Toward Breaking Watermarks in Distortion-free Large Language Models

Shayleen Reynolds[1], Saheed Obitayo[1], Niccolò Dalmasso[1], Dung Daniel T. Ngo[1], Vamsi K. Potluru[1], Manuela Veloso[1]

[1]J.P. Morgan AI Research, New York

{shayleen.reynolds, saheed.obitayo, niccolo.dalmasso, tuandung.ngo, 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. (2023), which perturbs the logit vector before sampling. In this work, we focus on reverse-engineering the other prominent LLM watermarking scheme, distortion-free watermarking (Kuditipudi et al. 2024), 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. Specifically, we propose a mixed integer linear programming framework that accurately estimates the secret key used for watermarking using only a few samples of the watermarked dataset. Our initial findings challenge the current theoretical claims on the robustness and usability of existing LLM 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 and other LLMs can now generate text that closely resembles human-written content. However, as generative models have been rapidly adopted by both businesses and individuals, their 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 recently emerged (Kirchenbauer et al. 2023; Kuditipudi et al. 2024; Aaronson 2023; Piet et al. 2024; Zhang et al. 2024a; Ning et al. 2024). The primary strategy in this research involves embedding a *hidden* signal (i.e., a secret watermark key) within the generated text, which can later be reliably detected by any third party who possesses knowledge of the secret watermark key.

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 (Jovanović, Staab, and Vechev 2024; Zhang et al. 2024b; Pang et al. 2024; Wu and Chandrasekaran 2024; Gloaguen et al. 2024a,b). Previous research on LLM watermarking often focuses on common attacks, such as deletion, insertion, or substitution, to simulate the behavior of users attempting to evade content detectors. For instance, a student might slightly modify a machine-generated essay, altering a few sentences with the hope of avoiding 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 when it is not. If large volumes of 'spoofed' content can be generated with minimal computational effort, the watermark becomes effectively useless, undermining its intended purpose and damaging the reputation of LLM providers by falsely attributing harmful or incorrect content to them.

Prior work on *watermark stealing* attacks mostly studies the distribution-modifying algorithm by Kirchenbauer et al. (2023). In contrast, our focus is on distribution-free watermarking (Kuditipudi et al. 2024), which does not change the underlying token distribution. A major difference between the two watermarking techniques is that Kuditipudi et al. (2024) uses a randomized watermark key, creating a correlation between the LLM-generated text and this secret key. During detection, a third party with knowledge of this secret watermark key can efficiently check for this correlation and verify whether the text is watermarked or not. With this approach in mind, we propose a mixed integer linear programming model that can accurately estimate the secret watermark key and enable 'spoofing' attacks with only a few samples from the watermarked LLM. Overall, we make the following contributions:

- We provide a framework that accurately estimates the secret watermark key used by the distortion-free watermarking algorithm (Kuditipudi et al. 2024). We show that our mixed integer linear programming is robust to watermarked input after it is shifted and corrupted by the LLM user.
- With this secret watermark key estimation, we demon-

Algorithm 1: Binary watermarked text generation

Input: Watermark key sequence $\xi \in \Xi^n$, generation length m, language model p, token sets $\mathcal{R}$ and $\mathcal{G}$ over vocabulary $\mathcal{V}$.
Output: Generated string $y \in \mathcal{V}^m$.

for $j \in [m]$ **do**
  $\mathbb{P}_j(\mathcal{G}) = \sum_{x_l \in \mathcal{G}} p(x_l|y_{:i-1})$
  if $\xi_j > \mathbb{P}_j(\mathcal{G})$ **then**
    $y_j = 1$
  else
    $y_j = 0$
return Generated binary string y

strate a simple 'spoofing' attack with a high success rate for both the stylized setting of binary vocabulary and a real-world LLM.

2 Methodology

This section outlines the process of generating watermarked text from an LLM and converting it into binary sequences, introduces the mixed integer linear programming approach for secret watermark key estimation, and details the detection procedure for watermarked text.

