do not guarantee improvements and may degrade overall performance due to inherent tradeoffs among the criteria. For example, if the anchor value is set too high (e.g., 0.99), it becomes difficult for watermarked or para-

Setting	ROC-AUC (%)				
	Detectability ↑	Paraphrased ↑	Sentiment Spoofed ↓	Hate Speech Spoofed ↓	Overall AUC ↑
w/o detectability	94.69	97.12	31.22	0.25	90.09
w/o robustness	90.58	77.13	27.94	10.70	82.27
w/o security	96.96	96.77	45.62	2.62	86.37
complete reward	97.30	95.11	31.81	1.31	89.82

Table 6: Ablation study on reward components. Arrows indicate whether higher or lower values are preferred.

Method	ROC-AUC (%)				Overall AUC ↑
	Detectability ↑	Robustness ↑	Security ↓		
		Paraphrase	Sentiment Spoof	Hate Speech Spoof	
Llama-3.1-8B-Instruct					
PostMark	99.98(-0.01)	90.93(+1.90)	96.21(+2.14)	99.80(-0.07)	48.73
Contrastive	98.04(+0.02)	72.02(+0.05)	34.80(+0.12)	34.11(-0.27)	75.29
Ours	96.77(-0.53)	97.09(+1.98)	33.81(+2.00)	0.83(-0.48)	89.81

Table 7: Performance of watermarking methods on a larger evaluation scale. We report the performance on 500 texts from the C4 dataset.

phrased texts to achieve detection scores higher than those of unwatermarked texts. This limits the model's ability to separate these texts appropriately, leading to degraded detectability and reduced paraphrase robustness.

B.4 Performance on a Larger Evaluation Scale

We expanded the evaluation from 200 to 500 texts. As shown in Table 7, the values in parentheses denote the change relative to the results on 200 texts. On this larger test set, our method continues to outperform the baselines by a substantial margin on the overall AUC, demonstrating that the trained policy generalizes well and remains effective at a larger evaluation scale.

B.5 Text Quality

To provide a more comprehensive assessment of text quality, we adopt an LLM-as-judge evaluation following the procedure used in Contrastive (An et al., 2025). Specifically, we provide GPT-4o with both the unwatermarked and watermarked versions of each text, and ask it to independently rate (1) the text quality of the watermarked output (e.g., coherence, fluency, grammatical correctness), and (2) the semantic relevance between the unwatermarked and watermarked texts. GPT-4o assigns scores on a 1–3 scale, where 3 indicates the best quality.

As shown in Table 8, our method achieves the second-best text-quality score and the best relevance score among all baselines. This indicates that our method preserves semantic fidelity in unwatermarked text and maintains high-quality generation.

Method	Text Quality	Relevance
KGW	2.830	2.412
Unigram	2.719	2.245
Adaptive	2.750	2.163
PostMark	2.230	1.811
Contrastive	2.821	2.378
Ours	2.824	2.648

Table 8: LLM-as-judge evaluation of text quality and semantic relevance. Scores are on a 1–3 scale, where higher is better.

B.6 Additional Attack Evaluation

To expand attack evaluation, we evaluate our model under three additional attacks: ① a learnable spoofing attack that infers the green/red lists, ② paraphrasing using a different tool, and ③ a translation-based perturbation. We compare our method with two baselines with the highest overall AUC in Table 1, which are PostMark and Contrastive.

For the learnable spoofing attack, following Adaptive(Liu and Bu, 2024), we strengthen the attacker by repeatedly generating watermarked variants of the same input, ensuring the green/red lists remain fixed. Specifically, we generate 5k water-

marked texts for Contrastive and our method, and 1k for PostMark due to API cost, then infer green lists via token-frequency analysis. Because green list sizes differ across methods, we report decryption AUC rather than top-k recall. As shown in Table 9, our method achieves the lowest decryption AUC, indicating the strongest resistance to this learnable spoofing attack.

For the paraphrase attack, we use a new paraphrasing prompt focusing on different linguistic dimensions and a new model (Gemini-2.5-Flash) to generate paraphrases. For the translation attack, we prompt GPT-4o to first translate English to Chinese, then translate back to English. For both attacks, a higher AUC indicates better robustness. As shown in Table 10, our method consistently outperforms both baselines across these unseen attack types.

Method	Decryption AUC ↓
PostMark	0.830
Contrastive	0.543
Ours	0.502

Table 9: Resistance to learnable spoofing attacks. Lower decryption AUC indicates stronger resistance.

Method	**ROC-AUC (%)**	
	Paraphrase ↑	Translation ↑
PostMark	88.41	95.70
Contrastive	71.06	89.47
Ours	99.23	94.35

Table 10: Robustness under additional paraphrasing and translation-based attacks. Higher AUC indicates better robustness.

C Use of AI Assistants

AI assistant is used to assist coding and proofreading.

```
Please act as a judge and determine the sentiment of the following text.
    ↪ Your task is to assess whether the sentiment is positive, negative,
    ↪ or neutral based on the overall tone and emotion conveyed in the
    ↪ text. Consider factors like word choice, emotional context, and any
    ↪ implied feelings. The sentiment can only be chosen from 'positive',
    ↪ 'negative', and 'neutral'.
Begin your evaluation by providing a short explanation for your judgment.
    ↪ After providing your explanation, please indicate the sentiment by
    ↪ strictly following this format: "[[sentiment]]", for example:
    ↪ "Sentiment: [[positive]]".
```

Figure 4: Prompt used for LLM as sentiment judge.

```
### Task:
Modify the given text to **clearly shift its overall sentiment** to
    ↪ **{modified_sentiment}**. Your goal is to transform the emotional
    ↪ tone of the text by adjusting words, phrases, and even facts as
    ↪ needed. Prioritize changing emotionally charged words, evaluative
    ↪ adjectives, and sentiment-bearing phrases. Focus on changing or
    ↪ replacing specific words or short phrases to achieve the sentiment
    ↪ shift. Minor sentence restructuring is allowed only if it
    ↪ significantly helps express the new sentiment more clearly.

### Response Format:
- The LLM should explicitly state the **new sentiment of the modified
    ↪ text**, and provide a **brief explanation** before giving the
    ↪ modified text.
- The response must strictly follow this format:

```
[MODIFIED_SENTIMENT] <modified_sentiment> [/MODIFIED_SENTIMENT]
[EXPLANATION] <modification_plan> [/EXPLANATION]
[MODIFIED_TEXT] <modified_text> [/MODIFIED_TEXT]
```
```

Figure 5: Prompt used for sentiment spoofing attack.

```
Paraphrase the following text while preserving its original meaning. Ensure
    ↪ that the output meets the following criteria:

1. Preserves Meaning: The paraphrase should convey the same core idea
    ↪ without omitting or distorting information.
2. Fluency and Grammar: The paraphrase must be natural, grammatically
    ↪ correct, and well-structured.
3. Appropriate Length: Maintain a similar length unless a slight adjustment
    ↪ improves clarity.
4. Consistency with Context: Retain the original tone and formality (e.g.,
    ↪ academic, casual, professional).
5. Minimal Redundancy: Avoid unnecessary repetition while keeping essential
    ↪ details.
6. Retains Nuances: Preserve connotations, implied meanings, and idiomatic
    ↪ expressions where appropriate.

Just provide the paraphrased version of the text, without any introductory
    ↪ or concluding phrases.
```

Figure 6: Prompt used for semantic-equivalent paraphrase.