Proceedings of Machine Learning Research vol 196:1–15, 2024 37th Annual Conference on Learning Theory

Undetectable Watermarks for Language Models

Miranda Christ MCHRIST@CS.COLUMBIA.EDU
Columbia University
Sam Gunn GUNN@BERKELEY.EDU
UC Berkeley
Or Zamir ORZAMIR@TAUEX.TAU.AC.IL
Tel Aviv University

Editors: Shipra Agrawal and Aaron Roth

Abstract

Recent advances in the capabilities of large language models such as GPT-4 have spurred increasing concern about our ability to detect AI-generated text. Prior works have suggested methods of embedding watermarks in model outputs, by *noticeably* altering the output distribution. We ask: Is it possible to introduce a watermark without incurring *any detectable* change to the output distribution?

To this end, we introduce a cryptographically-inspired notion of undetectable watermarks for language models. That is, watermarks can be detected only with the knowledge of a secret key; without the secret key, it is computationally intractable to distinguish watermarked outputs from those of the original model. In particular, it is impossible for a user to observe any degradation in the quality of the text. Crucially, watermarks remain undetectable even when the user is allowed to adaptively query the model with arbitrarily chosen prompts. We construct undetectable watermarks based on the existence of one-way functions, a standard assumption in cryptography.

Keywords: Language Models, Machine Learning, Cryptography, Security, Watermarking

1. Introduction

With the rise in the use of artificial models that churn out human-like text, there's also an increase in the potential for misuse. Imagine a student employing a language model to effortlessly write her "Machine Learning 101" homework or conjuring up tear-jerking emails to beg professors for easier exams. That's when the need arises to distinguish between texts penned by a language model and those crafted by human hands. The go-to method of employing a heuristic test to determine if a text was AI-generated, however, grows increasingly fragile as large language models (LLMs) advance. Even the cutting-edge detectors, like GPTZero (Tian (2023)), can be outsmarted with cleverly crafted prompts.

Ultimately, as LLM outputs move closer to becoming identical to human-generated text, this approach becomes hopeless. It is already very hard to tell, for instance, that the previous paragraph was written by such a model. To overcome this problem, it is reasonable to consider intentionally modifying the model to embed *watermarks* into the text. Recent work of Kirchenbauer et al. (2023) introduced such watermarks in the context of LLMs. However, existing watermarking schemes come with a cost: To plant a useful watermark, the distribution of texts the model generates has to be noticeably changed. In fact, for existing schemes it is possible for the *user* to distinguish between outputs of the original model and of the watermarked one, and it is hence possible that the quality of text degrades.

We show how to plant watermarks with the following properties, stated informally, in any LLM.

1. (Undetectability) It is computationally infeasible to distinguish between the original and the watermarked models, even when the user is allowed to make many adaptive queries. In particular, the quality of generated text remains identical.

2. (Completeness) There is a secret key that enables efficient detection of responses from the watermarked model, as long as "enough randomness" was used to generate the response. The detection works even when presented with only a contiguous substring from the response, and it doesn't require any other information.

3. (Soundness) Any text generated independently from the secret key has a negligible chance of being detected as watermarked.

We note that the existence of a secret key is necessary, as otherwise these properties would directly contradict each other. However, in practice the secret key can be published if one wishes; Property (1) still ensures that the quality of the text is imperceptibly changed for all uses not involving the secret key.

An important aspect of our construction is that Properties (1) and (3) will *always* hold, for any LLM with any choice of parameters, and without making any assumptions on the text. Our scheme is the first to have these properties, and we argue that they are completely crucial. First, the creator of a state-of-the-art LLM is unlikely to intentionally degrade the quality of their model, making Property (1) necessary for any practical watermarking scheme. As the quality and versatility of LLMs have reached such high levels, any noticeable change due to the watermark is liable to have adverse side effects in some situations. Second, falsely accusing humans of using LLMs to generate their texts should be completely unacceptable. When heuristics are used for detection, this will always be a possibility — indeed, instances of such false accusations against students have already made news headlines (Fowler (2023); Jimenez (2023)), and concerningly, false accusations may be more common for non-native English writers (Liang et al. (2023)). Property (3) in our construction rigorously guarantees that natural text will not be detected as watermarked.