2.1 Generation and Interaction Protocol

Let $\mathcal{V}$ be a discrete set of vocabulary. Let $\mathcal{R}$ and $\mathcal{G}$ represent two disjoint sets of vocabulary tokens, such that $\mathcal{R} \cap \mathcal{G} = \emptyset$ and $\mathcal{R} \cup \mathcal{G} = \mathcal{V}$. Let $p : \mathcal{V}^* \to \Delta(\mathcal{V})$ be an autoregressive *language model* that maps a string of arbitrary length to a distribution over the vocabulary $\Delta(\mathcal{V})$. Given a prefix $x \in \mathcal{V}^*$, we write $p(\cdot|x)$ as the distribution of the next token. Let Ξ denote the space of watermark key elements; for simplicity we assume each element of a random key sequence $\{\xi_i\}_{i=1}^m \in [0, 1]$. The interaction protocol is as follows:

1. The LM provider shares a random key sequence $\xi \in \Xi$ with the detector, and the sets $\mathcal{R}$ and $\mathcal{G}$.
2. The user sends a prompt $x \in \mathcal{V}^*$ to the LM provider.
3. The LM provider generates binary text $Y \in \mathcal{V}^*$ as $Y = \text{generate}(x, \xi, \mathcal{R}, \mathcal{G})$.
4. The user publishes text $\widetilde{Y}$, which is either 1) as-is or edited version of Y or 2) text independent of Y.
5. The detector determines if $\widetilde{Y}$ is watermarked or not.

Algorithm 1 provides detail on the binary watermarked text generation. In order to convert tokens to binary values, for every token generation step we calculate the probability mass for all tokens in the sets $\mathcal{R}$ and $\mathcal{G}$, namely $\mathbb{P}(\mathcal{R})$ and $\mathbb{P}(\mathcal{G})$[1]. The generation of the j^{th} binary token is equal to $y_i = \mathbb{I}\left(\xi_j > \mathbb{P}_j(\mathcal{G})\right)$, where $\mathbb{I}(\cdot)$ is the indicator function. Note that the binary conversion is a simplifying step we adopt in this work. One could generate text following Algorithm 1 by sampling the token y_j from the $\mathcal{G}$ set of tokens if $\xi_j > \mathbb{P}(\mathcal{G})$, and from the $\mathcal{R}$ set if not.

[1]By construction, $\mathbb{P}(\mathcal{R}) = 1 - \mathbb{P}(\mathcal{G})$.

2.2 Watermark Key Estimation

Given n samples of generated binary strings $\{y_i\}_{i=1}^n$ of length m, we propose to use a linear programming approach to estimate the underlying watermark key ξ^* used for the generation of those samples. For each watermark key ξ^*_j in the sequence, our proposed approach estimates a lower bound $\underline{\xi}^*_j$ and an upper bound $\bar{\xi}^*_j$ to construct an estimation interval $[\underline{\xi}^*_j, \bar{\xi}^*_j]$, with the estimated key selected to be the mid-point of this interval.

We consider 3 cases, which methodologically build on each other:

1. *no alteration:* the available binary strings y_i are available as-generated,
2. *watermark key shifting:* the elements of underlying key ξ^* have been shifted by an unknown amount k_i before the generation of each binary string y_i, and
3. *corruption:* some of the binary strings y_i are corrupted, and hence not reliable for estimation.

