# Breaking Distortion-free Watermarks in Large Language Models

**Shayleen Reynolds**[1], **Hengzhi He**[2], **Dung Daniel Ngo**[3], **Saheed Obitayo**[3], **Niccolò Dalmasso**[3], **Guang Cheng**[2], **Vamsi K. Potluru**[3], and **Manuela Veloso**[3]

[1]NTT DATA.* Email: shayleen.reynolds.d@gmail.com
[2]UCLA. Email: hengzhihe@g.ucla.edu, guangcheng@ucla.edu
[2]JPMorganChase AI Research. Email: {tuandung.ngo, niccolo.dalmasso, saheed.obitayo, vamsi.k.potluru, manuela.veloso}@jpmchase.com

## Abstract

In recent years, LLM watermarking has emerged as an attractive safeguard against AI-generated content, with promising applications in many real-world domains. However, there are growing concerns that the current LLM watermarking schemes are vulnerable to expert adversaries wishing to reverse-engineer the watermarking mechanisms. Prior work in 'breaking' or 'stealing' LLM watermarks mainly focuses on the distribution-modifying algorithm of Kirchenbauer et al. [13], which perturbs the logit vector before sampling. In this work, we focus on reverse-engineering the other prominent LLM watermarking scheme, distortion-free watermarking [15], which preserves the underlying token distribution by using a hidden watermarking key sequence. We demonstrate that, even under a more sophisticated watermarking scheme, it is possible to *compromise* the LLM and carry out a *spoofing* attack, i.e., generate a large number of (potentially harmful) texts that can be attributed to the original watermarked LLM. Specifically, we propose using adaptive prompting and a sorting-based algorithm to accurately recover the underlying secret key for watermarking the LLM. Our empirical findings on LLAMA-3.1-8B-Instruct, Mistral-7B-Instruct, Gemma-7b, and OPT-125M challenge the current theoretical claims on the robustness and usability of the distortion-free watermarking techniques.

## 1 Introduction

Recent advances in generative models have significantly improved their capabilities and applicability across various real-world domains. Notably, models like ChatGPT [21] and other large language models (LLMs) can now generate text closely resembling human-written content. However, as both businesses and individuals have rapidly adopted generative models, there is a growing concern within the research community about their potential for malicious use. To address this issue, a growing body of research around *watermarking* LLM-generated text has emerged [1, 13, 15, 19, 24, 31]. The primary strategy in this line of research involves embedding a *hidden* signal (i.e., a secret watermark key) within the generated text, which any third party with knowledge of the key can reliably detect.

While these watermarking techniques offer reliable and robust statistical guarantees to verify LLM-generated texts, they still fall short in addressing the potential attack models posed by malicious actors [7, 8, 12, 23, 28, 33]. Previous research on LLM watermarking often focuses on robustness against common attacks, such as deletion, insertion, and substitution, to simulate the behavior

*Work done while at AI Research, JPMorganChase.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

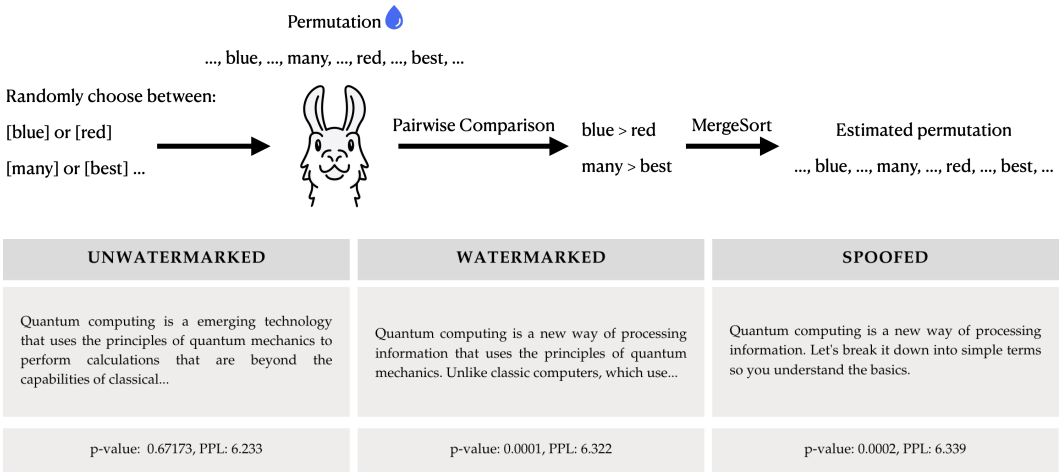


Figure 1: Overview of our watermark stealing framework. **Top**: Illustrative example of Algorithm 4 to estimate the secret permutation over the LLM's vocabulary. **Bottom**: Outputs using Llama-3.1-8B-Instruct **(left)**: without watermark, **(middle)**: with watermark, **(right)**: spoofing attack. The $p$-value reflects the watermark detection confidence, and the perplexity (PPL) measures text quality.

of users attempting to evade content detectors. For instance, a student might slightly modify a machine-generated essay by altering a few sentences to avoid detection by their professor. However, a determined adversary could go further by reverse-engineering the watermarking scheme. By repeatedly querying the API of the watermarked LLM, they could *steal* the watermark by approximating the hidden secret key. Once estimated, the most significant threat is *spoofing*, where an attacker generates (potentially harmful) text that appears to be watermarked. Being able to generate large volumes of *spoofed* content with minimal computational effort not only undermines the intended purpose of a watermark but is also a reputational risk for the LLM providers, whose model could have been falsely attributed to harmful or incorrect content.

Prior work on *watermark stealing* primarily studies the distribution-modifying algorithm by Kirchenbauer et al. [13] and its variants. In contrast, our focus is on the other prominent watermarking scheme by Kuditipudi et al. [15], which is *distortion-free*, i.e., the watermark does not change the underlying token distribution. A significant difference between the two watermarking techniques is that Kuditipudi et al. [15] use a randomized watermark key, creating a correlation between the LLM-generated text and this secret key. During detection, a third party with this secret watermark key can efficiently check for the correlation and verify whether the text is watermarked. Furthermore, prior work that attempted to break this distortion-free watermark specifically focus on the exponential minimum sampling variant. In this work, we complete the line of work on stealing Kuditipudi et al. [15]'s watermark by investigating the inverse transform sampling (ITS) variant (Section 2.1).

Considering this approach, we propose a sorting-based algorithm to accurately estimate the secret watermark key and enable *spoofing* attacks with only a few samples from the watermarked LLM. Our spoofed outputs both (i) pass the original watermark detection test, and (ii) maintain similar text quality (measured with perplexity score and cosine similarity) compared to the non-watermarked and watermarked outputs from the base LLM. Notably, we observe successful spoofing attempts even when the attacker has a partial knowledge of the secret key and permutation used for watermarking.

**Overview of Paper.** We introduce a watermark stealing framework for the distortion-free LLM watermarking technique from Kuditipudi et al. [15]. Specifically, in this work we focus on the watermark approach using ITS. At a high level, the decoder used in ITS has two main components: a secret key sequence of uniformly random variables and a permutation over the entire vocabulary. With a sequence of well-crafted prompts, our algorithm can probe the underlying watermark to accurately learn the permutation over the vocabulary used to generate watermarked text. Once the permutation is recovered, we repeatedly query the watermarked LLM and obtain an accurate estimate of the secret key sequence by iteratively narrowing down the confidence interval with each observed

sample. After both components of the watermark decoder have been learned, we can successfully perform *spoofing* attacks on the watermarked LLM, i.e., generate texts that appear to be watermarked. Overall, we make the following contributions:

- We provide a framework that accurately estimates underlying parameters of the distortion-free watermarking algorithm [15] under different threat models. Specifically, we target the watermark algorithm based on ITS, which uses a sequence of random secret keys and a permutation over the vocabulary to watermark the LLM.
- With this secret watermark key estimation, we demonstrate a spoofing attack with a high success rate on four LLMs (Llama-3.1-8B-Instruct [9], Mistral-7B-Instruct [11], Gemma-7b [27], and OPT-125M [32]). In conjunction with prior work in watermark stealing, these results highlight the need for refining SOTA watermarking techniques for language models.

## 2 Preliminary

In this section, we provide relevant background on the inverse-transform mapping-based watermarking method proposed in Kuditipudi et al. [15], our notations, and a concrete set of threat models. In the following, we use $x$ to denote a sequence of tokens, $x_i \in \mathcal{V}$ is the $i$-th token in the sequence, and $\mathcal{V}$ is the vocabulary. For $n \in \mathbb{N}^+$, we write $[n]$ to denote the set $\{1, \cdots, n\}$.

### 2.1 Watermark Generation and Interaction Protocol

Let $p : \mathcal{V}^* \to \Delta(\mathcal{V})$ be an auto-regressive *language model* (LM) that maps a string of arbitrary length to a probability distribution over the vocabulary $\Delta(\mathcal{V})$. Given a prefix $x \in \mathcal{V}^*$, we denote the conditional probability distribution over the next token by $p(\cdot|x)$. Let $\Xi$ represent the space of watermark key elements, and let $\xi \in \Xi^n$ be a secret key sequence. For simplicity, we assume that each element $\xi_i \in [0, 1] : i \in [n]$. Let $\pi : [|\mathcal{V}|] \to \mathcal{V}$ be a fixed bijective permutation mapping from integers to tokens, where $\pi(i)$ is the $i$-th token in the vocabulary sorted according to the permutation $\pi$. The watermarking mechanism is based on ITS as follows; for details, see Algorithm 1 (Appendix C).