Of course, a watermark is only useful if it can be detected with the secret key. If the model has a deterministic response to some prompt, then we should not be able to embed the watermark in that response (as any change to the output would necessarily be detectable). For Property (2), we therefore need to assume that enough "randomness" was used in the generation of the specific text we are considering. We introduce a formal notion that we call *empirical entropy*, and show that this condition is necessary. Our detection algorithm works when it is given text containing any consecutive sub-string with enough empirical entropy from an output of the model. Intuitively, we should think of empirical entropy as linear in the length of text, we substantiate this intuition both theoretically and empirically in Section 3.3.1.

Primary contributions of this work include the formal definition and construction of *undetectable watermarks*, and the notion of *empirical entropy* that quantifies the randomness used in the generation of a specific output. These definitions and the rigorous statement of our results appear in Section 3.

1.1. Related Work

Approaches for detecting AI-generated text largely fall into two categories. *Watermarking schemes* alter the output of a language model in a way that a corresponding detection algorithm can observe.

Post-hoc detectors leave the output of the model unchanged and instead identify AI-generated text using existing differences between natural language and the model's output.

Post-hoc detectors. The simplest post-hoc detectors use natural heuristics to distinguish between human- and AI-generated text. These heuristics include relative entropy scoring (Lavergne et al. (2008)), perplexity (Beresneva (2016)), and other statistical methods (Gehrmann et al. (2019)); see Beresneva (2016) for a survey of such methods. Other post-hoc detectors (e.g., Zellers et al. (2019); Mitchell et al. (2023); Tian (2023); Kirchner et al. (2023)) are themselves models, specifically trained for this binary classification task. Unfortunately, these heuristic and model-based methods lack formal guarantees, and it's possible to train a model to transform AI-generated text in a way that evades them. See Jawahar et al. (2020) for more comprehensive background on post-hoc detection of AI-generated text and attacks.

Language watermarking schemes. Several schemes (e.g., Abdelnabi and Fritz (2021); Qiang et al. (2023); Yoo et al. (2023); Munyer and Zhong (2023)) involve using a machine learning model in the watermarking algorithm itself. Abdelnabi and Fritz (2021) and Munyer and Zhong (2023) work by taking a passage of text and using a model to produce a semantically similar altered passage. By nature of using machine learning, these constructions have no formal guarantees and rely on heuristic arguments for undetectability, completeness, soundness.

In a recent work, Kirchenbauer et al. (2023) presented the first watermarking scheme for LLMs with any formal guarantees. They showed that a watermark can be planted in outputs with large enough entropy (with a definition different than ours, yet morally similar). Their watermark, and a similar ongoing project of Aaronson (2022), use a PRF applied to the previous $k-1$ tokens to bias the k^{th} token. The watermark can be detected by computing these PRF outputs and a score that reflects the correlation between tokens and the biases specified by the PRF outputs.

However, these schemes crucially *change* the distribution of generated texts and use this change to detect the watermark. Kirchenbauer et al. (2023) bounds the difference between the original distribution and the distribution of their watermarked model, using a quantity called *perplexity*. Aaronson (2022) guarantees that the two distributions will be indistinguishable, but only as long as no two output texts are seen that share a common substring of $k-1$ tokens. These guarantees of Kirchenbauer et al. (2023) and Aaronson (2022) are relatively weak, and in particular these watermarks may degrade the model's performance on downstream tasks.

A recent manuscript of Kuditipudi et al. (2023) (uploaded after this work and citing it) focuses on robustness of watermarks to edits but achieves a weaker quality notion that they define, called *distortion-freeness*. Roughly, it says that any *single* watermarked output is indistinguishable from the original model. However, their watermark introduces correlations between responses which are noticeable given multiple responses.

In contrast, in our work the original and watermarked output distributions are completely indistinguishable, without any assumption on the texts or the model. In particular we allow the distinguisher to make adaptive queries with arbitrary prompts, so it may force the model to return outputs that share long parts with each other.