No Alteration Case. For ease of notation, let the probability mass of the set $\mathcal{G}$ for the i^{th} sample in the j^{th} token be defined as q^i_j. We solve the following linear program to estimate simultaneously all the lower bounds $\underline{\xi}^*$ for the watermark key sequence:

$$\underline{\xi}^* = \underset{\underline{\xi}^*_j \in \mathbb{R}^m}{\text{minimize}} \quad \sum_{j=1}^{m} \underline{\xi}^*_j \qquad (1)$$
$$\text{s.t.} \quad \underline{\xi}^*_j \geq \mathbb{I}(y^i_j = 1) q^i_j \quad \forall i, j$$

The form of the constraint mirrors the generation process of the binary string y. For the j^{th} token in the i^{th} sample to be $y^i_j = 1$, then it must be that $\xi^*_j > q^i_j$, so q^i_j should be included as a lower bound for ξ^*_j. Conversely, if $y^i_j = 0$, then the lower bound should be 0, i.e., the i^{th} sample does not provide any information on lower bounding ξ^*_j. In other words, this linear program finds the maximum of the lower bounds given by the samples y_i.

For estimating the upper bounds, we solve an equivalent linear program to the above:

$$\bar{\xi}^* = \underset{\bar{\xi}^*_j \in \mathbb{R}^m}{\text{maximize}} \quad \sum_{j=1}^{m} \bar{\xi}^*_j \qquad (2)$$
$$\text{s.t.} \quad \bar{\xi}^*_j \leq \mathbb{I}(y^i_j = 0)(q^i_j - 1) + 1 \quad \forall i, j,$$

for which the constraints again include the information carried by the samples y_i for upper bounding ξ^*.

Watermark Key Shifting Case. In this setup, each sample watermark key might have been shifted by an unknown amount $k_i \in [0, ...m] \subset \mathbb{N}_0$, i.e., for the i^{th} sample the watermark key would be $\xi^*_0 = \xi^*_{0+k_i}$[2]. To account for that,

[2]We assume the watermark key sequence starts from the beginning if the sum of the indexes exceeds the length of the watermark key.

we introduce a set of binary variables $z_i^k \in \{0,1\}$ in the linear program, where $z_i^k = 1$ if the sample i has been shifted by an amount k, and $z_i^k = 0$ otherwise. The resulting lower bounds mixed integer linear program is as follows:

$$\underline{\xi}^* = \underset{\underline{\xi}_j^* \in \mathbb{R}^m \; z^0,\dots,z^m \in \{0,1\}^n}{\text{minimize}} \quad \sum_{j=1}^{m} \underline{\xi}_j^* \tag{3}$$

$$\text{s.t.} \qquad \underline{\xi}_j^* \geq \mathbb{I}(y_{j+k}^i = 1) q_{j+k}^i - C(1 - z_i^k) \quad \forall i, j, k$$
$$\sum_{k=1}^{m} z_i^k = 1 \;\forall i,$$

where $C > 1 \in \mathbb{R}$. The intuition behind the constraints is that only the constraint relative to the correct shift k should be satisfied in each sample i; in this form, if $z_i^k = 0$, i.e., the sample i has not been shifted by k, then the constraint becomes automatically satisfied and is vacuous. An equivalent form for the upper bound optimization can be written by adapting the constraints from problem (2).

Samples Corruption Case. In this final setting, we allow the mixed integer programming to ignore a pre-set amount of samples T so to make it robust to sample-level corruption. Such types of corruption can go from single token substitution to adversarial attacks, such as purposely injecting y_i samples completely independent of the watermark key ξ^*. To achieve this, we introduce a further n-dimensional boolean variable $v \in \{0,1\}^n$, where $v_i = 1$ indicates the mixed integer programming has chosen to ignore the sample i. The resulting mixed integer linear program for determining the lower bounds is equal to:

$$\underline{\xi}^* = \underset{\underline{\xi}_j^* \in \mathbb{R}^m \; z^0,\dots,z^m, v \in \{0,1\}^n}{\text{minimize}} \quad \sum_{j=1}^{m} \underline{\xi}_j^* \tag{4}$$

$$\text{s.t.} \qquad \underline{\xi}_j^* \geq \mathbb{I}(y_{j+k}^i = 1) q_{j+k}^i - C(1 - z_i^k) \quad \forall i, j, k$$
$$\sum_{k=1}^{m} z_i^k \geq (1 - v_i) \;\forall i \qquad \sum_{i=1}^{n} v_i \leq T,$$

where a value of $v_i = 1$ allows for all constraints for sample i to be vacuous. As in the other cases, an equivalent form for the upper bound optimization can be written by adapting the constraints from problem (2).

