Attacking LLM Watermarks by Exploiting Their Strengths

Qi Pang, Shengyuan Hu, Wenting Zheng, Virginia Smith
Carnegie Mellon University
{qipang, shengyuanhu, wenting, smithv}@cmu.edu

Abstract

Advances in generative models have made it possible for AI-generated text, code, and images to mirror human-generated content in many applications. *Watermarking*, a technique that aims to embed information in the output of a model to verify its source, is useful for mitigating misuse of such AI-generated content. However, existing watermarking schemes remain surprisingly susceptible to attack. In particular, we show that desirable properties shared by existing LLM watermarking systems such as quality preservation, robustness, and public detection APIs can in turn make these systems vulnerable to various attacks. We rigorously study potential attacks in terms of common watermark design choices, and propose best practices and defenses for mitigation—establishing a set of practical guidelines for embedding and detection of LLM watermarks.

1 Introduction

Modern generative models have notably enhanced the quality of AI-produced content Brown et al. (2020); Saharia et al. (2022); OpenAI (2023a; 2022). For example, large language models (LLMs) like those powering ChatGPT OpenAI (2022) can generate text that closely resembles human-crafted sentences. While this has led to exciting new applications of machine learning, there is also growing concern around the potential for misuse of these models, leading to a flurry of recent efforts on developing techniques to detect AI-generated content. A promising approach is to embed invisible *watermarks* into model-derived content, which can then be extracted and verified using a secret watermark key Kirchenbauer et al. (2023a); Fairoze et al. (2023); Christ et al. (2023); Kuditipudi et al. (2023); Zhao et al. (2024); Kirchenbauer et al. (2023b); Hu et al. (2023); Wu et al. (2023).

In this work, we identify that many of the key properties that make existing LLM watermarks successful also render them susceptible to attack. We study a number of simple, scalable attacks that can take advantage of common properties of existing watermarking schemes, including:

1. **Robustness** of the watermarks to potential modifications in the output text, so that the watermarks cannot be easily removed Kuditipudi et al. (2023); Kirchenbauer et al. (2023a); Zhao et al. (2024); Kuditipudi et al. (2023); Christ et al. (2023); Hu et al. (2023);
2. **Distribution preservation** of the output watermarked tokens, whose distribution is often desired to be close to the unwatermarked tokens to ensure the watermark does not degrade model utility Kirchenbauer et al. (2023a); Fairoze et al. (2023); Kuditipudi et al. (2023); Christ et al. (2023); Hu et al. (2023); Wu et al. (2023); and
3. **Public detection APIs**, which allow the general public to easily verify whether or not candidate text is AI-generated Fairoze et al. (2023); Solaiman et al. (2019); Mitchell et al. (2023).

While these common properties of existing watermarking schemes have clear benefits, we demonstrate that they can also enable malicious actors to easily launch two types of attacks: 1) *watermark-removal attacks*, which remove the watermark from the watermarked content, and 2) *spoofing attacks*, which create (toxic) content with a target watermark embedded, thus making the content appear to be generated by a specific LLM when it is in fact not (see motivating examples in Table 1).

Our work rigorously explores removal and spoofing attacks for LLM watermarks. In doing so, we identify a number of critical trade-offs between watermark security and utility that emerge as a result of watermarking design choices. With these trade-offs in mind we propose potential defenses as well

Prompt	`Alan Turing was born in ...`
Unwatermarked Z-Score: 0.16 ↓ PPL: 3.19	`Alan Turing was born in 1912 and died in 1954. He was an English mathematician, logician, cryptanalyst, and computer scientist. In 1938, Turing joined the Government Code and Cypher School (GC&CS), where he contributed ...`
Watermarked Z-Score: 5.98 ↑ PPL: 4.46	`Alan Turing was born in 1912 and died in 1954, at the age of 41. He was the brilliant British scientist and mathematician who is largely credited with being the father of modern computer science. He is known for ...`
(a) Spoofing attack Exploiting robustness Z-Score: 5.98 ↑	`Alan Turing was born in 1950 and died in 1994, at the age of 43. He was the brilliant American scientist and mathematician who is largely credited with being the father of modern computer science. He is known for ...`
(b) Watermark-removal attack Exploiting distribution-preserving PPL: 4.05, Z-Score: 2.40 ↓	`Alan Turing was born in 1912 and died in 1954. He was a mathematician, logician, cryptologist and theoretical computer scientist. He is famous for his work on code-breaking and artificial intelligence, and his ...`
(c) Watermark-removal attack Exploiting public detection API PPL: 4.57, Z-Score: 1.47 ↓	`Alan Turing was born in 1912 and died in 1954. He was an English mathematician, computer scientist, cryptanalyst and philosopher. Turing was a leading mathematician and cryptanalyst. He was one of the key players in ...`

Table 1: Examples generated using Llama-2-7B with/without the KGW watermark Kirchenbauer et al. (2023a) under various attacks. We mark the tokens in the green and red lists (see Appendix C). Z-score reflects the confidence of the watermark and perplexity (PPL) indicates the quality of the text. (a) In the *spoofing attack*, we exploit the robustness property of LLMs by generating incorrect content that appears as watermarked (matching the z-score of the watermarked baseline), potentially damaging the reputation of the LLM. Incorrect tokens modified by the attacker are marked in orange and watermarked tokens in blue. (b-c) In *watermark-removal attacks*, attackers can effectively lower the z-score below the detection threshold while preserving a high quality (low PPL) by exploiting either the (b) distribution-preserving property or (c) public watermark detection API.

as general guidelines to better enhance the security of next-generation LLM watermarking systems. Overall, we make the following contributions:

- We study how watermark *robustness*, despite being a desirable property to mitigate watermark removal attacks, can make the resulting systems highly susceptible to *spoofing attacks*, and show that challenges exist in detecting these attacks given that a single token can render an entire sentence inaccurate (Sec. 2).
- We demonstrate that preserving *distribution* of the output by using multiple watermarking keys can make the system susceptible to *watermark-removal attacks* (Sec. 3). We show both theoretically and empirically that a smaller number of keys may be necessary to defend against removal attacks, potentially at the cost of output quality.
- Finally, we identify that *public watermark detection APIs* can be exploited by attackers to launch both watermark-removal and spoofing attacks (Appendix D). We propose a defense using techniques from differential privacy to effectively counteract spoofing attacks, and recommend setting query rate limits on the detection API and verifying the identity of users as simple mitigations.

Throughout, we explore our attacks on three state-of-the-art watermarks Kirchenbauer et al. (2023a); Zhao et al. (2024); Kuditipudi et al. (2023) and two LLMs (Llama-2-7B Touvron et al. (2023) and OPT-1.3B Zhang et al. (2022))—demonstrating that these vulnerabilities are common to existing LLM watermarks, and providing caution for the field in deploying current solutions.

2 Attacking Robust Watermarks

Developing a watermark that is robust to output perturbations to defend against watermark-removal has been the inspiration for recent works Zhao et al. (2024); Kirchenbauer et al. (2023a); Kuditipudi et al. (2023); Sadasivan et al. (2023); Kirchenbauer et al. (2023b); Piet et al. (2023).

A more robust watermarking scheme can better defend against watermark-removal attacks. However, we note that this same property can be misused by malicious users to launch spoofing attacks. E.g., a small portion of toxic or incorrect content can be inserted into watermarked material, making it seem like generated by a specific watermarked LLM. With a robust watermark embedded, the entire toxic content will still seem watermarked, potentially damaging the reputation of the LLM.

Threat Model. We assume that the attacker can make poly(l) queries to the target watermarked LLM, where l is the token length of the generated content. We also assume that the attacker can edit the generated sentence (e.g., insert or substitute tokens).

Attack Procedure. 1) The attacker queries the target watermarked LLM to receive a high-entropy watermarked sentence $\mathbf{x}_{\text{wm}}$, 2) The attacker edits $\mathbf{x}_{\text{wm}}$ and forms a new piece of text $\mathbf{x}'$ and claims that $\mathbf{x}'$ is generated by the target LLM. The editing method can be defined by the attacker. Simple strategies could be inserting toxic tokens into the watermarked sentence $\mathbf{x}_{\text{wm}}$ at random positions to make the whole content harmful or editing the tokens to make the sentence inaccurate (see the example in Table 1). Please refer to Appendix E for our analysis of the attack feasibility.

2.1 Evaluation

Observation #1

Robust watermarks are susceptible to spoofing attacks.