1. **Key and Mechanism Sharing**: The LM provider shares a secret key sequence $\xi = (\xi_1, \xi_2, \ldots, \xi_n) \in \Xi^n$ and a bijective permutation function $\pi$ with the detector.
2. **User Prompt**: The user provides a prompt $x \in \mathcal{V}^*$ to the LM provider.
3. **Watermarked Text Generation**: The LM provider generates watermarked text $Y \in \mathcal{V}^*$ by applying the following procedure at each token generation step:
   (a) Compute the CDF of the LM's probability distribution over the vocabulary: $C_k = \sum_{i=1}^{k} p(\pi(i)|x), \quad \forall k \in [|\mathcal{V}|]$.
   (b) Retrieve the corresponding key element for the $t$-th token generation: $\zeta = \xi_{t \mod n}$.
   (c) Identify the smallest index $k$ such that $C_k \geq \zeta$.
   (d) Select the token $\pi(k)$ as the next token in $Y$.
   (e) Append the selected token to the current prefix and update it: $x \leftarrow x \oplus \pi(k)$.
   (f) Repeat until the desired text length is reached or another termination condition is met.

### 2.2 Watermark Detection

We adopt the watermark detection using permutation test to compute $p$-value similar to Kuditipudi et al. [15]. If the test returns a small $p$-value, then the text is likely to be watermarked; otherwise, the text is likely not watermarked. The details are summarized in Algorithm 2 (Appendix C).

### 2.3 Attacker Model

**Attacker's Objective and Motivation.** Our work focuses on the 'spoofing' attack, which aims to generate harmful or incorrect outputs that carry the original LM provider's watermark. For example, an attacker can 'spoof' the watermarked LM, generate fake news or defamatory remarks, and post them on social media. The attacker can damage the LM provider and their model's reputation by claiming that the LM provider's model generated these harmful texts.

**Attacker's Capabilities.** Similar to [12, 23], we make the following assumptions on the attacker's capabilities. The attacker has black-box access to the complete generation of the watermarked LM and is aware of the existence of the watermark used by the LM provider. The attacker aims to use a small number of queries to the watermarked LM to build an estimation of the underlying watermarking scheme parameterized by the secret key $\xi$ and permutation $\pi$. We further assume that the detection API is available to the public. The attacker can query the detection API and obtain the watermark confidence score in terms of $p$-value. This assumption allows the attacker to verify the effectiveness of their 'spoofing' attack. Following prior work [18, 22, 23] and enabled by OpenAI's API, we assume that the top tokens at each position and their probabilities are returned to the attacker.

## 3 Methodology: Breaking Distortion-free Watermarks

In this section, we explore three different regimes for breaking the distortion-free watermark, focusing on secret key estimation and recovering the token permutation used in the watermarking process.

**Threat Models.** The goal of the attack is to reverse-engineer both the permutation $\pi$ and the secret keys $\{\xi_i\}$ by querying a LM with a variety of carefully designed prompts. We assume that (i) the model consistently employs the same secret key sequence and permutation for text generation and (ii) the adversary can interact with the model through these crafted prompt queries to extract information about its watermarking process [2]. Concretely, we consider the following three threat models:

1. **Either $\pi$ is known or $\{\xi_i\}$ is known:** The attacker either knows the secret key sequence $\{\xi_i\}$ or the permutation $\pi$. The goal is to infer the permutation or the secret key sequence, respectively. The analysis for these two models is in Appendix D.1.
2. **Both $\{\xi_i\}$ and $\pi$ are Unknown:** The most realistic and challenging model where the attacker has no knowledge of the secret key sequence $\{\xi_i\}$ and the permutation $\pi$. Under this model, the attacker must simultaneously recover both parameters solely from the observed watermarked outputs. The analysis for this model is in Section 3.1 below.

### 3.1 Unknown secret key sequence $\{\xi_i\}$ and permutation $\pi$

In general, if both the random key sequence $\{\xi_i\}$ and the permutation $\pi$ are unknown, it is impossible to determine them uniquely. An interesting symmetry emerges in the parameterization of the watermarking process. Specifically, replacing each $\xi_i$ with its complement $1-\xi_i$ and simultaneously reversing the permutation $\pi$ leaves the watermarking process unchanged. Formally, given a permutation $\pi$, define its reverse counterpart $\pi'$ as $\pi' = \text{reverse}(\pi)$. Then the transformation $\xi_i \to 1-\xi_i, \pi \to \pi'$ results in the same watermarking behavior almost surely. Formally, we show in Theorem 3.1 that this is the only possible alternative parametrization. The proof is in Appendix B.

**Theorem 3.1.** *Given secret keys $\xi, \hat{\xi} \in [0,1]$, permutations $\pi, \hat{\pi} : [|\mathcal{V}|] \to \mathcal{V}$ and a probability distribution $p \in \Delta(\mathcal{V})$. Let $S(p, \xi, \pi)$ denote the token selection function that outputs the watermarked token in Algorithm 1. Suppose $S\big(p, \xi, \pi\big) = S\big(p, \widehat{\xi}, \widehat{\pi}\big)$ for almost every $p \in \Delta(\mathcal{V})$. Then exactly one of the following must hold: either (i) $\pi = \widehat{\pi}$ and $\xi = \widehat{\xi}$ or (ii) $\pi = \text{reverse}(\widehat{\pi})$ and $\xi = 1 - \widehat{\xi}$.*

We propose a method to learn *one* of these two equivalent parameterizations. Our approach relies on constructing queries that force the LM to choose randomly between two candidate tokens, allowing us to record the relative order between any pair of tokens. With these pairwise ordering, a comparison-based sorting algorithm can recover the global ordering $\pi$ (or its reverse) in $O(|\mathcal{V}| \log |\mathcal{V}|)$ queries. Specifically, we define a query interface, `QueryLLM(a, b)`, with two tokens $a$ and $b$ as input. With carefully designed prompts similar to Chen et al. [3], this interface ensures that the model considers only the two candidate tokens and assigns them equal probabilities. Under the ITS watermark, the model's selection between $a$ and $b$ is governed only by the hidden permutation $\pi$ and the secret key. In our scheme, if the model outputs token $b$, we interpret this as $a < b$; otherwise, we interpret it as $b < a$. With this ordering definition, we apply a comparison-based sorting algorithm (e.g., Merge Sort) to recover an ordered sequence of tokens. The resulting order will correspond either to $\pi$ or its

[2] Kuditipudi et al. [15] suggested using a single random permutation in practice to reduce overhead in both watermark generation and detection.

reverse, which are equally useful. This result reduces the problem to the case where $\pi$ is known. The algorithm details and analysis are summarized in Algorithm 7 (Appendix F).

## 4 Experimental Evaluation

To validate the effectiveness of our watermark spoofing methodology, we conduct a series of experiments under the most challenging threat model where the attacker has no prior knowledge of the watermark's secret permutation $\pi$ or secret key $\{\xi\}$ (section 3.1 and Appendix F). After inferring the permuted vocabulary, the problem reduces to the case where the permutation is known (or partially known), but the secret key still needs to be estimated (section D.2 and appendix E). Our evaluation addresses three key questions: **(Q1)** Can an attacker successfully carry out spoofing attacks on the Kuditipudi et al. [15]'s watermark? **(Q2)** Do the different LMs affect the spoofing result? and **(Q3)** How much computational resources does the attacker need to spoof successfully?

We test our approach using the OpenGen benchmark prompts [14] and the LFQA dataset [6], on four different LLMs: Llama-3.1-8B-Instruct [2], Mistral-7B-Instruct [11], Gemma-7b [27], and OPT-125M [32]. We employ three evaluation metrics: the watermark detection $p$-value from Algorithm 2, cosine similarity using nomic embed models [20] to calculate semantic similarity, and the output perplexity (PPL) to measure text fluency and quality. A successful spoofing attack should generate texts with $p$-value less than $\alpha = 0.05$ (Appendix C), PPL on par with ordinary model outputs and cosine similarity score close to 1 [16]. For additional experimental detail and results, see Appendix G.

**(Q1) Successful spoofing attacks on Kuditipudi et al. [15]'s watermark.** Our proposed algorithms can successfully spoof the ITS-based watermark by Kuditipudi et al. [15]. First, we evaluate the spoofing results from the estimated permutation $\pi$ and secret-key $\xi_i$. We follow the procedure outlined in Section 3 to generate new samples using the recovered permutation and secret key values (or subsets in the partial order scenarios). Our experimental setup compares three types of generated text: **(1)** watermarked text from the watermarked LLM using the true secret key and permutation – a baseline for expected detector behavior and text quality, **(2)** non-watermarked text from the same model (without watermarking mechanism) – a control LLM output, and **(3) spoofed text** produced by our attack using the recovered permutation and secret key. We generate 100 samples in each category with 50 tokens per sample. The results of this evaluation are summarized in Table 1 and Figure 2.

Table 1 shows the median $p$-values for 100 samples of each generation type across all language models. Watermarked (WM) text yields low $p$-values ($p < 0.05$), and non-watermarked (Non-WM) text shows high values ($p > 0.05$), as expected. Across all LLMs, once the attacker has learned the permutation of the first $50\%$ tokens in the tokenizer, most spoofed samples are below the detection threshold. Detection starts to fail when only the permutation of the first $25\%$ tokens are learned, with many spoofed samples detected as non-watermarked. We conclude that our method reliably passes the detector at moderate-to-high permutation recovery levels ($50 - 100\%$) across all models.