2.3 Detection

For detection, the detector uses a hypothesis test defined as:

$$H_0 : \widetilde{Y} \text{ is not watermarked} \tag{5}$$

$$H_1 : \widetilde{Y} \text{ is watermarked} \tag{6}$$

Specifically, with T samples, the detector computes a p-value with respect to a test statistic $\phi : \mathcal{V}^* \times \Xi^* \to \mathbb{R}$ for H_0, i.e., $\widetilde{Y}$ is independent of ξ. The output of detection progress is a non-asymptotic p-value: if ϕ returns a small p-value then the text is likely to be watermarked; otherwise if the p-value is large then the text is likely not watermarked. Hence, our goal is to design a test statistic ϕ such that $\hat{p}$ will be small when $\widetilde{Y}$ is watermarked. To this end, we rely on the notion of alignment cost $d : (\mathcal{V}^* \times \Xi^*) \to \mathbb{R}$ to measure the quality of a match between a subsequence of the input text and a subsequence of the watermark key. Then, the test statistics ϕ can be set as the minimum alignment cost between any length k subsequences of text and the watermark key.

3 Experimental Analysis

The main question raised in the previous section is whether our method of secret key estimation is capable of reliably 'stealing' the watermark used by the distortion-free watermarking algorithm (Kuditipudi et al. 2024) in a real-world LLM. In this section, we empirically evaluate our 'spoofing' attack across three different scenarios mentioned above using both synthetic data and tokens generated from a watermarked LLM (OPT-125M model Zhang et al. 2022).

3.1 Evaluate Secret Watermark Key Estimation using Synthetic Data

We evaluate the estimated watermark keys, denoted as $\hat{\xi}$, using synthetic data by varying both the text length m and the number of possible shifts in each sample k. The synthetic data are generated by randomly sampling both the watermark keys and the probability mass in the set $\mathcal{G}$ from a uniform distribution $\xi_i^*, q_j^i \sim \mathcal{U}[0,1]$ for all i, j. Figure 1 illustrates how the average L^1 error in watermark key estimation (i.e., $1/m \sum_j |\hat{\xi}_j - \xi_j^*|$), decreases as the number of samples n increases. As expected, while the estimation problem becomes more challenging with increases in text length m, the number of possible shifts k , or the presence of corrupted samples, the error tends to reduce with more samples. We also note that in the sample corruption case, if the number of corrupted samples is larger than the upper bound T for the boolean variable v in the optimization problem (4), the optimization problem is unable to ignore all corrupted samples, and a larger sample size does not necessarily imply a better estimation error. We report the mean and standard deviation of the estimation error over 10 runs (where in each run we change the random seed for the generation of the watermark keys as well as the probability mass in the set $\mathcal{G}$).

3.2 Breaking the Watermarks in OPT-125M LLM

For our watermark stealing experiments, we utilize the OPT-125M model (Zhang et al. 2022). To replicate a variation of language modeling scenarios, we generate our binary text using 100 random prompts generated from ChatGPT (Radford et al. 2019). We then follow the methodology outlined by (Kuditipudi et al. 2024) for watermark detection, where the block size is set equal to the length m of each sample text. This process involves comparing blocks of the binary sequence to the secret key using a test statistic which is then used to calculate a p-value for determining whether the sequence is watermarked.