Experiment Setup. We assess the effectiveness of the spoofing attack by determining the maximum portion of toxic tokens that can be inserted into the watermarked text without altering the watermark detection result. Specifically, we generate a list of 200 toxic tokens and insert them at random positions of the watermarked outputs. We utilize 500 prompts data from OpenGen Krishna et al. (2023) dataset and query the watermarked language models (including Llama-2-7B Touvron et al. (2023) and OPT-1.3B Zhang et al. (2022)) to generate the watermarked outputs. We evaluate three SOTA watermarks including KGW Kirchenbauer et al. (2023a), Unigram Zhao et al. (2024), and Exp Kuditipudi et al. (2023). We choose the default watermarking hyperparameters (see Appendix C).

Evaluation Result. In Fig. 1, we report the maximum portion of the inserted toxic tokens and the confidence of the OpenAI moderation model OpenAI (2023b) in identifying the content as violating their usage policy due to the inserted toxic tokens.

Our findings show that we can insert a significant number of toxic tokens into content generated by all the robust watermarks, with a median portion higher than 20%, i.e., for a 200-token sentence, the attacker can insert a median of 40 toxic tokens into it. These toxic sentences are then identified as violating OpenAI policy rules with high confidence scores, whose median is higher than 0.8 for all the watermarks we study. Please refer to Appendix E and F for more results on analyzing the attack feasibility. This attack can be generalized to all robust watermarks.

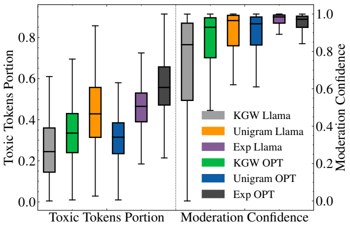


Figure 1: The LHS shows the maximum portion of toxic tokens permitted for insertion. The RHS shows the OpenAI moderation's confidence in detecting the toxic content.

2.2 Discussion

Guideline #1

Robust watermarks may need to compromise on robustness to mitigate the possibility of spoofing attacks.

Spoofing attacks are easy to execute in practice. No existing watermarks consider such attacks during the design or deployment, and existing robust watermarks are inherently vulnerable to such attacks. In particular, we highlight the contradiction between the watermark robustness and the spoofing feasibility: Enhancing robustness makes it more difficult to remove watermarks from edited sentences. However, this feature can be exploited as an attack vector, where the attacker might embed harmful contents into a watermarked sentence and claim that the whole sentence is watermarked. We deem that this attack is challenging to defend against, especially considering the spoofing example in Table 1, where by only editing a single token, the whole content becomes incorrect. It is hard, if not impossible, to detect whether a particular token is from the attacker by using robust watermark detection algorithms. Thus, practitioners should weigh the risks of removal vs. spoofing attacks, and consider reducing watermark robustness to mitigate the possibility of spoofing.

3 Attacking Distribution-Preserving Watermarks

SOTA watermarking schemes Kirchenbauer et al. (2023a); Fairoze et al. (2023); Christ et al. (2023); Kuditipudi et al. (2023); Zhao et al. (2024); Kirchenbauer et al. (2023b) aim to ensure the watermarked text retains its unwatermarked distribution by maintaining an "unbiasedness" property.

As a common practice in cryptography and also suggested by KGW or inherent in prior watermarks (e.g., Exp), the LLM service provider may use multiple watermark keys to enhance security against brute-force attacks in distinguishing the watermarked tokens Sadasivan et al. (2023); Jovanović et al. (2024). However, we demonstrate that using multiple keys can potentially introduce new vulnerabilities and allow malicious users to remove watermarks with only a few queries to the watermarked LLM. The intuition is that, given the quality-preserving nature, attackers can estimate the unwatermarked distribution by making multiple queries to the watermarked LLM under different keys for each token. As this attack estimates the original, unwatermarked distribution, the quality of the generated content is preserved.

Threat Model. We assume multiple watermark keys are utilized to embed the watermark to provide distortion-free guarantees in preserving the output quality or improve the security against brute-force attacks in distinguishing the watermarked tokens' distribution. For a sentence of length l, we assume the attacker can make poly(l) queries to the watermarked LLM under different watermark keys.

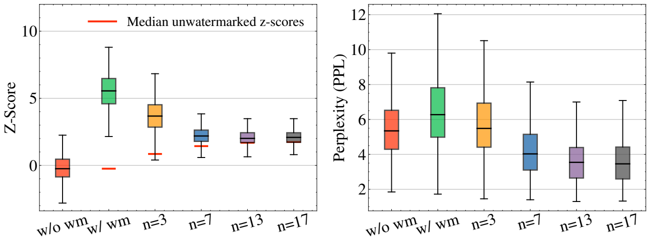
Attack Procedure. An attacker queries a watermarked model with an input $\mathbf{x}$ of length t under n different watermark keys, observing n subsequent tokens $\mathbf{x}_{t+1}$. They then create a frequency histogram of these tokens and sample the most frequent one. This sampled token matches the result of greedy sampling on an unwatermarked output distribution with a nontrivial probability. Consequently, the attacker can progressively eliminate watermarks while maintaining a high quality of the synthesized content. We analyze the number of keys/queries required in Appendix G and H.

3.1 Evaluation

Observation #2

Increasing the number of watermark keys reduces the output distribution shifting and enhances brute-force attack security, but increases vulnerability to watermark-removal attacks.

Experiment Setup. Similar to Sec. 2.1, we evaluate three SOTA watermarks, KGW, Unigram, and Exp on Llama-2-7B and Opt-1.3B models, and OpenGen dataset. We test the detection scores (z-score or p-value) and the output perplexity (PPL) evaluated using GPT3 Ouyang et al. (2022) (see the introduction of PPL in Appendix C). We use the default watermark hyperparameters.



(a) Z-Score.

(b) Perplexity.

Figure 2: Watermark-removal on KGW and Llama-2-7B with different numbers of watermark keys n. Higher z-score reflects more confidence in watermark and lower PPL indicates better sentence quality.

Evaluation Result. As shown in Fig. 2a, we significantly reduce the detection confidence by using more keys. We also present the median detection scores of the unwatermarked content using different numbers of keys in Fig. 2a. Since the detection algorithm returns the highest z-score among all the keys Kirchenbauer et al. (2023a); Kuditipudi et al. (2023), the expected z-scores of unwatermarked content become higher if using more keys. The scores of the watermark-removal closely resemble those of unwatermarked content detection results when we use more than 7 keys. In essence, to remove the watermark, an attacker only needs to *query the watermarked LLM* 7 times for each token under different keys. Fig. 2b suggests that using more keys improves the output content quality. This is because, with a greater number of keys, there's a higher probability for an attacker to accurately estimate the unwatermarked distribution, which is consistent with our analysis in Appendix G. We observe that in practice, 7 keys suffice to produce high-quality content comparable to the unwatermarked content. This observation remains consistent across various watermarks and models, we defer more results to Appendix J.

3.2 Discussion

Guideline #2

Using fewer keys, can potentially introduce distribution shifting and weaken security against brute-force attacks, but it may be necessary to mitigate the watermark-removal attack.

Many watermarking schemes recommend using multiple keys to ensure distortion-free watermarks and enhance security against brute-force attacks. However, we reveal a conflict between preserving output distribution, enhancing security, and the feasibility of removing watermarks in this study. We suggest using fewer keys and compromising slightly on distribution preservation and security against brute-force attacks in practice to achieve a balanced trade-off. Additionally, we recommend that LLM service providers identify potential malicious users and limit their query rates.

4 Conclusion & Discussion

In this work, we reveal and evaluate new attack vectors that exploit the common properties of LLM watermarks. In particular, while these properties may enhance robustness, output quality, and public detection ease, they also allow malicious actors to launch attacks that can easily remove the watermark or damage the model's reputation. Based on the theoretical and empirical analysis of our attacks, we suggest guidelines for designing and deploying LLM watermarks along with possible defenses to establish more reliable watermark embedding and detection systems.

Our work studies the security implications of common LLM watermarking design choices. By developing realistic attacks and defenses and a simple set of guidelines for watermarking in practice, we aim for the work to serve as a resource for the development of secure LLM watermarking systems. Of course, we note that by outlining such attacks, there is a risk that our work may in fact increase the prevalence of watermark removal or spoofing attacks performed in practice. We believe that this is nonetheless an important step towards educating the community about potential risks in watermarking systems and ultimately creating more effective defenses for secure LLM watermarking.