2. Organization of the Paper

In Section 3, we formally define our model, introduce the notions of empirical entropy and undetectable watermarks, and outline our results. We consider the formalization of watermarking in

language models and the rigorous definition of relevant security guarantees as the main contribution of this paper in comparison to prior work, hence we dedicate much of the short version to those. In Section 4, we give a high-level overview of our construction of undetectable watermarks. We also include the pseudo-codes for the construction, but the proofs of correctness are deferred to the full version. In Section 5 we review a short part of the discussion in the full paper on possible methods of removing watermarks from texts. In particular, we prove that it is impossible to create undetectable watermarks that are completely unremovable, under certain assumptions.

We defer the majority of proofs and further discussion to the full version of this paper, available at Christ et al. (2023a,b). Besides what is discussed above, the full version also contains discussion of the necessity of the assumptions we make.

3. Modeling the Problem

3.1. Preliminaries

Notation. Let λ denote the security parameter. A function f of λ is *negligible* if $f(\lambda) \in O(\frac{1}{\mathsf{poly}(\lambda)})$ for every polynomial $\mathsf{poly}(\cdot)$. We write $f(\lambda) \leq \mathsf{negl}(\lambda)$ to mean that f is negligible. The security parameter λ should be thought of as "the length of the password/key;" we intuitively think interchangeably of "feasible" and of "polynomial in λ." Cryptographic schemes are considered secure if breaking them requires time/work that is not polynomial (and frequently exponential) in λ. Due to this interpretation of the security parameter, it is standard to assume that everything we encounter (e.g., outputs of the LLM) is of size polynomial in λ.

For a vector or sequence of tokens $s = (s_1, \ldots, s_{|s|})$ and positive integers $b \geq a$, let $s[a : b]$ denote $(s_a, \ldots, s_b)$. We use $\log(x)$ to denote the logarithm base 2 of x, and $\ln(x)$ to denote the natural logarithm of x. For integer $n > 0$, we define $[n] := \{1, \ldots, n\}$. For integers $0 \leq k \leq n$, we define $[k, n] := \{k, \ldots, n\}$.

Pseudorandom function (PRF). Let $\mathcal{F} = \{F_{\mathsf{sk}} : \{0,1\}^{\ell_1(\lambda)} \to \{0,1\}^{\ell_2(\lambda)} \mid \mathsf{sk} \in \{0,1\}^{\lambda}\}$ be a family of functions. $\mathcal{F}$ is a PRF if F_{sk} is efficiently computable and for all probabilistic polynomial-time distinguishers D,

$$\left| \Pr_{\mathsf{sk} \leftarrow \{0,1\}^{\lambda}} \left[D^{F_{\mathsf{sk}}(\cdot)}(1^{\lambda}) = 1 \right] - \Pr_{f} \left[D^{f(\cdot)}(1^{\lambda}) = 1 \right] \right| \leq \mathsf{negl}(\lambda)$$

where f denotes a random function from $\{0,1\}^{\ell_1(\lambda)}$ to $\{0,1\}^{\ell_2(\lambda)}$. PRFs are a standard cryptographic primitive equivalent to one-way functions and can be constructed from standard assumptions (Goldreich et al. (1986); Håstad et al. (1999)).

3.2. Language Models

We loosely follow Kirchenbauer et al. (2023) in our definition of a *language model*. We will often refer to language models simply as *models*.

Definition 1 *A language model* Model *over token set $\mathcal{T}$ is a deterministic algorithm that takes as input a prompt* PROMPT *and tokens previously output by the model $x = (x_1, \ldots, x_{i-1})$, and outputs a probability distribution $p_i = \mathsf{Model}(\text{PROMPT}, x)$ over $\mathcal{T}$.*

A language model Model is used to generate text as a response to a prompt by iteratively sampling from the returned distribution until a special terminating token `done` $\in \mathcal{T}$ is drawn.