As seen in Figure 2 and Figure 3, we vary the text length m while fixing the watermark key length to 256 and report median p-values of our spoofed watermarked text for 100

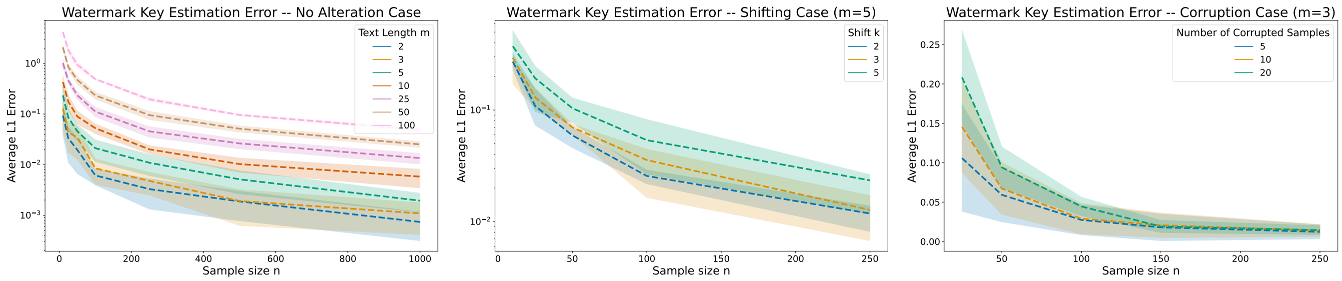

Figure 1: Average estimation error for the watermark keys versus sample sizes, in the no-alteration (left), the shifting case (center), and the sample corruption case (right), using synthetic data. The estimation error decreases as more samples n are available, although the text length m, the potential shift k and the number of corrupted samples make the estimation problem more challenging. We report the mean and standard deviation over 10 runs (see text for more details).

samples. We also evaluate the robustness of our watermark key estimation to the two other attacks, shifting and corruption, mentioned above. These attacks enable us to adjust the degree of alteration of our watermarked text. For the experiments done in this paper, we set our corruption rate = 5%, the shifting amount, k = 2, and vary the text length m.

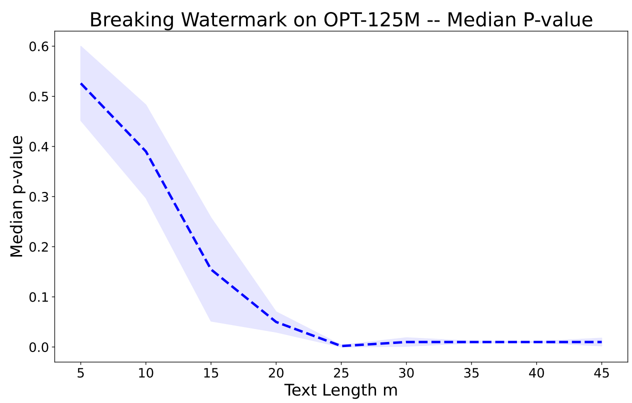

Figure 2: Median p-value of watermarked text generated using the true secret key for varying text length m. Figure shows the performance of the generated binary text. Across the text length for of OPT-125M model, the median p-values decrease rapidly with increasing text length m plateauing after 25 text length.

As mentioned in section 2.1, we initially generate binary text using the key sequence provided by the LM provider. Since this binary text is generated with the true secret key, it serves as the baseline against which our secret key estimation attacks are compared. With a watermark detection threshold of $p < 0.05$, it requires approximately 20 binary texts m before our generated text can be detected as fully watermarked. For binary texts generated exceeding roughly 25, watermark detection tends to yield p-values close to zero. We compare our binary texts generated from the estimated secret key (fig 3) against the texts generated from using the true secret key (fig 1).

Our watermark key estimations are competitive with that of the true key as the generated texts follow the same pattern of requiring about 20 binary texts generated before the watermark can be detected. The 'spoofing' also shows good results with regard to shifting the elements of the underlying key, as this has little to no effect on the p-value with results similar to the key estimation with no alterations. This is not the case when corrupting a percent of the binary texts as this degrades the quality of the watermarked text, requiring more than 45 generated binary texts before the watermark can be detected.

Overall, our results demonstrate that reliable spoofing of binary texts on the OPT-125M language model (Zhang et al. 2022) is possible using our mixed linear programming attack for watermark key estimation, even with various types of attacks and only a few samples of the watermarked text. Similar to the binary text generated from the true secret key, longer texts have to be generated to achieve stronger watermark detection.