More generally, our work shows that a number of trade-offs exist in LLM watermarking (e.g., between quality, robustness, detection effectiveness, and susceptibility to removal or spoofing attacks). The guidelines we propose provide rough proposals for considering these trade-offs, but we note that how to best navigate each trade-off will depend on the application at hand. Considering strategies to best navigate this space for specific LLM watermarking applications is an important direction for future study.

References

Scott Aaronson. Watermarking of large language models. https://simons.berkeley.edu/talks/scott-aaronson-ut-austin-openai-2023-08-17, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*, 2023.

Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly detectable watermarking for language models. *Cryptology ePrint Archive*, 2023.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. *arXiv preprint arXiv:2312.04469*, 2023.

Emil Julius Gumbel. *Statistical theory of extreme values and some practical applications: a series of lectures*, volume 33. US Government Printing Office, 1948.

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

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1875–1885, 2018.

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

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 23–29 Jul 2023a. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*, 2023b.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Frederick Wieting, and Mohit Iyyer. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=WbFhFvjjKj.

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

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3865–3878, 2018.

Zhe Lin, Yitao Cai, and Xiaojun Wan. Towards document-level paraphrase generation with sentence rewriting and reordering. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 1033–1044, 2021.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. Detectgpt: zero-shot machine-generated text detection using probability curvature. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Ali Naseh, Kalpesh Krishna, Mohit Iyyer, and Amir Houmansadr. Stealing the decoding algorithms of language models. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1835–1849, 2023.

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

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023a.

OpenAI. Openai moderation endpoint. https://platform.openai.com/docs/guides/moderation, 2023b.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. Mark my words: Analyzing and evaluating language model watermarks. *arXiv preprint arXiv:2312.00273*, 2023.

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

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, Miles McCain, Alex Newhouse, Jason Blazakis, Kris McGuffie, and Jasmine Wang. Release strategies and the social impacts of language models, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992*, 2023.

Yihan Wu, Zhengmian Hu, Hongyang Zhang, and Heng Huang. Dipmark: A stealthy, efficient and resilient watermark for large language models. *arXiv preprint arXiv:2310.07710*, 2023.

Hanlin Zhang, Benjamin Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: Impossibility of strong watermarking for generative models. *arXiv preprint arXiv:2311.04378*, 2023.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=SsmT8aO45L`.

A Related Work

Advances in large language models (LLMs) have given rise to increasing concerns that such models may be misused for applications including misinformation, phishing, and academic cheating. In response, numerous recent works have proposed watermarking schemes as a tool for detecting LLM-generated text to mitigate potential misuse Kirchenbauer et al. (2023a); Fairoze et al. (2023); Christ et al. (2023); Kuditipudi et al. (2023); Zhao et al. (2024); Kirchenbauer et al. (2023b); Hu et al. (2023); Wu et al. (2023); Wang et al. (2023). These approaches involve embedding invisible watermarks into the model-generated content, which can then be extracted and verified using a secret watermark key.

Existing watermarking schemes share a few natural goals: (1) the watermark should be *robust* in that it cannot be easily removed; (2) the watermark should *preserve the quality* of the output text (usually quantified via unbiasedness in terms of the token distribution); and (3) the watermark should be easy to *detect* when given new candidate text. Unfortunately, we show that existing methods that aim to achieve these goals also render the resulting systems susceptible to watermark-removal or spoofing attacks.

Removal attacks. Several recent works have highlighted that paraphrasing methods may be used to evade the detection of AI-generated text Krishna et al. (2023); Iyyer et al. (2018); Li et al. (2018); Lin et al. (2021); Zhang et al. (2023), with Krishna et al. (2023); Zhang et al. (2023) demonstrating effective watermark removal using a local LLM. However, these methods require expensive additional training for sentence paraphrasing which significantly impacts sentence quality, or assume a high-quality oracle model to guarantee the output quality is preserved. More importantly, our work differs in that we aim to directly connect and study how the inherent properties of watermarking schemes (such as quality preservation and detection APIs) can inform such removal attacks.

Spoofing attacks. Sadasivan et al. (2023); Gu et al. (2023) are the only works we are aware of that explore spoofing on watermarked LLMs. However, the work of Sadasivan et al. (2023) requires an unrealistic number of queries from attackers (1 million), limiting their method to one watermarking scheme Kirchenbauer et al. (2023a) and making it difficult to generalize to other watermarks. Gu et al. (2023) train a new model to learn the watermarked token distribution, which can be infeasible for computation-bounded attackers, especially given the large number of queries to construct the training data and the cost of training a new LLM. The spoofing attacks we propose in the public detection setting can be generalized across all types of watermarks while requiring only a handful of queries to identify each token. Additionally, we are the first to explore the natural tension that occurs between watermark robustness, which aims to mitigate the potential for removal attacks discussed above, and spoofing—showing that robust watermarks are at an increased risk for spoofing attacks.

B Preliminaries

Notations. We use $\mathbf{x}$ to denote a sequence of tokens, $\mathbf{x}_i \in \mathcal{V}$ is the i-th token in the sequence, and $\mathcal{V}$ is the vocabulary. M_{orig} denotes the original model without a watermark, M_{wm} is the watermarked model, and $sk \in \mathcal{S}$ is the watermark secret key sampled from the key space $\mathcal{S}$.

Language Models. The current state-of-the-art (SOTA) LLMs are auto-regressive models, which predict the next token based on the prior tokens. Below, we define language models (LM):

Definition 1 (LM). *We define a LM without a watermark:*

$$M_{orig} : \mathcal{V}^* \to \mathcal{V}, \tag{1}$$

where the input is a sequence of length t tokens $\boldsymbol{x}$. $M_{orig}(\boldsymbol{x})$ firstly returns the probability distribution for the next token $\boldsymbol{x}_{t+1}$ and then the LLM samples $\boldsymbol{x}_{t+1}$ from this distribution.

Watermarks for LLMs. In this work, we focus on three SOTA decoding-based watermarking schemes: KGW Kirchenbauer et al. (2023a), Unigram Zhao et al. (2024) and Exp Kuditipudi et al. (2023). Informally, the decoding-based watermark is embedded by perturbing the output distribution of the original LLM. The perturbation is determined by secret watermark keys held by the LLM owner. Formally, we define the watermarking scheme:

Definition 2 (Watermarked LLMs). *The watermarked LLM takes token sequence $\boldsymbol{x} \in \mathcal{V}^*$ and secret key $sk \in \mathcal{S}$ as input, and outputs a perturbed probability distribution for the next token. The* *perturbation is determined by sk:*

$$M_{wm} : \mathcal{V}^* \times \mathcal{S} \to \mathcal{V} \tag{2}$$

The watermark detection outputs the statistical testing score for the null hypothesis that the input token sequence is independent of the watermark secret key:

$$f_{\text{detection}} : \mathcal{V}^* \times \mathcal{S} \to \mathbb{R} \tag{3}$$

The output score reflects the confidence of the watermark's existence in the input. Please refer to Appendix C for the details of the three watermarks.

C Watermarking Schemes

In this section, we introduce the three watermarking schemes we evaluate in the paper—KGW Kirchenbauer et al. (2023a), Unigram Zhao et al. (2024), and Exp Kuditipudi et al. (2023). We also introduce the perplexity, a metric to evaluate the sentence quality.

KGW. In the KGW watermarking scheme, when generating the current token $\mathbf{x}_{t+1}$, all the tokens in the vocabulary are pseudorandomly shuffled and split into two lists—the green list and the red list. The random seed used to determine the green and red lists is computed by a watermark secret key sk the prior token $\mathbf{x}_t$ using Pseudorandom functions (PRFs):

$$\textsc{seed} = F_{sk}(\mathbf{x}_t)$$

Then, the seed is used to split the vocabulary into the green and red lists of tokens, with γ portion of tokens in the green list:

$$L_{\text{green}}, L_{\text{red}} = \text{Shuffle}(\mathcal{V}, \textsc{seed}, \gamma)$$

Then, KGW generates a binary watermark mask vector for the current token prediction, which has the same size as the vocabulary. All the tokens in the green list L_{green} have a value 1 in the mask, and all the tokens in the red list have a value 0 in the mask:

$$\textsc{mask} = \text{GenerateMask}(L_{\text{green}}, L_{\text{red}})$$

To embed the watermark, KGW adds a constant to the logits of the LLM's prediction for token $\mathbf{x}_{t+1}$:

$$\textsc{WatermarkedProb} = \text{Softmax}(\text{logits} + \delta \times \textsc{mask}),$$

where the logits is from the LLM, and the δ is the watermark strength. Then the LLM will sample the token $\mathbf{x}_{t+1}$ according to the watermarked probability distribution.