Definition 2 *A language model's* response *to* PROMPT *is a random variable* $\overline{\mathsf{Model}}(\text{PROMPT}) \in \mathcal{T}^\star$ *that is defined algorithmically as follows. We begin with an empty list of tokens* $x = ()$*. As long as the last token in* x *is not* `done`*, we draw a token* x_i *from the distribution* $\mathsf{Model}(\text{PROMPT}, x)$ *and append it to* x*. Finally, we set* $\overline{\mathsf{Model}}(\text{PROMPT}) = x$*.*

Throughout the text we will make use of a security parameter λ. We will assume that our model never outputs text of length super-polynomial in λ. (For OpenAI's language models, there is actually a fixed limit to the length of generated text.)

3.3. Entropy and Empirical Entropy

Let $\log(x)$ denote the logarithm base 2 of x. For a probability distribution D over elements of a finite set X, we define the Shannon *entropy* of D as

$$H(D) = \mathop{\mathbb{E}}_{x \sim D}[-\log D(x)],$$

where $D(x)$ is the probability of x in the distribution D. The *empirical entropy* (also known as Shannon information or surprisal) of x in D is simply $-\log D(x)$. The expected empirical entropy of $x \sim D$ is exactly $H(D)$. Intuitively, the empirical entropy of x (with respect to D) is the number of random bits that were required to draw x out of the distribution D. The entropy $H(D)$ is thus the expected number of random bits needed to draw an element out of the distribution D.

We thus define the empirical entropy of a model's response as follows.

Definition 3 *For a language model* Model*, a prompt* PROMPT*, and a possible response* $x \in \mathcal{T}^\star$*, we define the* empirical entropy *of* $\overline{\mathsf{Model}}$ *responding with* x *to* PROMPT *as*

$$H_e(\mathsf{Model}, \text{PROMPT}, x) := -\log \Pr\left[\overline{\mathsf{Model}}\left(\text{PROMPT}\right) = x\right].$$

We next generalize the definition of empirical entropy from whole outputs to *substrings* out of a model's output. This will quantify how much entropy was involved in the generation of a particular contiguous substring of the output.

Definition 4 *For a language model* Model*, a prompt* PROMPT*, a possible response* $x \in \mathcal{T}^\star$*, and indices* $i, j \in [|x|]$ *with* $i \leq j$ *we define the* empirical entropy on substring $[i, j]$ *of* $\overline{\mathsf{Model}}$ *responding with* x *to* PROMPT *as*

$$H_e^{[i,j]}(\mathsf{Model}, \text{PROMPT}, x) := -\log \Pr\Big[\overline{\mathsf{Model}}\left(\text{PROMPT}\right)[i:j] = x[i:j] \mid \overline{\mathsf{Model}}\left(\text{PROMPT}\right)[1:(i-1)] = x[1:(i-1)]\Big].$$

We sometimes write $H_e^i := H_e^{[i,i]}$ to denote the empirical entropy of a single token i. We remark that in expectation, Definition 3 simply captures the entropy in the response generation. That is, we have

$$\mathop{\mathbb{E}}_{x}\left[H_e(\mathsf{Model}, \text{PROMPT}, x)\right] = H\left(\overline{\mathsf{Model}}\left(\text{PROMPT}\right)\right),$$

where $x \sim \overline{\mathsf{Model}}\left(\text{PROMPT}\right)$.

3.3.1. EMPIRICAL ENTROPY IN NATURAL LANGUAGE

Quantitative linguistics research (Genzel and Charniak (2002); Shi and Lei (2022)) concludes that in natural language, the entropy of each unit of text (e.g., a paragraph, a sentence or a word) is usually constant throughout the entire text. As a corollary, the empirical entropy of a text generated from a natural language should be linear in the length of the text. In the full version of this paper, we also run empirical experiments in which we compute the empirical entropy in the responses of popular LLMs, and indeed verify the above claim. For the purpose of this work, this means that any time we say "large enough empirical entropy" it is informally equivalent to saying "long enough".

3.4. Watermarks

We formally define a watermarking scheme as follows.