Additionally, we examine the quality of the spoofed text to ensure that our attack does not significantly degrade the coherence of the generated content. We evaluate the text quality using PPL and cosine similarity, which are viable proxies for human preference [5, 35]. In Table 1 and Figure 2, the spoofed texts (Spoof @ $50\%$ and Spoof @ $25\%$) have almost the same perplexity distribution as the watermarked text. With a fully learned permutation, the spoofed text's perplexity is slightly higher on average than genuine watermarked outputs across all LLMs. In Appendix G, we provide examples of texts generated by our spoofing attack, the watermarked LLM's generated output and the

Table 1: Comparison of baseline and spoofed outputs on OpenGen dataset. For detection, we report the median $p$-value (p-val). For text quality, we report perplexity (PPL) and cosine similarity (co-sim).

| **Model** | **Non-WM** | | **WM** | | | **Spoof @ $50\%$** | | | **Spoof @ $25\%$** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | p-val | PPL | p-val | PPL | co-sim | p-val | PPL | co-sim | p-val | PPL | co-sim |
| Llama-3.1-8B | 0.48 | 5.13 | 1.0e-04 | 16.32 | 0.871 | 1.0e-04 | 26.51 | 0.871 | 5.5e-04 | 24.09 | 0.859 |
| Mistral-7B | 0.39 | 8.91 | 1.0e-04 | 51.49 | 0.861 | 1.0e-04 | 56.26 | 0.864 | 1.0e-04 | 68.48 | 0.863 |
| Gemma-7B | 0.45 | 9.89 | 1.0e-04 | 59.02 | 0.85 | 1.0e-04 | 81.41 | 0.837 | 1.0e-04 | 86.35 | 0.829 |
| OPT-125M | 0.54 | 8.77 | 1.0e-04 | 132.46 | 0.836 | 1.0e-04 | 130.5 | 0.834 | 1.0e-04 | 133.69 | 0.845 |

non-watermarked output using the same prompt. Hence, we conclude that humans would be unlikely to notice any difference in quality, and the outputs are coherent and on-topic given the prompts.

**(Q2) Larger models impact spoofing quality.** We observe that Gemma-7B underperforms relative to other models in both spoofing detection success and output perplexity. In contrast, Llama-3.1-8B-Instruct shows strong spoofing performance due to its highly consistent token ranking, low-entropy output distributions, and robust response to prompt-based token comparisons. We found that smaller model like OPT-125M achieve strong spoofing performance, with spoofed outputs consistently passing the watermark detection test. We do see these smaller models produce text with higher perplexity, reflecting their limited language fluency and smaller effective vocabulary. While spoofing is statistically easier, the generated text lacks the quality of outputs from larger LLMs.

**(Q3) Query-efficient spoofing attack compared to prior work.** An advantage of our framework is its sample efficiency. By directly querying the model with well-crafted comparisons and leveraging the structure of the ITS watermarking scheme, we require a low number of queries to steal the watermark. Although the exact number of queries depends on the vocabulary size and the model's consistency, it is typically in order of only tens of thousands for a full reconstruction. This improved efficiency translates to real-world cost savings for the adversary. If the target model is accessed via a paid API, an attacker's job requiring a million queries could be cost-prohibitive [25]. For comparison, Jovanović et al. [12] reported using 30k queries with 800k tokens to learn token distribution statistics for breaking a watermark, while our permutation recovery requires significantly fewer queries (<100) and tokens (<50). Our proposed approach translates to a significant reduction in attack cost and time.

In summary, our experimental evaluation demonstrates that an attacker can (i) **reverse-engineer** the secret watermark parameters of a distortion-free watermarked LLM and (ii) use these to **generate a large quantity of spoofed text** that **fools the watermark detector** and remains high-quality.

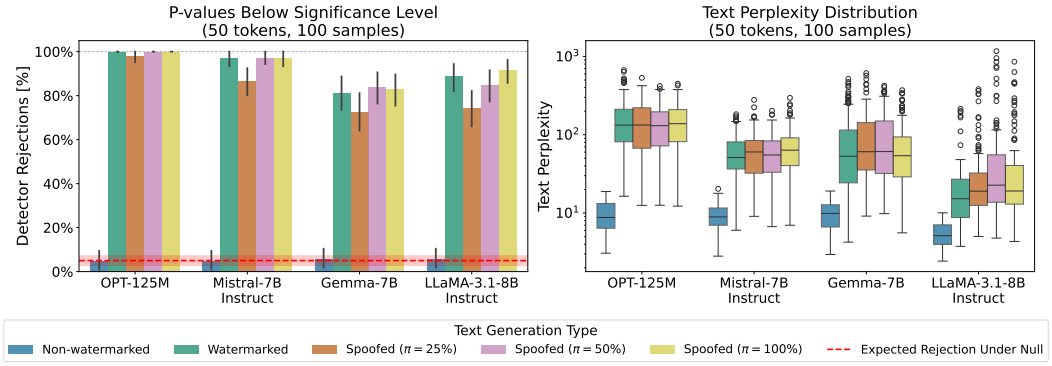


Figure 2: *Left*: Percentage of $p$-values below the significance level ($\alpha = 0.05$) for detection (Algorithm 2). Spoofed samples have similar rejection rates as watermarked samples. *Right:* Distribution of text perplexity. Spoofed samples are not significant different in perplexity compared to the watermarked text. All experiments are across LLMs and known permutation $\pi$ proportions.

## 5 Conclusion and Future Work

This paper presents a novel attack that spoofs Kuditipudi et al. [15]'s LLM watermarking scheme by using adaptive prompting and a sorting-based query strategy. This allows us to generate 'spoofed' high-quality texts that are statistically indistinguishable from genuine watermarked text. Our attack succeeds even when both the watermark key and the permutation are not known a-priori, with even a partial recovery of the permutation being sufficient. Our results challenge the security of distortion-free watermarks and complement existing watermark attacks, showing that the alternative ITS watermark scheme is also vulnerable and urging caution in using LLM watermarks as a line of defense. Future works include (i) developing watermarking schemes that are resilient to key recovery (e.g., dynamic key rotation, randomness during generation), (ii) testing attacks under stricter constraints (e.g., limited queries or partial detector access), (iii) scaling to larger models and (iv) exploring how spoofing interacts with other provenance mechanisms like model fingerprinting [29].

## Acknowledgments and Disclosure of Funding

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates ("JP Morgan") and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

## References

[1] S. Aaronson. 'Reform' AI Alignment with Scott Aaronson. AXRP - the AI X-risk Research Podcast, 2023. URL https://axrp.net/episode/2023/04/11/episode-20-reform-ai-alignment-scott-aaronson.html.

[2] M. AI. Llama 3.1 8b instruct, 2024. URL https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct.

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

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

[5] C.-H. Chiang and H. yi Lee. Can large language models be an alternative to human evaluations?, 2023. URL https://arxiv.org/abs/2305.01937.

[6] A. Fan, Y. Jernite, E. Perez, D. Grangier, J. Weston, and M. Auli. Eli5: Long form question answering, 2019. URL https://arxiv.org/abs/1907.09190.

[7] T. Gloaguen, N. Jovanović, R. Staab, and M. Vechev. Black-box detection of language model watermarks. *arXiv preprint arXiv:2405.20777*, 2024.

[8] T. Gloaguen, N. Jovanović, R. Staab, and M. Vechev. Discovering clues of spoofed lm watermarks. *arXiv preprint arXiv:2410.02693*, 2024.

[9] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan, et al. The llama 3 herd of models, 2024.

[10] C. Gu, X. L. Li, P. Liang, and T. Hashimoto. On the learnability of watermarks for language models, 2024. URL https://arxiv.org/abs/2312.04469.

[11] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. L. Scao, T. Lavril, T. Wang, T. Lacroix, and W. E. Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

[12] N. Jovanović, R. Staab, and M. Vechev. Watermark stealing in large language models, 2024. URL https://arxiv.org/abs/2402.19361.

[13] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. *arXiv preprint arXiv:2301.10226*, 2023.

[14] K. Krishna, Y. Song, M. Karpinska, J. F. Wieting, and M. Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023. URL https://openreview.net/forum?id=WbFhFvjjKj.

[15] R. Kuditipudi, J. Thickstun, T. Hashimoto, and P. Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=FpaCL1MO2C.

[16] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan. Cosine similarity to determine similarity measure: Study case in online essay assessment. In *2016 4th International Conference on Cyber and IT Service Management*, pages 1–6, 2016. doi: 10.1109/CITSM.2016.7577578.

[17] A. Liu, L. Pan, Y. Lu, J. Li, X. Hu, X. Zhang, L. Wen, I. King, H. Xiong, and P. Yu. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36, 2024.

[18] A. Naseh, K. Krishna, M. Iyyer, and A. Houmansadr. Stealing the decoding algorithms of language models. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, CCS '23, page 1835–1849. ACM, Nov. 2023. doi: 10.1145/3576915.3616652. URL http://dx.doi.org/10.1145/3576915.3616652.