The detection involves computing the z-score:

$$z = \frac{g - \gamma l}{\sqrt{\gamma(1-\gamma)l}},$$

where g is the number of tokens in the green list, l is the total number of tokens in the input token sequence, and γ is the portion of the vocabulary tokens in the green list. Similar to the watermark embedding, the green and red lists for each token position are determined by the watermark secret key and the token prior to the current token in the input token sequence.

Unigram. Similar to KGW, Unigram also splits the vocabulary into green and red lists and prioritizes the tokens in the green list by adding a constant to the logits before computing the softmax. The difference is that Unigram uses global red and green lists instead of computing the green and red lists for each token. That is, the seed to shuffle the list is only determined by the watermark secret key and generated by a Pseudo-Random Generator (PRG):

$$\textsc{seed} = G(sk)$$

Then, similar to KGW, the seed is used to split the vocabulary into the green and red lists of tokens, with γ portion of tokens in the green list:

$$L_{\text{green}}, L_{\text{red}} = \text{Shuffle}(\mathcal{V}, \textsc{seed}, \gamma)$$

The watermark embedding and detection procedures are the same as KGW: Unigram first computes the watermark mask:

$$\textsc{mask} = \text{GenerateMask}(L_{\text{green}}, L_{\text{red}})$$

And then embed the watermark by perturbing the logits of the LLM outputs:

$$\textsc{WatermarkedProb} = \text{Softmax}(\text{logits} + \delta \times \textsc{mask}),$$

where the logits is from the LLM, and the δ is the watermark strength. Then the LLM will sample the token $\mathbf{x}_{t+1}$ according to the watermarked probability distribution.

The detection also computes the z-score:

$$z = \frac{g - \gamma l}{\sqrt{\gamma(1-\gamma)l}},$$

where g is the number of tokens in the green list, l is the total number of tokens in the input token sequence, and γ is the portion of the vocabulary tokens in the green list. According to the analysis in Zhao et al. (2024) and also consistent with our results in Sec. 2.1, by decoupling the green and red lists splitting with the prior tokens, Unigram is twice as robust as KGW. But it's more likely to leak the pattern of the watermarked tokens given that it uses a global green-red list splitting.

Exp. The Exp watermarking scheme from Kuditipudi et al. (2023) is an extension of Aaronson (2023). Instead of using a single key as in KGW and Unigram, the usage of multiple watermark keys is inherent in Exp to provide the distortion-free guarantee. Each key is a vector of size $|\mathcal{V}|$ with values uniformly distributed in $[0, 1]$. That is, $sk = \xi_1, \xi_2, \cdots, \xi_n$, where $\xi_k \in [0, 1]^{|\mathcal{V}|}, k \in [n]$, and n is the length of the watermark keys, default to 256.

For the prediction of the token $\mathbf{x}_{t+1}$, Exp firstly collects the output probability vector $\mathbf{p} \in [0, 1]^{|\mathcal{V}|}$ from the LLM. A random shift $r \overset{\$}{\leftarrow} [n]$ is sampled at the beginning of receiving the prompt. Then the token $\mathbf{x}_{t+1}$ is sampled using the Gumbel trick Gumbel (1948):

$$\mathbf{x}_{t+1} = \arg\max_i \ (\xi_{k,i})^{1/\mathbf{p}_i},$$

where $k = r + t + 1 \bmod n$, i.e., each position uses a different watermark key which determines the uniform distribution sampling used in the Gumbel trick sampling. This method guarantees that the output distribution is distortion-free, whose expectation is identical to the distribution without a watermark given sufficiently large n.

The watermark detection also computes test statistics. The basic test statistics are:

$$\phi = \sum_{t=1}^{l} -\log(1 - \xi_{k,\mathbf{x}_t}),$$

where $k = t \bmod n$. Exp computes the minimum Levenshtein distance using the basic test statistic as a cost (see Sec. 2.4 in Kuditipudi et al. (2023)).

Instead of using single keys as KGW and Unigram, Exp uses multiple keys and incorporates the Gumbel trick to rigorously provide the distortion-free (unbiased) guarantee, whose expected output distribution over the key space is identical to the unwatermarked distribution.

Sentence Quality. Perplexity (PPL) is one of the most common metrics for evaluating language models. It can also be utilized to measure the quality of the sentences Zhao et al. (2024); Kirchenbauer et al. (2023a) based on the oracle of high-quality language models. Formally, PPL returns the following quality score for an input sentence $\mathbf{x}$:

$$\text{PPL}(\mathbf{x}) = \exp\{-\frac{1}{t}\sum_{i=1}^{t} \log[\Pr(\mathbf{x}_i | \mathbf{x}_0, \cdots \mathbf{x}_{i-1})]\} \tag{4}$$

In our evaluation, we utilize the GPT3 Ouyang et al. (2022) as the oracle model to evaluate sentence quality.

Watermark Setups and Hyper-parameters. For KGW Kirchenbauer et al. (2023a) and Unigram Zhao et al. (2024) watermarks, we utilize the default parameters in Zhao et al. (2024), where the watermark strength is $\delta = 2$, and the green list portion is $\gamma = 0.5$. We employ a threshold of $T = 4$ for these two watermarks. For the Exp watermark (referred to as Exp-edit in Kuditipudi et al. (2023)), we use the default parameters, where the watermark key length is $n = 256$ and the block size k defaults to be identical to the token length. We set the p-value threshold for Exp to 0.05 in our experiments. In our experiments, we default to a maximum of 200 new tokens for KGW and Unigram, and 70 for Exp, due to its complexity in the watermark detection. 70 is also the maximum number of tokens the authors of Exp evaluated in their paper Kuditipudi et al. (2023).

D Attacking Watermark Detection APIs

In addition to the robustness (Sec. 2) and quality-preserving (Sec. 3) properties, another common feature of modern LLM watermarks is their ease of detection, allowing the general public to verify if a text is AI-generated Fairoze et al. (2023); Kirchenbauer et al. (2023a); Solaiman et al. (2019); Mitchell et al. (2023). However, this property can also be exploited to launch watermark-removal and spoofing attacks. In the following sections, we first introduce the attack procedures and then propose suggestions and defenses to mitigate these attacks.

D.1 Attack Procedures

Watermark-Removal Attack. For the watermark-removal attack, we consider an attacker who has access to the target watermarked LLM API, and can query the watermark detection results. The attacker feeds a prompt into the watermarked LLM, which generates the response in an auto-regressive manner. For the token $\mathbf{x}_i$ the attacker will generate a list of possible replacements for $\mathbf{x}_i$. This list can be generated by querying the watermarked LLM, querying the local model, or simply returned by the watermarked LLM. In this work, we choose the third approach because of its simplicity and guarantee of synthesized sentences' quality. This is a common assumption made by prior works Naseh et al. (2023), and such an API is also provided by OpenAI ($\text{top\_logprobs} = 5$). Additionally, returning the top logprob values can benefit normal users in understanding the model confidence, debugging and analyzing the model's behavior, customizing sampling strategies, etc.

Consider that the top $L = 5$ tokens and their probabilities are returned to the attackers. The probability that the attacker can find an unwatermarked token in the token candidates' list of length L is $1 - \gamma^L$ for KGW and Unigram, which becomes sufficiently large given $L = 5$ and $\gamma = 0.5$.

The attacker will query the detection using these replacements and sample a token based on their probabilities and detection scores to remove the watermark while preserving a high output quality. See Alg. 1 in Appendix I.

Spoofing Attack. Spoofing attacks follow a similar procedure where the attacker can generate (harmful) content using a local model, and select tokens that yield higher confidence scores upon watermark detection queries. Thanks to the robustness of the LLM watermarks, the attackers don't need to ensure every single token carries a watermark; only that the overall detection confidence score surpasses the threshold, thereby treating synthesized content as if generated by the watermarked LLM. Please refer to Alg. 2 in Appendix I for the detailed algorithm.

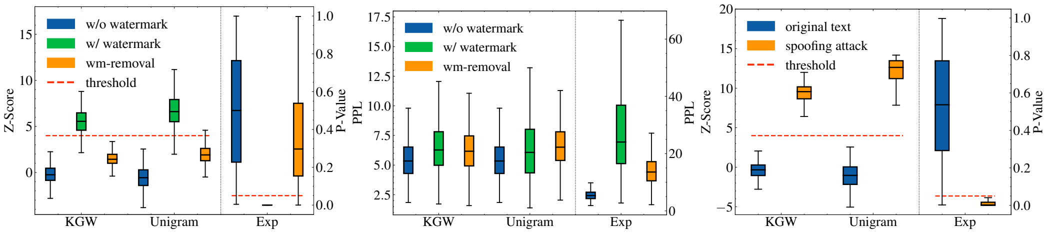
D.2 Evaluation

> Observation #3
>
> Public detection APIs can be exploited to launch both watermark-removal and spoofing attacks.