Definition 5 (Watermarking Scheme) *A* watermarking scheme *for a model* Model *over* $\mathcal{T}$ *is a tuple of algorithms* $\mathcal{W} = (\mathsf{Setup}, \mathsf{Wat}, \mathsf{Detect})$ *where:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{sk}$ *outputs a secret key, with respect to a security parameter* λ*.*
- $\mathsf{Wat}_{\mathsf{sk}}(\text{PROMPT})$ *is a randomized algorithm that takes as input a prompt* PROMPT *and generates a response in* $\mathcal{T}^\star$*.*
- $\mathsf{Detect}_{\mathsf{sk}}(x) \to \{\mathsf{true}, \mathsf{false}\}$ *is an algorithm that takes as input a sequence* $x \in \mathcal{T}^\star$ *outputs* true *or* false*.*

Ideally, $\mathsf{Detect}_{\mathsf{sk}}(x)$ should output true if x is generated by $\mathsf{Wat}_{\mathsf{sk}}(\text{PROMPT})$, and should output false if x is generated independently of sk. The former property is called *completeness* and the latter *soundness*.

Definition 6 (Soundness) *A watermarking scheme* $\mathcal{W}$ *is* sound *if for every security parameter* λ *and token sequence* $x \in \mathcal{T}^\star$ *of length* $|x| \leq \mathsf{poly}(\lambda)$*,*

$$\Pr_{\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)} [\mathsf{Detect}_{\mathsf{sk}}(x) = \mathsf{true}] \leq \mathsf{negl}(\lambda).$$

A scheme is sound if *any* text that is generated independently from sk has negligible probability of being detected as watermarked by $\mathsf{Detect}_{\mathsf{sk}}$. Essentially, this means we will *never* see a false-positive detection.

Defining completeness requires care: It is not reasonable to require $\mathsf{Detect}_{\mathsf{sk}}$ to detect any sequence x generated by $\mathsf{Wat}_{\mathsf{sk}}(\text{PROMPT})$ for some PROMPT, as it is possible that x is very short, or that $\overline{\mathsf{Model}}(\text{PROMPT})$ is deterministic or has very low entropy. Instead, we require $\mathsf{Detect}_{\mathsf{sk}}$ to detect watermarks only in responses for which the entropy in the generation process is high enough.

Definition 7 (Completeness) *A watermarking scheme* $\mathcal{W}$ *is* $b(L)$-complete *if for every security parameter* λ *and prompt* PROMPT *of length* $|\text{PROMPT}| \leq \mathsf{poly}(\lambda)$*,*

$$\Pr_{\substack{\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda) \\ x \leftarrow \mathsf{Wat}_{\mathsf{sk}}(\text{PROMPT})}} [\mathsf{Detect}_{\mathsf{sk}}(x) = \mathsf{false} \textit{ and } H_e(\mathsf{Model}, \text{PROMPT}, x) \geq b(|x|)] \leq \mathsf{negl}(\lambda).$$

Definition 7 guarantees that any output generated by $\mathsf{Wat}_{\mathsf{sk}}$ with empirical entropy at least $b(L)$, where L is the length of the output, will be detected as watermarked with high probability. Essentially, this means we will *never* see a false-negative detection on any output of high enough empirical entropy. It is provably necessary to consider the empirical entropy of the specific output rather than the standard entropy of the entire model.

We also generalize Definition 7 to capture contiguous substrings of outputs. That is, we should be able to detect a watermarked output of $\mathsf{Wat}_{\mathsf{sk}}$ even if $\mathsf{Detect}_{\mathsf{sk}}$ is only given a long enough contiguous substring from it.