[19] K. Ning, J. Chen, Q. Zhong, T. Zhang, Y. Wang, W. Li, Y. Zhang, W. Zhang, and Z. Zheng. Mcgmark: An encodable and robust online watermark for llm-generated malicious code. *arXiv preprint arXiv:2408.01354*, 2024.

[20] Z. Nussbaum and B. Duderstadt. Training sparse mixture of experts text embedding models, 2025. URL https://arxiv.org/abs/2502.07972.

[21] OpenAI. Chatgpt: Optimizing language models for dialogue, 2022. URL https://openai.com/blog/chatgpt.

[22] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022. URL https://arxiv.org/abs/2203.02155.

[23] Q. Pang, S. Hu, W. Zheng, and V. Smith. No free lunch in llm watermarking: Trade-offs in watermarking design choices. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[24] J. Piet, C. Sitawarin, V. Fang, N. Mu, and D. Wagner. Mark my words: Analyzing and evaluating language model watermarks, 2024. URL https://arxiv.org/abs/2312.00273.

[25] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, and S. Feizi. Can ai-generated text be reliably detected?, 2025. URL https://arxiv.org/abs/2303.11156.

[26] H. Steck, C. Ekanadham, and N. Kallus. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM Web Conference 2024*, WWW '24, page 887–890. ACM, May 2024. doi: 10.1145/3589335.3651526. URL http://dx.doi.org/10.1145/3589335.3651526.

[27] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love, P. Tafti, L. Hussenot, P. G. Sessa, A. Chowdhery, A. Roberts, A. Barua, A. Botev, A. Castro-Ros, A. Slone, A. Héliou, A. Tacchetti, A. Bulanova, A. Paterson, B. Tsai, B. Shahriari, C. L. Lan, C. A. Choquette-Choo, C. Crepy, D. Cer, D. Ippolito, D. Reid, E. Buchatskaya, E. Ni, E. Noland, G. Yan, G. Tucker, G.-C. Muraru, G. Rozhdestvenskiy, H. Michalewski, I. Tenney, I. Grishchenko, J. Austin, J. Keeling, J. Labanowski, J.-B. Lespiau, J. Stanway, J. Brennan, J. Chen, J. Ferret, J. Chiu, J. Mao-Jones, K. Lee, K. Yu, K. Millican, L. L. Sjoesund, L. Lee, L. Dixon, M. Reid, M. Mikuła, M. Wirth, M. Sharman, N. Chinaev, N. Thain, O. Bachem, O. Chang, O. Wahltinez, P. Bailey, P. Michel, P. Yotov, R. Chaabouni, R. Comanescu, R. Jana, R. Anil, R. McIlroy, R. Liu, R. Mullins, S. L. Smith, S. Borgeaud, S. Girgin, S. Douglas, S. Pandya, S. Shakeri, S. De, T. Klimenko, T. Hennigan, V. Feinberg, W. Stokowiec, Y. hui Chen, Z. Ahmed, Z. Gong, T. Warkentin, L. Peran, M. Giang, C. Farabet, O. Vinyals, J. Dean, K. Kavukcuoglu, D. Hassabis, Z. Ghahramani, D. Eck, J. Barral, F. Pereira, E. Collins, A. Joulin, N. Fiedel, E. Senter, A. Andreev, and K. Kenealy. Gemma: Open models based on gemini research and technology, 2024. URL https://arxiv.org/abs/2403.08295.

[28] Q. Wu and V. Chandrasekaran. Bypassing llm watermarks with color-aware substitutions, 2024. URL https://arxiv.org/abs/2403.14719.

[29] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

[30] H. Zhang, B. L. Edelman, D. Francati, D. Venturi, G. Ateniese, and B. Barak. Watermarks in the sand: Impossibility of strong watermarking for generative models, 2024. URL https://arxiv.org/abs/2311.04378.

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

[32] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[33] Z. Zhang, X. Zhang, Y. Zhang, L. Y. Zhang, C. Chen, S. Hu, A. Gill, and S. Pan. Large language model watermark stealing with mixed integer programming, 2024. URL https://arxiv.org/abs/2405.19677.

[34] X. Zhao, P. Ananth, L. Li, and Y.-X. Wang. Provable robust watermarking for ai-generated text, 2023. URL https://arxiv.org/abs/2306.17439.

[35] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL https://arxiv.org/abs/2306.05685.

## A Related Work

**LLM Watermark.** Advances in large language models have led to promising applications in various real-world domains. However, there is growing concern that language models may be misused for spreading fake news, generating harmful content, and facilitating academic dishonesty. In response, a growing body of research has proposed watermarking as a framework to reliably detect LLM-generated content and mitigate the potential for malicious use [1, 13, 15, 19, 24, 33]. These approaches embed an 'invisible' watermark in the model-generated output, which can later be identified and verified using a secret key. Existing LLM watermarking schemes share various desirable properties: (i) the watermark should be easily detected given knowledge of the secret key; (ii) the watermark should not degrade model-generated outputs; (iii) the watermark should be robust against adversarial attacks; and (iv) the watermark should not be easily stolen for spoofing or removal attacks. However, recent work on 'watermark stealing' directly challenges these theoretical claims.

**LLM Watermark Stealing.** The literature on LLM watermark and watermark stealing primarily considers two attacks: *scrubbing* and *spoofing*. Prior work on watermark scrubbing studies effective watermark removal by paraphrasing [4, 12, 14, 14, 30] or leveraging the LM's side information [23].

In this work, we instead focus on the 'spoofing' attack. The primary approach in this literature is first to estimate the watermarking scheme, then embed this secret key approximation into arbitrary content to generate spoofing attacks. The first work that comprehensively studied spoofing is Sadasivan et al. [25], which targets Kirchenbauer et al. [13]'s distribution-modifying watermark. The authors query the watermarked LLM a million times to learn the underlying token pair distribution, then manually compose texts to spoof the watermark. Follow-up works [10, 12, 17] further highlight the importance of spoofing by operating in more realistic settings. Notably, Jovanović et al. [12] spoofs Kirchenbauer et al. [13]'s soft watermark without access to the watermark $z$-score detector and no base (non-watermarked) responses. The work most related to ours is Pang et al. [23], who provide spoofing attacks on the three most prominent LLM watermarking schemes: KGW [13], Unigram [34], and EXP [15]. Instead of directly estimating the watermarking scheme, the authors leverage the public detection API to enable more sample-efficient spoofing and propose a differential privacy approach to mitigate the risks of having a public detection API. Our work differs from this work in that we consider the inverse transform sampling (ITS) watermark approach in Kuditipudi et al. [15] instead of the exponential minimum sampling (EXP) approach. Kuditipudi et al. [15] suggests the EXP watermark in practice when robustness is of higher priority and the ITS watermark in practice when detection throughput is of higher priority. Thus, our results complement their findings and complete the spoofing attacks on both watermark approaches in Kuditipudi et al. [15].

## B Identifiability of the Reverse-Transform-Based Watermarking Scheme

In this section, we establish an identifiability result for the reverse-transform-based watermarking scheme. Specifically, we address whether it is possible to uniquely recover both the secret key $\xi \in [0, 1]$ and the permutation $\pi$, given outputs from queries to the language model. Formally, we define a selection function $S$ as follows: given a probability distribution $p \in \Delta(\mathcal{V})$, a secret key $\xi \in [0, 1]$, and a permutation $\pi$, the output $S(p, \xi, \pi)$ denotes the token generated according to the procedure described in Algorithm 1.

**Lemma B.1.** *Suppose the selection function satisfies*

$$S(p, \xi, \pi) = S(p, \widehat{\xi}, \widehat{\pi}), \qquad \text{for } p \text{ almost everywhere on } \Delta(\mathcal{V}),$$

*where $\xi, \widehat{\xi} \in [0, 1]$ are secret keys, and $\pi, \widehat{\pi}$ are permutations over the vocabulary. Then, it must hold that either $\pi = \widehat{\pi}$ or $\pi$ is the reverse permutation of $\widehat{\pi}$.*

*Proof.* We prove by contradiction. Assume there exist two permutations $\pi^1$ and $\pi^2$ that are neither identical nor mutual reverses, such that $S(p, \xi, \pi^1) \equiv S(p, \widehat{\xi}, \pi^2)$ for some secret keys $\xi$ and $\widehat{\xi}$.

Then by mathematical induction, we show that there exists tokens $a, b, c$ such that $b$ lies between $a$ and $c$ in $\pi^1$ but not in $\pi^2$.

Now consider a probability distribution $p(x)$ parametrized by $x$ over these tokens defined as follows:

- Token $a$ is assigned probability $x$,
- Token $b$ is assigned probability $0.5(1-x)$,
- Token $c$ is assigned probability $0.5(1-x)$,
- All other tokens have probability zero.

As $x$ increases from 0 to 1, the sampled tokens under permutation $\pi^1$, $S(p(x), \xi, \pi^1)$ emerge in an ordered progression, sometimes involving only two of $\{a, b, c\}$, sometimes all three, depending on the secret key $\xi$, but in every case $b$ invariably appears as the second token in that sequence.