Experiment Setup. We use the same evaluation setup as Sec. 2.1 and Sec. 3.1. We evaluate the feasibility of the attacks exploiting the detection API on three watermarks (KGW, Unigram, and Exp), two LLMs (Llama-2-13B, OPT-1.3B), and OpenGen dataset. We evaluate the detection scores for both the watermark-removal and the spoofing attacks. We also report the number of queries to the detection API. Furthermore, for the watermark-removal attack, where the attackers care more about the output quality, we report the output PPL. For spoofing attacks, the attackers' local models are Llama-2-7B and OPT-1.3B.

Evaluation Result. The results are shown in Fig. 3. Watermark-removal attack exploiting the detection API significantly reduces detection confidence while maintaining high output quality as shown in Fig. 3a and Fig. 3b. For instance, for the KGW watermark on Llama-2-7B model, we achieve a median z-score of 1.43, which is much lower than the threshold 4. The PPL is also close to the watermarked outputs (6.17 vs. 6.28). We observe that the Exp watermark has a higher PPL than the other two watermarks. This is because that Exp watermark is deterministic, while other watermarks enable random sampling during inference. Our attack also employs sampling based on the token probabilities and detection scores, thus we can improve the output quality for the Exp watermark.

The spoofing attacks also significantly boost the detection confidence even though the content is not from the watermarked LLM, as depicted in Fig. 3c. We report the attack success rate (ASR) and the number of queries for both of the attacks in Table 2. The ASR quantifies how much of the generated

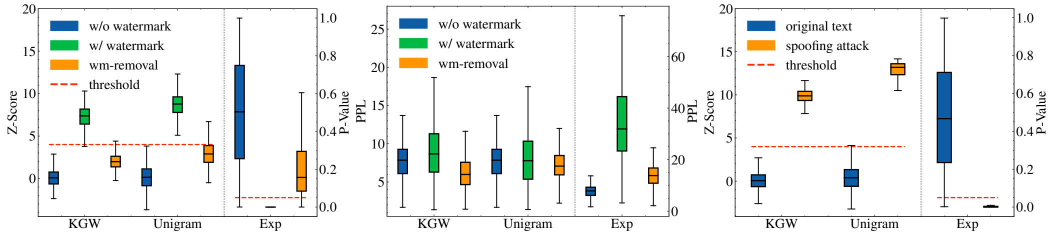


(a) Z-Score/P-Value of wm-removal. **(b)** Perplexity of wm-removal. **(c)** Z-Score/P-Value of spoofing.

Figure 3: Attacks exploiting detection APIs on Llama-2-7B model.

	wm-removal		spoofing	
	ASR	#queries	ASR	#queries
KGW	1.00	2.42	0.98	2.95
Unigram	0.96	2.66	0.98	2.96
Exp	0.96	1.55	0.85	2.89

Table 2: The attack success rate (ASR), and the average query numbers per token for the watermark-removal and spoofing attacks exploiting the detection API on Llama-2-7B model.



(a) Z-Score/P-Value of wm-removal. **(b)** Perplexity of wm-removal. **(c)** Z-Score/P-Value of spoofing.

Figure 4: Attacks exploiting detection APIs on OPT-1.3B model.

	wm-removal		spoofing	
	ASR	#queries	ASR	#queries
KGW	0.99	2.87	1.00	2.96
Unigram	0.77	3.25	1.00	2.97
Exp	0.86	2.07	0.93	2.92

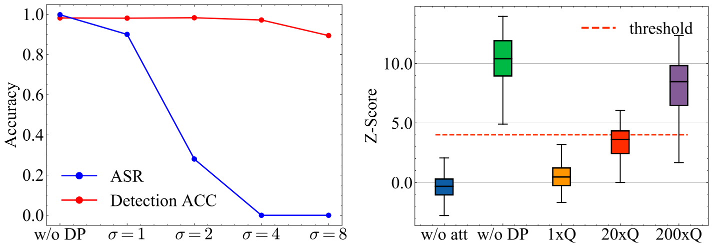
Table 3: The attack success rate (ASR), and the average query numbers per token for the watermark-removal and spoofing attacks exploiting the detection API on OPT-1.3B model.

content surpasses or falls short of the detection threshold. These attacks use a reasonable number of queries to the detection API and achieve a high success rate, demonstrating practical feasibility.

We present the results of watermark-removal and spoofing attacks on OPT-1.3B model in Fig. 4 and Table 3. The results are consistent with the Llama-2-7B model, with all the attack success rates higher than 75% using a small number of queries to the detection API of around 3 per token. The results on OPT-1.3B model further demonstrate the effectiveness of our attacks exploiting the detection API.

D.3 Defending Detection with Differential Privacy

We propose an effective defense using ideas from differential privacy (DP) Dwork et al. (2014) to counteract spoofing attacks exploiting detection API. DP adds random noise to function results evaluated on private dataset. Such that the results from neighboring datasets are indistinguishable. We apply the idea in DP by adding Gaussian noise to the distance score in the watermark detection, and make the detection (ϵ, δ)-DP Dwork et al. (2014). Such that the attackers cannot tell the difference between two queries by replacing a single token in the content, which increases the hardness of



(a) Spoofing attack success rate and detection accuracy.

(b) Z-scores with/without DP and under multiple queries.

Figure 5: Evaluation of DP detection on KGW watermark and Llama-2-7B model. **(a).** Spoofing attack success rate (ASR) and detection accuracy (ACC) without and with DP watermark detection under different noise parameters. **(b).** Z-scores of original text without attack, spoofing attack without DP, and spoofing attacks with DP under different query numbers (marked as 1×Q, 20×Q, and 200×Q). We use the best $\sigma = 4$ from **(a)**.

launching the attacks. In the following, we evaluate the utility of the DP defense and its performance in mitigating spoofing attacks.

Experiment Setup. Firstly, we assess the utility of DP defense by evaluating the accuracy of detecting watermarked and non-watermarked content under various noise scale parameters. Next, we evaluate the efficacy of the spoofing attack against differential privacy detection defense using the same method as in Appendix D.1. We select the optimal noise scale parameter that provides the best defense while keeping the drop in utility accuracy within 2%. Considering an attacker can average multiple queries to reduce noise and estimate original scores without DP protection, we evaluate the spoofing attack again using 20 × and 200 × queries to detection APIs with the optimal noise scale.

Evaluation Result. As shown in Fig. 5a, with a noise scale of $\sigma = 4$, the DP detection's accuracy drops from the original 98.2% to 97.2% on KGW watermark and Llama-2-7B model, while the spoofing attack success rate becomes 0% using the same attack procedure as Appendix D.1. We further evaluate the attacker averaging over 20 and 200 queries to remove the noise in Fig. 5b. We show that with 20× queries, even though the z-scores are much higher than the case with 1× query, its attack success rate is around 50%, which is still significantly lower than that without DP protection. Even with an extremely large number of 200× queries, the attacker cannot achieve the same result as the scenario without DP defense. The results are consistent for Unigram and Exp watermarks and OPT-1.3B model as shown in Appendix K, which illustrates that the DP defense has a great utility-defense trade-off, with a negligible accuracy drop, and significantly mitigates the spoofing attacks.

D.4 Discussion

Guideline #3

DP techniques can effectively mitigate spoofing attacks exploiting detection APIs. Detection services should identify malicious behaviors and limit query rates from potential attackers, and also verify the users' identity.

The detection API, available to the public, aids users in differentiating between AI-generated and human-created materials. However, it can be exploited by malicious users to gradually remove watermarks or launch spoofing attacks. We propose a defense by employing the ideas in differential privacy, which significantly increases the difficulty for attackers to launch spoofing attacks. However, this method is less effective against watermark-removal attacks that exploit the detection API because attackers' actions will be close to random sampling, which, even though with lower success rates, remains an effective way of removing watermarks. Therefore, we leave developing a more powerful defense mechanism against watermark-removal attacks exploiting detection API in future work. We recommend that companies providing these detection services should detect and curb malicious behavior by limiting query rates from potential attackers, and also verify the identity of the users to protect against Sybil attacks.