Definition 8 (Substring Completeness) *A watermarking scheme $\mathcal{W}$ is $b(L)$-*substring-complete *if for every prompt* PROMPT *and security parameter λ,*

$$\Pr_{\substack{\mathsf{sk}\leftarrow\mathsf{Setup}(1^\lambda)\\ x\leftarrow\mathsf{Wat}_{\mathsf{sk}}(\text{PROMPT})}}\Big[\,\exists\, i, L\in[|x|] \textit{ such that } \mathsf{Detect}_{\mathsf{sk}}(x[i:i+L])=\mathsf{false} \textit{ and } H_e^{[i:i+L]}(\mathsf{Model},\text{PROMPT},x)\geq b(L)\,\Big]\leq \mathsf{negl}(\lambda).$$

This means that every contiguous part of an output of the watermarking procedure, that has high enough empirical entropy, is detected as watermarked with high probability. We stress that the empirical entropies in Definitions 7,8 are defined with respect to the original model Model, without reference to the watermarking procedure $\mathsf{Wat}_{\mathsf{sk}}$. We also note that the empirical entropy $H_e(\mathsf{Model}, \text{PROMPT}, x)$ is only used as part of the definition, and is not necessarily known to $\mathsf{Detect}_{\mathsf{sk}}$. It is in general not possible to compute $H_e(\mathsf{Model}, \text{PROMPT}, x)$ without knowledge of PROMPT.

3.5. Undetectable Watermarks

Finally, we define the notion of computationally undetectable watermarking schemes. Intuitively, a scheme is undetectable if it is infeasible to distinguish between the distributions of $\overline{\mathsf{Model}}$ and $\mathsf{Wat}_{\mathsf{sk}}$, even when the given distribution can be queried adaptively with arbitrary prompts.

Definition 9 (Undetectability) *A watermarking scheme $\mathcal{W}=(\mathsf{Setup},\mathsf{Wat},\mathsf{Detect})$ is* undetectable *if for every security parameter λ and all polynomial-time distinguishers D,*

$$\left|\Pr[D^{\mathsf{Model},\overline{\mathsf{Model}}}(1^\lambda)\to 1]-\Pr_{\mathsf{sk}\leftarrow\mathsf{Setup}(1^\lambda)}[D^{\mathsf{Model},\mathsf{Wat}_{\mathsf{sk}}}(1^\lambda)\to 1]\right|\leq \mathsf{negl}(\lambda),$$

where the notation $D^{\mathcal{O}_1,\mathcal{O}_2}$ means that D is allowed to adaptively query both $\mathcal{O}_1$ and $\mathcal{O}_2$ with arbitrary prompts.

Remark 10 *In the above definition, we allow the distinguisher access to* Model *itself as well as* $\overline{\mathsf{Model}}$ *or* $\mathsf{Wat}_{\mathsf{sk}}$*. The only thing that is kept secret from the distinguisher is the secret key. A natural alternate definition, where D has access only to* $\overline{\mathsf{Model}}$ *or* $\mathsf{Wat}_{\mathsf{sk}}$ *(and not* Model*), in fact implies this definition since the adversary may have* Model *hard-coded.*

It is important to remark that in any undetectable watermarking scheme, the *quality of outputs* must be identical between $\overline{\mathsf{Model}}$ and $\mathsf{Wat}_{\mathsf{sk}}$, as otherwise it would be possible to distinguish between them. In particular, embedding the watermark does not degrade the quality of the generated text *at all*.

We finally note that a watermarking scheme can be made *public* by publishing the secret key sk. Then, everyone can run the detection algorithm $\mathsf{Detect}_{\mathsf{sk}}$. In particular, the scheme is no longer undetectable as $\mathsf{Detect}_{\mathsf{sk}}$ can be used to distinguish between $\overline{\mathsf{Model}}$ and $\mathsf{Wat}_{\mathsf{sk}}$. Nevertheless, we still maintain the property that there is no degradation in the quality of watermarked outputs, as long as the definition of "quality" does not depend on the secret key sk.

3.6. Statement of our Theorems

We are now ready to formally state the guarantees of the watermarking schemes that we present.

Theorem 11 *For any model* Model *we construct a watermarking scheme $\mathcal{W}$ that is undetectable, sound, and $O(\lambda\sqrt{L})$-complete.*

This means that our watermarking scheme is *always* undetectable and sound, and is also complete as long as there is enough empirical entropy in the model's response.