However, under permutation $\pi^2$, since token $b$ is not positioned between tokens $a$ and $c$, the sequential appearance of these three tokens as $x$ changes will differ substantially. In fact, token $b$ will never appear as the second token shown in $S(p(x), \widehat{\xi}, \pi^2)$, no matter what the $\widehat{\xi}$ is. This contradicts with the assumption $S(p, \xi, \pi^1) \equiv S(p, \widehat{\xi}, \pi^2)$. □

**Lemma B.2.** *Suppose the selection function satisfies*

$$S(p, \xi, \pi) = S(p, \widehat{\xi}, \pi), \quad \textit{for p almost everywhere on } \Delta(\mathcal{V}),$$

*where* $\xi, \widehat{\xi} \in [0, 1]$ *are secret keys. Then, it must hold that* $\xi = \widehat{\xi}$.

*Proof.* Assume, for the sake of contradiction, that there exist keys $\xi \neq \widehat{\xi}$ such that $S(p, \xi, \pi) = S(p, \widehat{\xi}, \pi)$ for a.e. $p \in \Delta(\mathcal{V})$. Without loss of generality, let $\xi > \widehat{\xi}$.

Denote the first and second tokens in the permutation $\pi$ by $a := \pi(1)$ and $b := \pi(2)$, respectively. Consider a probability distribution $p(x)$

$$p_x(a) = x, \qquad p_x(b) = 1 - x, \qquad p_x(c) = 0 \text{ for all } c \notin \{a, b\}, \quad 0 \leq x \leq 1.$$

For any $x \in (\widehat{\xi},\ \xi)$ we then have

$$S\big(p_x,\ \xi,\ \pi\big) = a \quad \text{while} \quad S\big(p_x,\ \widehat{\xi},\ \pi\big) = b,$$

contradicting the assumed equality of the two outputs. This finishes the proof. □

With Lemma B.1 and Lemma B.2, we are ready to present the proof of Theorem 3.1, which highlights an intriguing equivalence between the two parameterizations:

*Proof.* By lemma B.1, we know either $\pi = \widehat{\pi}$ or $\pi = \text{reverse}(\widehat{\pi})$ holds. If $\pi = \widehat{\pi}$, we have $S(p, \xi, \pi) = S(p, \widehat{\xi}, \pi)$ for almost every $p \in \Delta(\mathcal{V})$. Then, by lemma B.2, we obtain $\xi = \widehat{\xi}$.

If $\pi = \text{reverse}(\widehat{\pi})$, first note that

$$S\big(p, \widehat{\xi}, \widehat{\pi}\big) = S\big(p, 1 - \widehat{\xi}, \text{inv}(\widehat{\pi})\big) \quad \text{for almost every } p \in \Delta(\mathcal{V}), \tag{1}$$

which holds for any $\widehat{\xi}, \widehat{\pi}$. Because $\pi = \text{inv}(\widehat{\pi})$, (1) becomes

$$S\big(p, \widehat{\xi}, \widehat{\pi}\big) = S\big(p, 1 - \widehat{\xi}, \pi\big) \quad \text{for almost every } p \in \Delta(\mathcal{V}).$$

By assumption,

$$S\big(p, \xi, \pi\big) = S\big(p, \widehat{\xi}, \widehat{\pi}\big) \quad \text{for almost every } p \in \Delta(\mathcal{V}),$$

so we further have

$$S\big(p, \xi, \pi\big) = S\big(p, 1 - \widehat{\xi}, \pi\big) \quad \text{for almost every } p \in \Delta(\mathcal{V}).$$

Applying lemma B.2 again yields $\xi = 1 - \widehat{\xi}$.

□

# C Appendix: Additional Background on Watermarking

In this section, we include the full algorithms used by Kuditipudi et al. [15] to generate the watermark using inverse transform sampling (Algorithm 1) and to detect the watermark (Algorithm 2) for completion.

**Algorithm 1:** Watermarked Text Generation via Inverse Transform Sampling

**Input:** Watermark key sequence $\xi = (\xi_1, \xi_2, \ldots, \xi_n) \in \Xi^n$, generation length $m \in \mathbb{N}$, language model $p : \mathcal{V}^* \to \Delta(\mathcal{V})$, bijective permutation function $\pi : [|\mathcal{V}|] \to \mathcal{V}$, vocabulary $\mathcal{V}$.
**Output:** Generated watermarked string $y = (y_1, \ldots, y_m) \in \mathcal{V}^m$

**for** $t = 1$ **to** $m$ **do**
  $C_k = \sum_{i=1}^{k} p(\pi(i)|y_{:t-1}), \quad \forall k \in [|\mathcal{V}|]\}$
  $\zeta = \xi_{t \bmod n}$
  $k = \min\{k \mid C_k \geq \zeta\}$
  $y_t = \pi(k)$
**return** $y$

At detection time, Algorithm 2 is used to detect the watermark by using a permutation test to obtain a $p$- value. If the test returns a small $p$-value (typically chosen to be $p < 0.05$), then the text is watermarked.

**Algorithm 2:** Watermark Detection via Alignment Cost

**Input:**
- Watermark key sequence $\xi = (\xi_1, \ldots, \xi_n) \in [0, 1]^n$.
- Candidate text $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_m)$.
- Permutation $\pi : \{1, \ldots, |\mathcal{V}|\} \to \mathcal{V}$.
- Language model $p : \mathcal{V}^* \to \Delta(\mathcal{V})$.
- Number of random samples $T$ for the permutation test.

**Output:** p-value $p$ indicating the likelihood that $\tilde{y}$ is watermarked.

$D \leftarrow \textsc{ComputeAlignmentCost}(\tilde{y}, \xi)$
$c \leftarrow 0$
**for** $i = 1$ **to** $T$ **do**
  Sample $\mu^{(i)} \sim \mathcal{U}[0, 1]^n$
  $D^{(i)} \leftarrow \textsc{ComputeAlignmentCost}(\tilde{y}, \mu^{(i)})$
  **if** $D \geq D^{(i)}$ **then**
    $c \leftarrow c + 1$
$p \leftarrow \frac{1+c}{T+1}$
**return** $p$
**Function** $\textsc{ComputeAlignmentCost}(\tilde{y}, \xi)$:
  $D \leftarrow 0$
**for** $t = 1$ **to** $m$ **do**
    $k_t \leftarrow \pi^{-1}(\tilde{y}_t)$ ▷ Obtain token index under $\pi$
    $C_{k_t} \leftarrow \sum_{j=1}^{k_t} p\big(\pi(j) \mid \tilde{y}_{:t-1}\big)$ ▷ Compute cumulative probability
    $\zeta_t \leftarrow \xi_{t \bmod n}$ ▷ Retrieve key element
    $d_t \leftarrow |C_{k_t} - \zeta_t|$ ▷ Token alignment cost
    $D \leftarrow D + d_t$
  **return** $D$
**End Function**

**Algorithm 3:** Dataset Construction for Reverse-Engineering the Permutation

**Input:** Fixed prompt $x_{\text{prefix}}$, number of queries $N$
**Output:** Dataset $\mathcal{D} = \{(\xi_i, x_i) \mid i \in [N]\}$
1: **for** $i = 1$ to $N$ **do**
2: Sample a random key $\xi_i \sim \mathcal{U}[0, 1]$.
3: Query the language model with the fixed prompt $x_{\text{prefix}}$ using $\xi_i$.
4: Receive response and extract the first token $x_i$.
5: **return** $\mathcal{D}$

# D Methodology: First two threat models

## D.1 Known secret key sequence $\{\xi_i\}$, reverse-engineer permutation $\pi$:

When the random key sequence $\{\xi_i\}$ is known, the attacker's goal is to reconstruct the permutation map $\pi$. In this regime, we assume that the attacker can modify the random key sequence $\{\xi_i\}$ to perform queries for inferring the permutation. To facilitate this, we construct a dataset of (secret key, prompt) pairs $\{(\xi_i, x_i)\}$ as follows (details are summarized in Algorithm 3):

1. **Fixed Prompt:** A fixed prompt $x_{\text{prefix}}$, e.g., 'Once upon a time,' is used across multiple queries to ensure the conditional probability distribution $p(\cdot|x_{\text{prefix}})$ remains the same.
2. **Multiple Queries:** For each query to the LM, a different random key $\xi_i \in [0, 1]$ is used. The key $\xi_i$ determines the CDF threshold for selecting the first token $x_i$ in the response.
3. **Dataset Construction:** By repeatedly querying the language model with the fixed prompt $x_{\text{prefix}}$ and different random keys $\xi_i$, we obtain a collection of pairs $\{(\xi_i, x_i)\}$, where each $x_i$ is the first token selected in response to $\xi_i$.

Then, we utilize a sorting-based algorithm to reverse engineer by first sorting the observed data pairs $\{\xi_i, x_i\}$ in ascending order of $\xi_i$. Then, we record the order in which each unique token $x_i$ appears for the first time. This sequence of first occurrences defines the estimated permutation $\pi$ over the vocabulary. The detail of this sorting-based algorithm is summarized in Algorithm 4.

**Algorithm 4:** Reverse-Engineer Secret Permutation $\pi$

**Input:** Dataset $\mathcal{D} = \{(\xi_i, x_i) \mid i \in [N]\}$
**Output:** Recovered permutation $\pi$
1: Sort $\mathcal{D}$ in ascending order by $\xi_i$ to obtain $\{(\xi_{(1)}, x_{(1)}), (\xi_{(2)}, x_{(2)}), \ldots, (\xi_{(N)}, x_{(N)})\}$.
2: Initialize empty list $R = \varnothing$.
3: **for** $i = 1$ to $N$ **do**
4: **if** $R$ is empty or $x_{(i)}$ is not the last element of $R$ **then**
5: Append $x_{(i)}$ to $R$.
6: Set $\pi \leftarrow R$.
7: **return** $\pi$.