E SPOOFING ATTACK FEASIBILITY BY EXPLOITING WATERMARK ROBUSTNESS

We first define the watermark robustness:

Definition 3 (Watermark robustness)**.** *Given a watermarked text* $\boldsymbol{x}$*, for all its neighboring text within the* ϵ *editing distance, the probability that the detection fails to detect the edited text is bounded by* δ*, given the detection confidence threshold* T*:*

$$\forall \boldsymbol{x}, \boldsymbol{x}' \in \mathcal{V}^*,\ \Pr[f_{detection}(\boldsymbol{x}', sk) < T] < \delta, \quad s.t. \quad f_{detection}(\boldsymbol{x}, sk) \geq T,\ d(\boldsymbol{x}, \boldsymbol{x}') \leq \epsilon,$$

Attack Feasibility Analysis. We study the bound on the maximum number of tokens that are allowed to be inserted into a watermarked sentence, and we present the following theorem on Unigram due to its clean robustness guarantee:

Theorem 1 (Maximum insertion portion)**.** *Consider a watermarked token sequence* $\boldsymbol{x}$ *of length* l*. The Unigram z-score threshold is* T*, the portion of the tokens in the green list is* γ*, the detection z-score of* $\boldsymbol{x}$ *is* z*, and the number of inserted tokens is* s*. Then to guarantee the expected z-score of the edited text is greater than* T*, it suffices to guarantee:*

$$\frac{s}{l} \leq \frac{z^2 - T^2}{T^2} \tag{5}$$

Proof. Now we present the proof of Theorem 1. According to Eq. 5, as long as the number of inserted toxic tokens is bounded by $l\frac{z^2-T^2}{T^2}$, the attacker can execute a spoofing attack to generate toxic content with the target watermark embedded. For token insertion editing, the editing distance bound (Def. 3) for a sentence is $\epsilon = l\frac{z^2-T^2}{T^2}$. A stronger watermark increases the ease of launching spoofing attacks by allowing more toxic tokens to be inserted. This conclusion applies universally to all robust watermarking schemes. If a watermark is robust, such attacks are inevitable and extremely difficult to detect, as even one toxic token can render the entire content harmful or inaccurate.

In the following, we prove the bound on the maximum number of tokens that are allowed to be inserted into a watermarked sentence for Unigram Zhao et al. (2024).

Proof. Recall that the watermarking schemes' detections usually involve computing the statistical testing. Unigram splits the vocabulary into two lists—the green list and the red list. It prioritizes the tokens in the green list during watermark embedding, and the detection computes the z-score:

$$z = \frac{g - \gamma l}{\sqrt{\gamma(1-\gamma)l}},$$

where g is the number of tokens in the green list, l is the total number of tokens in the input token sequence, and γ is the portion of the vocabulary tokens in the green list. Let the number of the inserted toxic tokens be s. Since toxic tokens are independent of the secret key sk, the expected new z-score z' is:

$$\mathbb{E}(z') = \frac{g + \gamma s - \gamma(l+s)}{\sqrt{\gamma(1-\gamma)(l+s)}} = z\sqrt{\frac{l}{l+s}},$$

To guarantee that $\mathbb{E}(z') \geq T$, it suffices to guarantee

$$\frac{s}{l} \leq \frac{z^2 - T^2}{T^2}$$

□

F VALIDATION OF THEOREM 1

In this section, we validate Theorem 1 by using watermarked texts of varying lengths l and z-scores z to study the relationship between $\frac{s}{l}$ and $\frac{z^t - T^2}{T^2}$ of Unigram watermark. The results are shown in Fig. 6. As anticipated, 85.78% of the maximum allowable tokens to be inserted into the watermarked content satisfy Eq. 5. Given that this equation analyzes expected s/l, a small portion of outliers is reasonable. We primarily visualize this result for Unigram due to its clean robustness guarantee. Other watermarks can also reach similar conclusions, but their bounds on s are either complex Kirchenbauer et al. (2023a) or lack a closed form Kuditipudi et al. (2023), making them difficult to visualize. Our empirical findings in Fig. 1 sufficiently prove an attacker can insert nontrivial portions of toxic or incorrect tokens into the watermarked text to launch the spoofing attack, which can be generalized across all robust watermarking schemes.

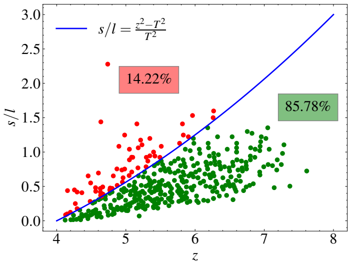


Figure 6: The relationship between s/l and z. The data points are evaluated on Unigram using LLAMA-2-7B and 500 samples from OpenGen dataset.

G Probability Bound of Unwatermarked Token Estimation for KGW and Unigram

Key/Query Number Analysis. Now we analyze the number of required queries under different keys to estimate the token with the highest probability without a watermark. We have the following probability bound for KGW and Unigram, and present the bound for Exp in Appendix H.

Theorem 2 (Probability bound of unwatermarked token estimation)**.** *Suppose there are n observations under different keys, the portion of the green list in KGW or Unigram is γ. Then the probability that the most frequent token is the same as the original unwatermarked token is*

$$1 - \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{k} \gamma^k (1-\gamma)^{n-k} \times p(k), \tag{6}$$

where $p(k) = 1 - \left(\sum_{m=0}^{k-1} \binom{n-k}{m} \gamma^m (1-\gamma)^{n-k-m} \right)^c$, c is the number of other tokens whose watermarked probability can exceed that of the highest unwatermarked token.

In a practical scenario where $n = 13, \gamma = 0.5$, and $c = 3$, Theorem 2 suggests that the attacker has a probability of 0.71 in finding the token with the highest unwatermarked probability. This implies that we can successfully remove watermarks from over 71% of tokens using a small number of observations under different keys ($n = 13$), yielding high-quality unwatermarked content.

Proof. Now we present the proof for Theorem 2.

Proof. Recall that KGW and Unigram randomly split the tokens in the vocabulary into the green list and the red list. We consider greedy sampling, where the token with the highest (watermarked) probability is sampled. We have n independent observations under different watermark keys. For each key, the token $\mathbf{x}_i$ with the highest unwatermarked probability in the green list is γ. As long as $\mathbf{x}_i$ is the green list, the greedy sampling will always yield $\mathbf{x}_i$ since the watermarks add the same constant to all the tokens' logits in the green list.

Thus, the probability that the most frequent token among these n observations is $\mathbf{x}_i$ is at least:

$$1 - \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{k} \gamma^k (1-\gamma)^{n-k},$$

which is the probability that $\mathbf{x}_i$ is in the green list for at least half of the n keys.

For another token $\mathbf{x}_j$ whose probability can exceed $\mathbf{x}_i$, if $\mathbf{x}_j$ is in the green list and $\mathbf{x}_i$ is in the red list. Then if $\mathbf{x}_i$ is in the green list for k keys, the probability that $\mathbf{x}_j$ is in the green list for at least k keys among the other $n-k$ keys is:

$$1-\sum_{m=0}^{k-1}\binom{n-k}{m}\gamma^m(1-\gamma)^{n-k-m}$$

Consider we have c such tokens having the potential to exceed $\mathbf{x}_i$. Then at least one of the c tokens is in the green list for at least k keys among the other $n-k$ keys:

$$1-\Big(\sum_{m=0}^{k-1}\binom{n-k}{m}\gamma^m(1-\gamma)^{n-k-m}\Big)^c$$

Thus, with all the above analysis, we have that if there are c tokens that have the potential to exceed the probability of the token with the highest unwatermarked probability (i.e., $\mathbf{x}_i$), the probability that the most frequent token among the n observations is the same as $\mathbf{x}_i$ is:

$$1-\sum_{k=0}^{\lfloor n/2\rfloor}\binom{n}{k}\gamma^k(1-\gamma)^{n-k}\times\left(1-\Big(\sum_{m=0}^{k-1}\binom{n-k}{m}\gamma^m(1-\gamma)^{n-k-m}\Big)^c\right),$$

which concludes the proof. □

H Probability Bound of Unwatermarked Token Estimation for Exp

In this section, we present and prove the probability bound of unwatermarked token estimation for the Exp watermark Kuditipudi et al. (2023).