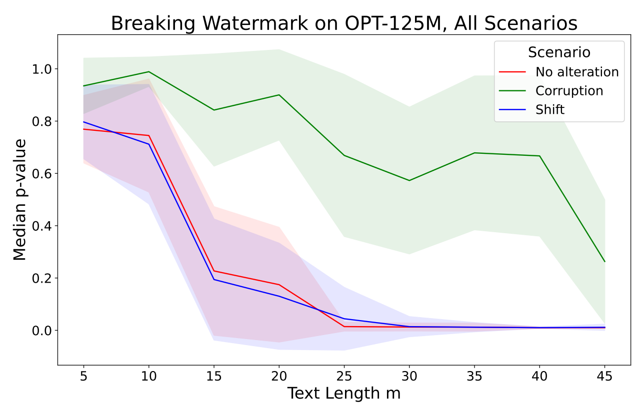

Figure 3: Median p-value of watermarked text generated using our mixed linear programming framework to accurately estimate the secret watermark key. Figure shows the performance using our watermark key estimation 'spoofing' on the three cases of attacks -*no alteration*, *corruption*, and *shifting*. Across the text length m for OPT-125M model, the median p-values of decrease with increasing text length m.

4 Conclusion and Future Work

This work highlights the effectiveness of our mixed linear programming framework in accurately estimating the secret watermark key used by the distortion-free watermarking algorithm. We demonstrate that this framework is robust to input shifts and corruption introduced by the LLM user, as evidenced by the results that closely match those of the distortion-free watermarking method. Moving forward, we plan to extend our approach from the binary generative text to cover the full vocabulary of an LLM. Additionally, we aim to explore the impact of varying the corruption rate and the k parameters, examining how these factors affect watermark detection as the length of the generated binary text increases. We also intend to test our framework with different LLMs. These investigations could offer valuable insights and drive further advancements in the field.

Disclaimer. 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.

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.

Bibliography

Aaronson, S. 2023. 'Reform' AI Alignment with Scott Aaronson. AXRP - the AI X-risk Research Podcast.

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

Gloaguen, T.; Jovanović, N.; Staab, R.; and Vechev, M. 2024b. Discovering Clues of Spoofed LM Watermarks. *arXiv preprint arXiv:2410.02693*.

Jovanović, N.; Staab, R.; and Vechev, M. 2024. Watermark Stealing in Large Language Models. arXiv:2402.19361.

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

Kuditipudi, R.; Thickstun, J.; Hashimoto, T.; and Liang, P. 2024. Robust Distortion-free Watermarks for Language Models. *Transactions on Machine Learning Research*.

Ning, K.; Chen, J.; Zhong, Q.; Zhang, T.; Wang, Y.; Li, W.; Zhang, Y.; Zhang, W.; and Zheng, Z. 2024. MCGMark: An Encodable and Robust Online Watermark for LLM-Generated Malicious Code. *arXiv preprint arXiv:2408.01354*.

Pang, Q.; Hu, S.; Zheng, W.; and Smith, V. 2024. Attacking LLM Watermarks by Exploiting Their Strengths. *arXiv preprint arXiv:2402.16187*.

Piet, J.; Sitawarin, C.; Fang, V.; Mu, N.; and Wagner, D. 2024. Mark My Words: Analyzing and Evaluating Language Model Watermarks. arXiv:2312.00273.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners.

Wu, Q.; and Chandrasekaran, V. 2024. Bypassing LLM Watermarks with Color-Aware Substitutions. arXiv:2403.14719.

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

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

Zhang, Z.; Zhang, X.; Zhang, Y.; Zhang, L. Y.; Chen, C.; Hu, S.; Gill, A.; and Pan, S. 2024b. Large Language Model Watermark Stealing With Mixed Integer Programming. arXiv:2405.19677.