## D.2 Known permutation $\pi$, reverse-engineer secret key sequence $\{\xi_i\}$

In this regime, where the permutation $\pi$ is known to the adversary, each generated token $x$ can be mapped to its corresponding index in the vocabulary via the inverse mapping $k = \pi^{-1}(x)$. Given an index $k$, the secret key $\xi$ corresponding to the token $x$'s selection must lie within the following interval determined by the cumulative distribution function of the language model, i.e., $C_{k-1} < \xi \leq C_k$, where the $C_{k-1}$ and $C_k$ are the CDF values with respect to permutation $\pi$ defined as $C_k = \sum_{j=1}^{k} p\big(\pi(j) \mid \text{prefix}\big)$ with the convention $C_0 = 0$.

With this principle, we use Algorithm 5 to estimate the secret keys when the permutation is known. When the attacker only knows a partial ordering over a subset of tokens, we could modify Algorithm 5 with the same principle to estimate the secret keys. The details are in Algorithm 6 (Appendix E).

**Algorithm 5:** Reverse-Engineering Pseudorandom Secret Key from Watermarked Outputs

**Input:**
- A set of output sentences $\mathcal{Y}$ generated by the watermarked model.
- Known permutation $\pi$ (and its reverse mapping $\pi^{-1}$).
- Watermark key length $n$ (i.e., there are $n$ pseudorandom numbers $\xi_1, \xi_2, \ldots, \xi_n$).

**Output:** Estimated lower bound $\mathrm{LB}_i$ and upper bound $\mathrm{UB}_i$ for each pseudorandom number $\xi_{i:i\in[n]}$.

**Initialization:** For $i = 1$ to $n$, set $\mathrm{LB}_i \leftarrow 0$ and $\mathrm{UB}_i \leftarrow 1$.
**for** each sentence $y \in \mathcal{Y}$ **do**
  **for** each token $y_s$ in $y$ (with $s$ as the token index) **do**
    $i \leftarrow s \mod n$.
    Compute $k \leftarrow \pi^{-1}(y_s)$.
    Compute the cumulative probabilities $C_{k-1}$ and $C_k$ with respect to permutation $\pi$, where $C_k = \sum_{j=1}^{k} p\big(\pi(j) \mid \text{prefix}\big)$ and $C_0 = 0$.
    **Update bounds:** $\mathrm{LB}_i \leftarrow \max(\mathrm{LB}_i,\, C_{k-1}), \quad \mathrm{UB}_i \leftarrow \min(\mathrm{UB}_i,\, C_k)$.
**return** $\{(\mathrm{LB}_i, \mathrm{UB}_i)\}_{i=1}^{n}$

# E Appendix: Reverse-Engineering Pseudorandom Numbers from Watermarked Outputs with Partial Ordering

As mentioned in section D.2, when the attacker only knows a partial ordering over a subset of tokens, we can use the same principles to estimate the secret keys. Algorithm 6 below includes the details on how to modify Algorithm 5 in such settings.

**Algorithm 6:** Reverse-Engineering Pseudorandom Numbers from Watermarked Outputs with Partial Ordering

**Input:**
- A set of output sentences $\mathcal{Y}$ generated by the watermarked model.
- A subset $S \subset V$ with known ordering given by $\pi^{-1}$ (i.e., for each $t \in S$, $\pi^{-1}(t)$ is known).
- Watermark key length $n$ (i.e., there are $n$ pseudorandom numbers $\xi_1, \xi_2, \ldots, \xi_n$).

**Output:** Estimated bounds for each pseudorandom number $\xi_i$ (i.e., lower bound $LB_i$ and upper bound $UB_i$ for $i = 1, \ldots, n$).

**Initialization:** For $i = 1$ to $n$, set $LB_i \leftarrow 0$ and $UB_i \leftarrow 1$.
**for** each sentence $y \in \mathcal{Y}$ **do**
  **for** each token $y_s$ in $y$ (with $s$ as the token index) **do**
    $i \leftarrow s \mod n$
    **if** $y_s \in S$ **then**
      Let $r \leftarrow \pi^{-1}(y_s)$.
      Compute the lower temporary bound:

$$L_{\text{temp}} = \sum_{\substack{t \in S \\ \pi^{-1}(t) < r}} p\big(t \mid \text{prefix}\big).$$

      Compute the upper temporary bound:

$$U_{\text{temp}} = 1 - \sum_{\substack{t \in S \\ \pi^{-1}(t) > r}} p\big(t \mid \text{prefix}\big).$$

      **Update bounds:**

$$LB_i \leftarrow \max\big(LB_i,\, L_{\text{temp}}\big), \quad UB_i \leftarrow \min\big(UB_i,\, U_{\text{temp}}\big).$$

**return** $\{(LB_i, UB_i)\}_{i=1}^{n}$

# F Appendix: Recover Permutation via MergeSort

In this section we include the details of the algorithm to recover the permutation order via MergeSort, as mentioned in section 3.1.

**Algorithm 7:** Recover Global Ordering via QueryLLM and MergeSort

**Input:** Vocabulary of tokens $T = \{t_1, t_2, \ldots, t_n\}$
**Output:** Ordered token sequence $T_{\text{sorted}}$ equivalent to the hidden permutation $\pi$ (or its reverse)

$T_{\text{sorted}} \leftarrow$ MERGESORT$(T)$
**return** $T_{\text{sorted}}$
**Function** `QueryLLM`$(a, b)$:
  Query the language model with a prompt that forces a random choice between $a$ and $b$.
  **If** the model outputs token $b$, **then** interpret as $a < b$;
    **Else** interpret as $b < a$.
  **Return** the corresponding comparison result.
**End Function**
**Function** MERGESORT($arr$):
  **If** length($arr$) $\leq 1$
      **return** $arr$
  $mid \leftarrow \lfloor$length$(arr)/2 \rfloor$
  $left \leftarrow$ MERGESORT$(arr[0 : mid])$
  $right \leftarrow$ MERGESORT$(arr[mid :])$

**return** MERGE$(left, right)$
**End Function**
**Function** MERGE$(left, right)$:
  $merged \leftarrow$ empty list; $i \leftarrow 0$; $j \leftarrow 0$
  **While** $i <$ length$(left)$ and $j <$ length$(right)$:
    **If** QueryLLM($left[i]$, $right[j]$) returns "a < b":
      Append $left[i]$ to $merged$
      $i \leftarrow i + 1$
    **Else**:
      Append $right[j]$ to $merged$
      $j \leftarrow j + 1$
  **End While**
  Append remaining elements of $left[i :]$ to $merged$
  Append remaining elements of $right[j :]$ to $merged$

**return** $merged$
**End Function**

We provide the following result to illustrate why Algorithm 7 is indeed a valid approach in our setting.

**Fact F.1.** *Let the probability distribution $p$ be defined over tokens as*

$$p(a) = 0.5, \quad p(b) = 0.5, \quad p(c) = 0 \text{ for all other tokens } c.$$

*Suppose the permutation $\pi$ satisfies $\pi^{-1}(b) > \pi^{-1}(a)$. Then the selection output satisfies:*

$$S(p, \xi, \pi) = \begin{cases} a, & \text{if } \xi \leq 0.5, \\ b, & \text{if } \xi > 0.5. \end{cases}$$

The proof of fact F.1 is straight forward, therefore we omit it.

With this fact established, the validity of algorithm 7 becomes clear: if the corresponding secret key satisfies $\xi \leq 0.5$, then `QueryLLM`$(a, b)$ returns the token appearing earlier in the permutation $\pi$, causing the algorithm to recover $inv(\pi)$. Otherwise, it returns the token appearing later, enabling direct recovery of $\pi$. In either case, the algorithm remains valid.

# G Appendix: Additional Experiments

In this section, we provide additional experimental detail and results previously omitted from Section 4.

**Text Quality Evaluation** We apply the watermark detection and compute the perplexity of each sample using the base Llama-3.1-8B model to generate the perplexity scores. In addition, we use the Nomic Embed Model [20] to generate embeddings to calculate the cosine similarity between spoofed text and the non-watermarked text as well as between the genuine watermarked text and the non-watermarked text.

We compute perplexity using a separate reference language model (Llama-3.1-8B) that is different from the generation model. We decode each output using the generation model's tokenizer to preserve lexical fidelity, then re-tokenize the resulting text with the perplexity model's tokenizer for scoring. To mitigate distortion from rare token artifacts or malformed completions, we filter the samples with perplexity values that exceed the 95th percentile within each batch (similar to how Jovanović et al. [12] define their attack success metric). For completion, we also report the results using unfiltered samples in Table 3 and Table 4 in Appendix G. Cosine similarity measures how similar two vectors are regardless of their magnitude [26]. We generate learned embeddings using the pretrained nomic embed language model [20] to turn each generated text into a vector which captures the meaning of the sentences. We then calculate the cosine similarity of the non-watermarked text vectors compared to the watermarked text vectors and spoofed text vectors which tells us how close the vectors point in the same directions. Table 1 presents these statistics for the same sets of samples, where low perplexity and high cosine similarity indicates more fluent text.