Theorem 3 (Probability bound of unwatermarked token estimation for Exp)**.** *Suppose there are n observations under different keys, the highest probability for the unwatermarked tokens is p. Then the probability that the most frequently appeared token among the n observations is the same as the original unwatermarked token with the highest probability is:*

$$1-\sum_{k=0}^{\lfloor n/2\rfloor}\binom{n}{k}p^k(1-p)^{n-k} \tag{7}$$

Proof. The proof of Theorem 3 is straightforward. As we have introduced in Appendix C, the Exp watermark employs the Gumbel trick sampling Gumbel (1948) when embedding the watermark. Thus, the probability that we observe the token whose original unwatermarked probability is p is exactly p for each of the independent keys. Thus, if we make n observations under different keys, then at least half of them yield the token with the highest original probability p:

$$1-\sum_{k=0}^{\lfloor n/2\rfloor}\binom{n}{k}p^k(1-p)^{n-k},$$

which concludes the proof. □

I Algorithms of Attacks Exploiting the Detection API

In this section, we provide the detailed algorithm of the attacks exploiting the detection API as we have introduced in Appendix D. Specifically, we present the algorithm for watermark-removal attack exploiting the detection API in Alg. 1 and the algorithm for spoofing attack exploiting the detection API in Alg. 2.

Algorithm 1 Watermark-removal attack exploiting the detection API.

```
Input: Prompt $\mathbf{x}_{\text{prompt}}$, watermarked LLM $M_{\text{wm}}$, detection API $f_{\text{detection}}$, maximum output token number $m \geq 2$
Let $k \leftarrow 5$, $\mathbf{x}_1 \sim M_{\text{wm}}(\mathbf{x}_{\text{prompt}})$
for $t = 2$ to $m$ do
    $(\mathbf{x}_t^1, \mathbf{x}_t^2, \cdots, \mathbf{x}_t^k), (\mathbf{p}_t^1, \mathbf{p}_t^2, \cdots, \mathbf{p}_t^k) \leftarrow M_{\text{wm}}(\mathbf{x}_{prompt}||\mathbf{x}_1 \cdots \mathbf{x}_{t-1})$    {The watermarked LLM returns the top $k$ tokens and their corresponding probabilities in descending order.}
    for $i = 1$ to $k$ do
        $d_i \leftarrow f_{\text{detection}}(\mathbf{x}_1|| \cdots ||\mathbf{x}_{t-1}||\mathbf{x}_t^i)$
    $d_{\text{min}} \leftarrow \min(d_1, d_2, \cdots, d_k)$, $l_{\text{candidate}} \leftarrow$ empty    {Get the detection score with the lowest confidence.}
    for $i = 1$ to $k$ do
        if $d_{\text{min}} = d_i$ then
            $l_{\text{candidate}} \leftarrow l_{\text{candidate}}||\mathbf{x}_t^i$    {Get all the tokens with the lowest detection confidence.}
    if $\mathbf{x}_t^1 \in l_{\text{candidate}}$ then
        $j \leftarrow 0$    {If the token with the highest probability (the first token) is in the list, output that token.}
    else
        $c \leftarrow 1$
        for $\mathbf{x}_t^i \in l_{\text{candidate}}$ do
            $\mathbf{p}_t^i \leftarrow \mathbf{p}_t^1/c$    {Update the tokens' probabilities that have lowest detection confidence scores.}
            $c \leftarrow c + 1$
        $\mathbf{p}_t^1 \leftarrow 0$
        $j \leftarrow$ Sample$(\mathbf{p}_t^1, \cdots, \mathbf{p}_t^k)$    {Sample the tokens according to the updated probabilities.}
    $\mathbf{x}_t \leftarrow \mathbf{x}_t^j$
Return $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$
```

Algorithm 2 Spoofing attack exploiting the detection API.

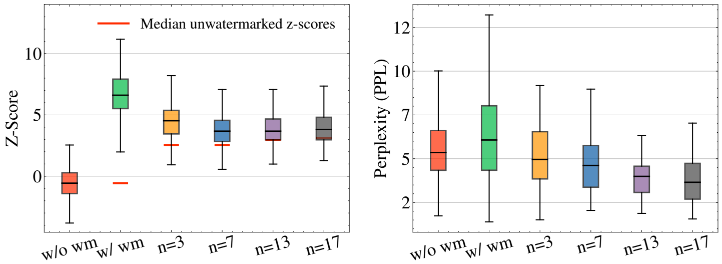
```
Input: Prompt $\mathbf{x}_{\text{prompt}}$, local LLM $M$, detection API $f_{\text{detection}}$, maximum output token number $m$
Let $k \leftarrow 3$
for $t = 1$ to $m$ do
    $(\mathbf{x}_t^1, \mathbf{x}_t^2, \cdots, \mathbf{x}_t^k), (\mathbf{p}_t^1, \mathbf{p}_t^2, \cdots, \mathbf{p}_t^k) \leftarrow M(\mathbf{x}_{prompt}||\mathbf{x}_1 \cdots \mathbf{x}_{t-1})$ {The local LLM returns the top $k$ tokens and their corresponding probabilities in descending order.}
    for $i = 1$ to $k$ do
        $d_i \leftarrow f_{\text{detection}}(\mathbf{x}_1|| \cdots ||\mathbf{x}_{t-1}||\mathbf{x}_t^i)$
    $j \leftarrow \arg\max(d_1, d_2, \cdots, d_k)$    {Get the token resulting in the highest confidence.}
    $\mathbf{x}_t \leftarrow \mathbf{x}_t^j$
Return $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$
```

J Additional Results of Watermark-Removal Attacks Exploiting Quality-Preserving Property

In this section, we provide more evaluation results of the watermark-removal attack exploiting the quality-preserving property (see Sec. 3) on all three watermarks (KGW, Unigram, and Exp) and two models (Llama-2-7B and OPT-1.3B). The results are shown in Fig. 7, 8, 9, 10, 11. For KGW watermark on OPT-1.3B model and Unigram watermark on Llama-2-7B and OPT-1.3B models, we have consistent observations with the KGW watermark on Llama-2-7B as we present in Sec. 3.1, demonstrating the effectiveness of our attacks. For the Exp watermark, our results in Fig. 8 and Fig. 11 also show the watermark can be easily removed using multiple queries to estimate the unwatermarked tokens.

For Exp watermark Kuditipudi et al. (2023), the use of multiple watermark keys is inherent in its design, which is default to 256. Thus, Exp has the same unwatermarked detection results for various numbers of queries under different watermark keys. From the results in Fig. 8 and Fig. 11, we conclude that using $n = 13$ queries under different keys, the resulting p-value is very close to that of the content without watermark and is significantly different from the watermarked p-value, which shows that we can effectively remove the watermark using 13 queries for each token. We note that for Exp, the perplexity of watermarked content is significantly higher than that of unwatermarked content. This is primarily because Exp does not allow sampling in watermark embedding, which becomes a deterministic algorithm when the key is fixed. Conversely, our watermark-removal attack generates content with much lower perplexity, making it comparable to unwatermarked content when query number under different keys exceeds 13. This can be attributed to our attack functioning as a layer of random sampling. Unlike greedy sampling methods, we have a probability to sample the token with the highest unwatermarked probability (refer Sec. 2, Appendix G, and Appendix H).

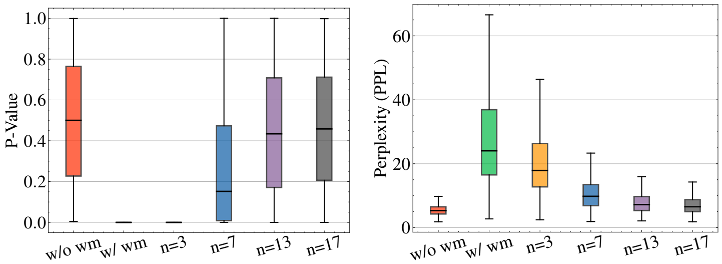
The results of the three watermarks and two models prove that the watermark-removal attack exploiting the quality-preserving property using multiple keys can effectively eliminate the watermarks while maintaining high output quality. We anticipate that this attack can be generalized to all watermarking schemes that have quality-preserving properties.



(a) Z-Score.

(b) Perplexity.

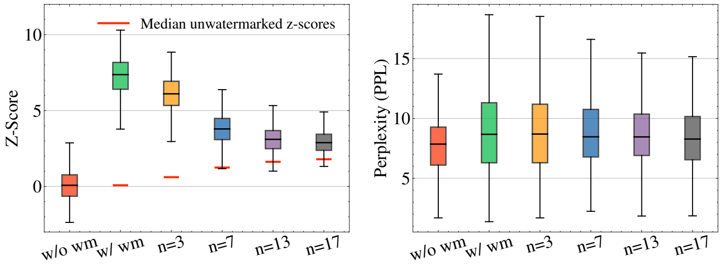
Figure 7: Watermark-removal on Unigram watermark Zhao et al. (2024) and Llama-2-7B model with multiple watermark keys.