As proven in the full version, it is *necessary* for the completeness parameter $b(L)$ to be reasonably large, with respect to λ. In fact, we show that it is *inherent* that low empirical entropy outputs are not watermarked in any undetectable watermarking scheme for any model. Intuitively, an output of low empirical entropy appears too frequently for its probability to be modified without detection. However, it is not clear that the completeness parameter needs to grow with the length of text L. Our scheme has this requirement because, roughly, the empirical entropy needs to be large enough that the signal outweighs random fluctuations. On the other hand, in the full version we present a scheme with completeness parameter independent of L — but that scheme is not sound. Closing this gap remains an interesting open question.

To strengthen Theorem 11, we also present a modified scheme which obtains substring completeness, with similar parameters.

Theorem 12 *For any model* Model *we construct a watermarking scheme $\mathcal{W}$ that is undetectable, sound, and $O(\lambda\sqrt{L})$-substring-complete.*

4. Constructing Undetectable Watermarks

4.1. Reduction to a Binary Alphabet

For ease of presentation and analysis, we describe our watermarking scheme as operating on text encoded as a binary string. That is, we assume that the token set is $\mathcal{T} = \{0, 1\}$.

Note that this assumption is without loss of generality: We can easily convert a model M with an arbitrary token set $\mathcal{T}$ into a model M' with a binary token set. First, we encode each token in $\mathcal{T}$ as a distinct string in $\{0, 1\}^{\log|\mathcal{T}|}$.

Let E denote this encoding function, and let p_i be a distribution over $\mathcal{T}$ output by M. We convert p_i into a series of distributions $p'_{i,j}$ for M', where $p'_{i,j}$ is the distribution of the j^{th} bit of $E(t_i)$ for a token $t_i \leftarrow p_i$.

That is, if we have already sampled $b_{i,1}, \ldots, b_{i,j-1}$, then for $b \in \{0, 1\}$ we let

$$p'_{i,j}(b) = \Pr_{t_i \leftarrow p_i}[E(t_i)_j = b \mid E(t_i)_k = b_{i,k} \text{ for } k < j].$$

Note that these distributions can be easily computed as

$$p'_{i,j}(b) = \sum_{t \in \mathcal{T}} p_i(t) \cdot \mathbb{1}[E(t)_k = b_{i,k} \text{ for } k < j \text{ and } E(t)_j = b].$$

Therefore, a watermarking scheme for binary alphabets can be used on models with token alphabets of arbitrary size using the above reduction. For GPT-4, the number of tokens is $|\mathcal{T}| = 100,277$, and thus these strings can be chosen to have length $\log |\mathcal{T}| \approx 17$ (OpenAI (2023)). We note that the expected length of the encoding can be reduced by using a Huffman encoding of the token set instead of an arbitrary encoding.

4.2. Overview of the Construction

If we don't require undetectability, an easy way to watermark is to use a $\{0,1\}$-valued hash function h and sample tokens x_j with preference for those satisfying $h(x_j) = 1$. Given some text, we can determine whether a watermark is present by computing the hash of each token. In watermarked text, more tokens should hash to 1 than to 0; in un-watermarked text, there should be no bias. This is a classic idea in steganography, discussed in Hopper et al. (2009), and is essentially the idea used in Zhao et al. (2023).

Unfortunately, this strategy significantly alters the output distribution, making it easily detectable: it prefers half of the words in the token set. Our objective is to plant a signal *without* noticeably changing the distribution of generated text.

We first discuss a watermarking scheme that can only be used to generate a single output text of a predetermined maximum length L, for an arbitrary prompt. The secret key shared by the watermarked model and the detection algorithm will be a sequence $\boldsymbol{u} = u_1, \ldots, u_L$ of uniformly chosen real numbers in the range $[0,1]$. Even though this state is independent of the prompt the model will receive, we show that this shared state is enough to plant a watermark in any single response. From the perspective of a user who doesn't know the secret key $\boldsymbol{u}$, the distribution of outputs is not changed at all.