## G.1 Evaluating Estimated Vocabulary Permutation $\pi$ and Estimated Secret Key

First, we investigate whether an attacker can successfully recover the secret permutation $\pi$ and key sequence $\{\xi\}$. As described in Section 3.1, when the permutation is unknown, the attacker can estimate it by adaptively querying the LLM with pairwise token comparisons (the `QueryLLM(a,b)` procedure in Appendix F) and aggregating these comparisons via a merge-sort algorithm. This process would accurately estimate the true permutation in an ideal scenario with a perfectly consistent LLM that always follows instructions. In practice, LLMs may not provide a full-ordered preference for all token pairs (due to model uncertainties or equal probabilities). We address this issue by forcing a decision in ambiguous comparisons: if the model returns a tie, we break the tie arbitrarily to obtain a complete ordered list. We find this strategy effective: using well-engineered comparison prompts, an attacker can obtain an accurate estimation of the permutation over the vocabulary. We observe that smaller model (e.g., OPT-125M) are often easier to spoof, i.e., spoofed generations more consistently pass watermark detection despite only partial permutation recovery. We attribute this to two primary factors: (i) smaller models tend to have lower-entropy token distributions, making the ITS watermarking mechanism more brittle to approximate key values and permutation errors; and (ii) the lower diversity of output sequences in smaller models reduces alignment cost variance, leading to more forgiving p-values under the permutation test. However, while spoofing detection is easier, the output quality suffers: larger models like Llama-3.1-8B and Mistral-7B produce significantly more fluent and coherent text, as evidenced by consistently lower perplexity and higher cosine similarity scores.

Figure 3 shows the mean absolute error (MAE) when estimating the secret keys as a function of the percentage of known permutation $\pi$ for the model vocabulary. As expected, we observe that the more information on $\pi$, the more accurate the estimation process, with the estimation error being virtually unchanged after at least $60\%$ of the permutation is known, validating the use of Algorithm 5 and the theoretical insights in Appendix B. Additionally, we note that in practice a successful spoofing attack might depend on the dataset, model and prompt used, and hence we leave for a future work exploring the connection between the estimation error of secret keys and the feasibility of spoofing attacks.

## G.2 Additional spoofing evaluations

**Partial recovery.** Our results in Table 1 and Figure 2 show that perfect recovery of the permutation is not needed to carry out a spoofing attack - a partially correct permutation can suffice. We verify this by evaluating the attacker's success rate when only some of the true permutation is

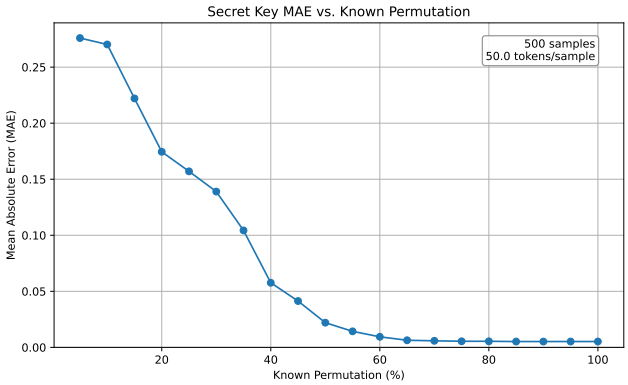


Figure 3: Secret key estimation error (MAE) as a function of known permutation fraction $\pi$ using 500 samples of 50 tokens each.

known. Even with only approximately the first half of the permutation recovered, spoofing the watermarked text with high success rate and minimal PPL penalty is possible across all model types. As permutation knowledge increases, the attack is more effective. Even with moderate permutation recovery (e.g., when the attacker successfully recovered the permutation of the first $50\%$ of tokens), the secret key estimation error remains low and enables successful spoofing. Overall, we demonstrate that an attacker can recover both components of the distortion-free watermark. With a sufficiently accurate estimate of $\pi$ and $\{\xi\}$– achieved with a feasible number of queries – the attacker is then able to generate spoofed text.

**(Q2) Larger models impact spoofing quality.** We observe that Gemma-7B underperforms relative to other models in both spoofing detection success and output perplexity. This behavior may be due to the LLM's more diffuse token distribution and less consistent top-k token ranking, which impair permutation recovery and increase alignment cost under the ITS watermark detection metric. In contrast, Llama-3.1-8B-Instruct shows strong spoofing performance due to its highly consistent token ranking, low-entropy output distributions, and robust response to prompt-based token comparisons. These traits make it easier to recover the watermark permutation and generate spoofed text that passes detection with low perplexity. We found that smaller model like OPT-125M achieve strong spoofing performance, with spoofed outputs consistently passing the watermark detection test. We attribute this to their flatter output distributions, which make the ITS watermarking more tolerant to approximation errors in the spoofing parameters. We do see these models produce text with higher perplexity, reflecting their limited language fluency and smaller effective vocabulary. While spoofing is statistically easier, the generated text lacks the quality of outputs from larger LLMs.

**Filtering out high perplexity outliers** In Section 4, we have presented Table 1 containing the summary statistics for evaluating the spoofing attack on different LLMs. For completeness, we also provide additional tables of the same format in the appendix. Particularly, we see some outliers with high $p$-values bringing the mean $p$-value up (Table 2). The main differences between the results in Table 1 and these additional tables are two-fold: (i) in Table 3 and 4, we report the results without filtering outliers as described in Section 4 (we filter the samples with PPL values that exceed the 95th percentile within each batch); and (ii) in Table 2 and 4, we report the mean $p$-value across 100 samples instead of the median $p$-value. Furthermore, we provide the comparison of $p$-value and perplexity across different LLMs without filtering outliers in PPL values Figure 4. In these new results, our observations are consistent with the findings in Section 4. Our proposed approach can successfully spoof the watermark by Kuditipudi et al. [15] across different LLMs, and the spoofing results generally improves with more capable LLMs.

**Additional dataset: LFQA.** In addition to the spoofing results on the OpenGen dataset, we also evaluate the robustness of our spoofing attack on the LFQA dataset[6], which contains long-form question-answering prompts. Despite the domain shift and the stylistic differences in prompt structure, we found that our attack remained consistently effective across different LLMs (Llama-3.1-8B, Mistral-7B, OPT-125M). The spoofed generations on LFQA achieves low $p$-value, are

Table 2: Comparison of baseline and spoofed outputs with various LLMs. We report the mean $p$-value (p-val) for detection test, and a combination of perplexity (PPL) and cosine similarity (co-sim) with thresholding for the text quality assessment.

| **Model** | **Non-WM** | | **WM** | | |
|---|---|---|---|---|---|
| | p-val | PPL | p-val | PPL | co-sim |
| Llama-3.1-8B | 0.5 | 5.67 | 3.4e-02 | 29.79 | 0.866 |
| Mistral-7B | 0.5 | 9.56 | 8.2e-03 | 65.08 | 0.855 |
| Gemma-7B | 0.5 | 9.96 | 8.7e-02 | 119.41 | 0.846 |
| OPT-125M | 0.5 | 9.84 | 1.1e-04 | 176.73 | 0.834 |

| **Model** | **Spoof @** $50\%$ | | | **Spoof @** $25\%$ | | |
|---|---|---|---|---|---|---|
| | p-val | PPL | co-sim | p-val | PPL | co-sim |
| Llama-3.1-8B | 5.6e-02 | 108.04 | 0.869 | 1.2e-01 | 55.53 | 0.854 |
| Mistral-7B | 6.1e-03 | 65.83 | 0.86 | 3.4e-02 | 74.79 | 0.858 |
| Gemma-7B | 7.6e-02 | 119.84 | 0.838 | 1.6e-01 | 134.15 | 0.833 |
| OPT-125M | 1.0e-04 | 150.08 | 0.832 | 1.4e-02 | 155.6 | 0.836 |

Table 3: Comparison of baseline and spoofed outputs with various LLMs. We report the median $p$-value (p-val) for detection test, and a combination of perplexity (PPL) and cosine similarity (co-sim) without thresholding for the text quality assessment.

| **Model** | **Non-WM** | | **WM** | | |
|---|---|---|---|---|---|
| | p-val | PPL | p-val | PPL | co-sim |
| Llama-3.1-8B | 0.5 | 5.5 | 1.0e-04 | 18.09 | 0.871 |
| Mistral-7B | 0.5 | 13.01 | 1.0e-04 | 66.91 | 0.861 |
| Gemma-7B | 0.5 | 11.22 | 1.0e-04 | 65.53 | 0.85 |
| OPT-125M | 0.5 | 10.08 | 1.0e-04 | 142.79 | 0.836 |

| **Model** | **Spoof @** $50\%$ | | | **Spoof @** $25\%$ | | |
|---|---|---|---|---|---|---|
| | p-val | PPL | co-sim | p-val | PPL | co-sim |
| Llama-3.1-8B | 1.0e-04 | 24.52 | 0.871 | 1.0e-04 | 27.47 | 0.859 |
| Mistral-7B | 1.0e-04 | 66.89 | 0.864 | 1.0e-04 | 80.61 | 0.863 |
| Gemma-7B | 1.0e-04 | 84.61 | 0.837 | 1.0e-04 | 92.22 | 0.829 |
| OPT-125M | 1.0e-04 | 142.04 | 0.834 | 1.0e-04 | 147.35 | 0.845 |

indistinguishable from genuine watermarked text, and maintain high text quality as measured through perplexity. These results demonstrate that our spoofing attack generalizes well across datasets with different linguistic characteristics, and is not overfitted to a particular prompt style or topic distribution. We summarize these results in Table 5 below.