(a) P-Value.

(b) Perplexity.

Figure 8: Watermark-removal on Exp watermark Kuditipudi et al. (2023) and Llama-2-7B model with multiple watermark keys.



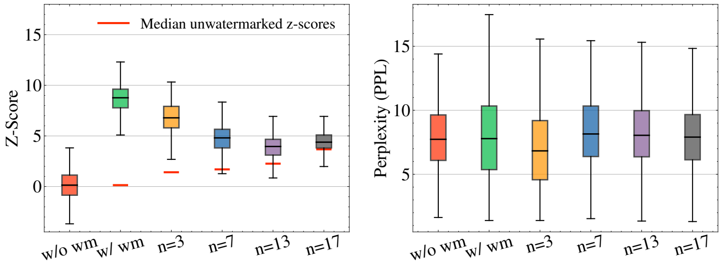
(a) Z-Score.

(b) Perplexity.

Figure 9: Watermark-removal on KGW watermark Kirchenbauer et al. (2023a) and OPT-1.3B model with multiple watermark keys.

K Additional Results of DP Defense

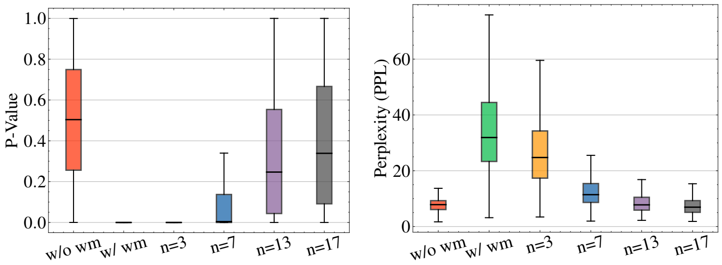
We present additional evaluation results of our defense technique that enhances the watermark detection by utilizing the techniques of differential privacy (see Appendix D). Consistent with Appendix D.3, we evaluate the utility of the DP defense as well as its performance in mitigating the spoofing attack exploiting the detection API. The results are shown in Fig. 12, 13, 14, 15, 16.



(a) Z-Score.

(b) Perplexity.

Figure 10: Watermark-removal on Unigram watermark Zhao et al. (2024) and OPT-1.3B model with multiple watermark keys.



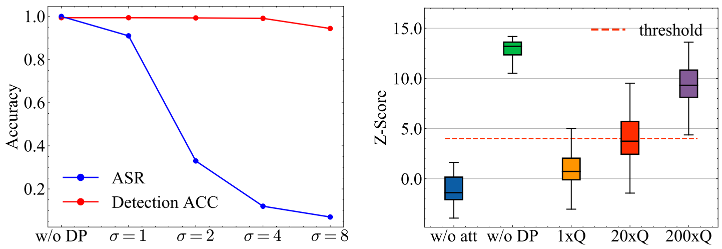
(a) P-Value.

(b) Perplexity.

Figure 11: Watermark-removal on Exp watermark Kuditipudi et al. (2023) and OPT-1.3B model with multiple watermark keys.

We first identify the optimal noise scale parameter σ based on its detection accuracy and attack success rate, aiming for a drop in detection accuracy within 2% and the lowest attack success rate. Then we assess the performance of the defense where the attackers average results from multiple queries to the detection API. Our findings across three watermarks and two models consistently demonstrate that we can significantly reduce the attack success rate to around or below 20% in single query scenarios. Even when an attacker uses multiple queries to reduce noise, we can still limit their success rate to approximately 50%. Despite an attacker potentially achieving high success rates with a large number of queries ($200\times$ more queries in our experiments), their resulting detection confidence scores remain lower than those without any defense.

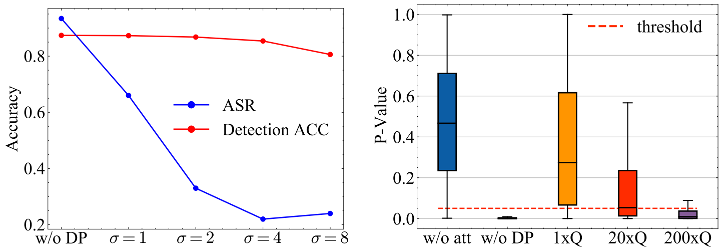
Our defense can be generalized to all LLM watermarking schemes. It allows us to substantially mitigate spoofing attacks exploiting the detection API while having a negligible impact on utility.



(a) Detection accuracy and spoofing attack success rate.

(b) Z-scores with/without DP and under multiple queries.

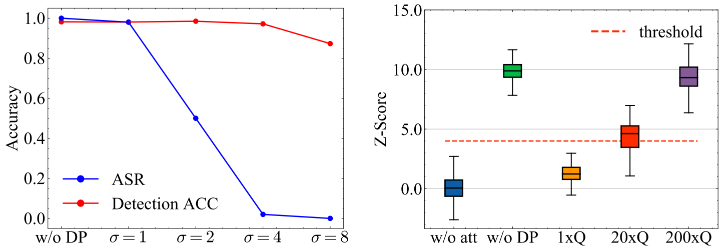
Figure 12: Evaluation of DP watermark detection on Unigram watermark and Llama-2-7B model. **(a).** Detection accuracy and spoofing attack success rate without and with DP watermark detection under different noise parameters. **(b).** Z-scores of original text without attack, spoofing attack without DP, and spoofing attacks with DP under different query numbers. We use the best $\sigma = 4$ from **(a)**.



(a) Detection accuracy and spoofing attack success rate.

(b) P-values with/without DP and under multiple queries.

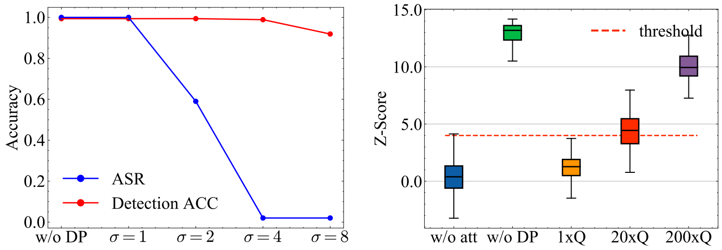
Figure 13: Evaluation of DP watermark detection on Exp watermark and Llama-2-7B model. **(a).** Detection accuracy and spoofing attack success rate without and with DP watermark detection under different noise parameters. **(b).** Z-scores of original text without attack, spoofing attack without DP, and spoofing attacks with DP under different query numbers. We use the best $\sigma = 4$ from **(a)**.



(a) Detection accuracy and spoofing attack success rate.

(b) Z-scores with/without DP and under multiple queries.

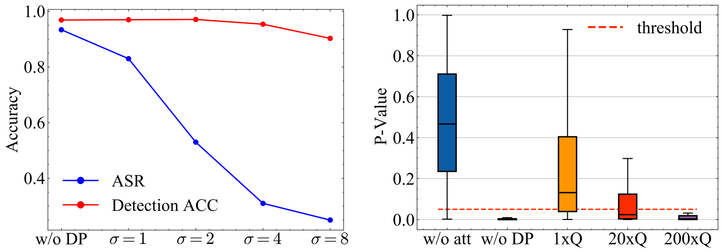
Figure 14: Evaluation of DP watermark detection on KGW watermark and OPT-1.3B model. **(a).** Detection accuracy and spoofing attack success rate without and with DP watermark detection under different noise parameters. **(b).** Z-scores of original text without attack, spoofing attack without DP, and spoofing attacks with DP under different query numbers. We use the best $\sigma = 4$ from **(a)**.



(a) Detection accuracy and spoofing attack success rate.

(b) Z-scores with/without DP and under multiple queries.

Figure 15: Evaluation of DP watermark detection on Unigram watermark and OPT-1.3B model. **(a).** Detection accuracy and spoofing attack success rate without and with DP watermark detection under different noise parameters. **(b).** Z-scores of original text without attack, spoofing attack without DP, and spoofing attacks with DP under different query numbers. We use the best $\sigma = 4$ from **(a)**.



(a) Detection accuracy and spoofing attack success rate.

(b) P-values with/without DP and under multiple queries.

Figure 16: Evaluation of DP watermark detection on Exp watermark and OPT-1.3B model. **(a).** Detection accuracy and spoofing attack success rate without and with DP watermark detection under different noise parameters. **(b).** Z-scores of original text without attack, spoofing attack without DP, and spoofing attacks with DP under different query numbers. We use the best $\sigma = 4$ from **(a)**.