### G.3 Spoofing Examples

We provide additional examples of our spoofing attacks. Specifically, under the same prompt "What is Quantum Computing", we report the outputs from a non-watermarked LM, watermarked LM and our spoofing attack as well as the $p$-value (for watermark detection) and perplexity score (for text quality) using Llama-3.1-8B-Instruct for all generation types.

**Prompt:** `"What is Quantum Computing?"` **(200 tokens)**

**Watermarked Output**

> A quantum computer is a developing field of technology that uses the principles of quantum mechanics, such as superposition and entanglement, to perform calculations and operations that are beyond the capabilities of traditional computers.

Table 4: Comparison of baseline and spoofed outputs with various LLMs. We report the mean $p$-value (p-val) for detection test, and a combination of perplexity (PPL) and cosine similarity (co-sim) without thresholding for the text quality assessment.

| **Model** | **Non-WM** | | **WM** | | |
|---|---|---|---|---|---|
| | p-val | PPL | p-val | PPL | co-sim |
| Llama-3.1-8B | 0.5 | 6.27 | 3.2e-02 | 409.25 | 0.866 |
| Mistral-7B | 0.5 | 14.69 | 7.8e-03 | 89.11 | 0.855 |
| Gemma-7B | 0.5 | 12.04 | 8.1e-02 | 172.82 | 0.846 |
| OPT-125M | 0.5 | 11.62 | 1.1e-04 | 199.52 | 0.834 |

| **Model** | **Spoof @ 50%** | | | **Spoof @ 25%** | | |
|---|---|---|---|---|---|---|
| | p-val | PPL | co-sim | p-val | PPL | co-sim |
| Llama-3.1-8B | 3.3e-02 | 299.5 | 0.869 | 1.6e-02 | 717.58 | 0.854 |
| Mistral-7B | 5.7e-03 | 87.97 | 0.86 | 4.2e-02 | 98.18 | 0.858 |
| Gemma-7B | 1.0e-01 | 153.86 | 0.838 | 1.5e-01 | 163.26 | 0.833 |
| OPT-125M | 1.0e-04 | 185.79 | 0.832 | 1.3e-02 | 172.88 | 0.836 |

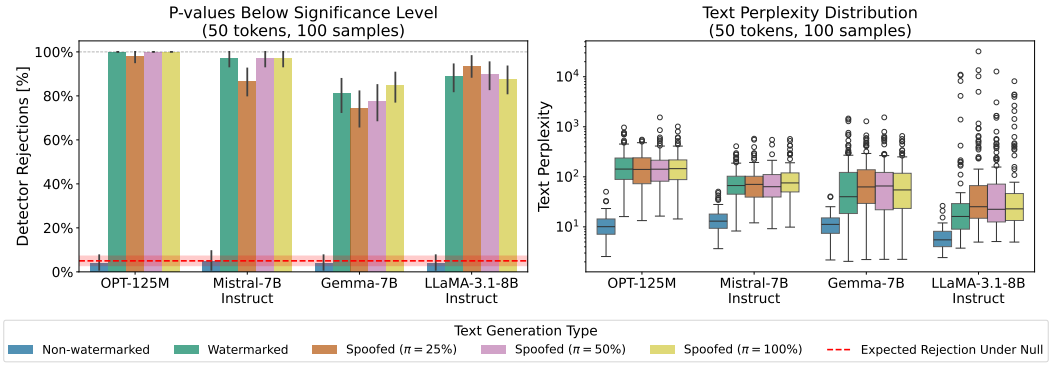


Figure 4: Spoofing results as in Figure 2, but without removing generated low-quality text. *Left*: Percentage of $p$-values below the significance level ($\alpha = 0.05$) for watermark detection in Algorithm 2, across LLMs and known permutation $\pi$ proportions. Our spoofing attacks achieve similar rejection rates as watermarked samples from each model. *Right:* Distribution of text perplexity for the generated samples, across LLMs and known permutation $\pi$ proportions. Our spoofed attacks do not exhibit significant difference in perplexity distributions with respect to the watermarked text.

Table 5: Comparison of baseline and spoofed outputs with various LLMs on the LFQA dataset. We report the median $p$-value (p-val) for detection test, and perplexity (PPL) for the text quality assessment.

| **Model** | **Non-WM** | | **WM** | | **Spoof 50** | | **Spoof 25** | |
|---|---|---|---|---|---|---|---|---|
| | p-val | PPL | p-val | PPL | p-val | PPL | p-val | PPL |
| Llama-3.1-8B | 0.49 | 3.68 | 1.0e-04 | 11.58 | 2.0e-04 | 14.02 | 8.8e-03 | 19.14 |
| Mistral-7B | 0.44 | 5.13 | 1.0e-04 | 6.28 | 1.0e-04 | 31.57 | 1.0e-04 | 37.23 |
| OPT-125M | 0.52 | 5.77 | 1.0e-04 | 56.34 | 1.0e-04 | 66.60 | 1.0e-04 | 65.89 |

Quantum computers are designed to solve complex problems, such as simulating chemical reactions or factoring large numbers, exponentially faster than classical computers. They work by manipulating qubits (quantum bits) that can exist in multiple states simultaneously, allowing for massive parallel processing and exploration of vast solution spaces.

The core concept of quantum computing is to exploit the inherent properties of quantum systems to perform operations that cannot be simulated with classical computers. These properties include:

**Quantum superposition:** A qubit can represent both 0 and 1 simultaneously, enabling it to process multiple possibilities at once.

**Quantum entanglement:** Qubits can become "entangled," meaning their properties are connected, allowing for instant communication and correlation between them.

**Quantum measurement:** Measuring a quantum system collapses it into a definite state, which is used to extract information from the system.

**Non-Watermarked Output**

What is quantum computing? Quantum computing is a new paradigm for computing that uses the principles of quantum mechanics to perform calculations and operations on data.

Quantum computing is based on the principles of superposition, entanglement, and interference, which allow for the processing of vast amounts of data in parallel. Quantum computing has the potential to solve complex problems that are currently unsolvable with classical computers.

Quantum computing is a new and rapidly evolving field that has the potential to revolutionize the way we approach complex problems in fields such as cryptography, optimization, and machine learning. Quantum computing is based on the principles of quantum mechanics, which describe the behavior of matter and energy at the smallest scales. Quantum computing uses quantum bits, or qubits, which can exist in multiple states simultaneously, unlike classical bits, allowing for massive parallelism in computation.

**Spoofed Output**

A quantum computer is a computer that can use quantum mechanical phenomena, such as superposition and entanglement, that enable it to solve certain problems that are difficult for a classical computer to solve, faster and more efficiently. Quantum computing is a new and rapidly evolving field which has seen significant advances in recent years.

Quantum computers are based on quantum mechanical systems, such as quantum bits (qubits) and quantum gates, and are able to perform certain operations that are beyond the capabilities of classical computers. Quantum computers have the potential to revolutionize the field of computation and may have important applications in fields such as cryptography, optimization, and simulation.

A classical computer works on binary (0, 1) bits to process information and make decisions. In contrast, a quantum computer will work on quantum bits (0, 1, 2, and other states), known as qubits, which allow for complex operations and parallelism.

Quantum computing has the potential to accelerate countless applications and industries.

Table 6: Summary statistics comparing generation types by detection p-value and perplexity.

| **Generation Type** | **p-value** | **Perplexity** |
|---|---|---|
| Watermarked | $9.999 \times 10^{-5}$ | 2.74 |
| Non-Watermarked | 0.469 | 1.56 |
| Spoofed | $9.999 \times 10^{-5}$ | 11.49 |

## H Computational Resources

Both experiments and evaluations are run on an Ubuntu Machine with 96 Intel Xeon Platinum 8259CL CPUs, 384GB of RAM, a storage volume of 500 Gb and 8 A100 NVIDIA Tensor Core GPUs.

## I Limitations and Broader Impact

**Limitations.** We focus on the ITS variant of the robust distortion-free watermark, which augment the line of work on stealing SOTA robust watermark. A comprehensive study on other distortion-free watermarks that share this (secret key, permutation) structure would be an interesting future direction. Furthermore, in our experiments, we use the same LLM architecture for the base watermarked model and the spoofing model, e.g., we use OPT-125M to generate the watermarked outputs and spoof by running our proposed approach on OPT-125M. An interesting extension of our work could be evaluating whether the current approach can still spoof successfully when the base model and the attacking models are different. Particularly, if our approach can spoof a large, watermarked LLM by using a smaller, distilled model, then our approach would be more applicable in practice and further challenge the theoretical claims of Kuditipudi et al. [15]'s watermark. Finally, our experiments only use a single dataset: OpenGen [14]. Evaluating our approach on more datasets used in the watermark stealing literature would strengthen our theoretical and empirical claims.

**Impact Statement.** As outlined in prior research, watermarking plays a crucial role in addressing social issues such as detecting plagiarism, tracing text origins, and combating misinformation. Our work investigates vulnerabilities in LLM watermarking that could potentially be exploited by attackers to break the watermark mechanism, posing risks to the owners or users of the model. However, we believe that our research has a positive societal impact by revealing the current weaknesses of watermarking methods, highlighting the need for stronger, more reliable systems, and advocating for improved evaluation frameworks. Ultimately, this work contributes to advancing the field toward more effective LLM watermarking solutions.