When generating a response, the watermarked model will use the secret key to decide on each output token. Consider the generation of the j-th token in the response, after the previous tokens are already decided. Let $p_j(1)$ denote the probability, according to the real model, of this token being 1. The watermarked model outputs $x_j = 1$ if $u_j \leq p_j(1)$ and $x_j = 0$ otherwise. As u_j was drawn uniformly from $[0,1]$, the probability that the watermarked model output $x_j = 1$ is *exactly* $p_j(1)$. Therefore, the distribution of generated text (in a single response) does not change at all. Nevertheless, we next show that the detection algorithm can compare the generated text to the shared sequence $\boldsymbol{u}$, and deduce that the generated output was drawn from the watermarked model.

For each text bit x_j, the detection algorithm can compute a score

$$s(x_j, u_j) = \begin{cases} \ln \frac{1}{u_j} & \text{if } x_j = 1 \\ \ln \frac{1}{1-u_j} & \text{if } x_j = 0 \end{cases}.$$

Given a string $x = (x_1, \ldots, x_L)$, the detection algorithm sums the score of all text bits

$$c(x) = \sum_{j=1}^{L} s(x_j, u_j).$$

Crucially, the detection algorithm does not need to know the distributions with which the model produced $x_1, \ldots, x_L$. Since the detection algorithm does not have access to the prompt, it would not be able to compute those distributions.

We observe that the expected score is higher in watermarked text, as u_j is correlated with the output bit x_j. In non-watermarked text, the value of u_j is independent of the value of x_j. Therefore, $s(x_j, u_j)$ is simply an exponential random variable with mean 1:

$$\mathop{\mathbb{E}}_{u_j}[s(x_j, u_j)] = \int_0^1 \ln(1/x)\, dx = 1,$$

and we have $\mathbb{E}_{\boldsymbol{u}}[c(x) - |x|] = 0$.

For watermarked outputs, on the other hand,

$$\begin{aligned}
\mathop{\mathbb{E}}_{u_j}[s(x_j, u_j)] &= \int_0^{p_j(1)} \ln \frac{1}{u}\, du + \int_{p_j(1)}^1 \ln \frac{1}{1-u}\, du \\
&= \int_0^{p_j(1)} \ln \frac{1}{u}\, du + \int_0^{p_j(0)} \ln \frac{1}{u}\, du \\
&= \big(p_j(1) - p_j(1) \cdot \ln p_j(1)\big) + \big(p_j(0) - p_j(0) \cdot \ln p_j(0)\big) \\
&= 1 + \ln(2) \cdot H(p_j),
\end{aligned}$$

and the total expected score is

$$\mathop{\mathbb{E}}_{\boldsymbol{u}}[c(x) - |x|] = \ln 2 \cdot H(\overline{\mathsf{Model}}(\textsc{prompt})).$$

We've shown that there's a substantial gap between the expected scores of watermarked and natural text, as long as the text generation has high entropy. This should give us hope that this biasing strategy yields a reliable detector, but there are a few obstacles left on the way.

First, the expectation argument turns out to not be very useful because the variance of the score could be large. This requires a much more careful analysis of the distribution. In the full version of the paper, we discuss why this implies that we must consider *empirical entropy* instead of the entropy of the entire model.

Second, the scheme described above is only indistinguishable for a single response, and that response must be shorter than the secret key. A natural idea is to use a pseudorandom function (refer to Section 3.1 for a definition) to determine the values u_j, instead of drawing them all in advance. For example, by setting $u_j = F_{\mathsf{sk}}(j)$ the length of any single response no longer has to be bounded. As F_{sk} is queried on each input j at most once, the values of u_j are pseudorandom and the distribution of a single watermarked output is computationally indistinguishable from the original distribution. But can we deal with multiple responses? One of our main contributions, and the most substantial difference from all prior work, is to answer this question in the affirmative.

As a first attempt, let $r^{(i)}$ be a unique identifier assigned to each response. This might be a global counter or a random string (sometimes referred to as a *nonce*). To sample the j-th token of the i-th response we can use $u_j^{(i)} = F_{\mathsf{sk}}(r^{(i)}, j)$. If all pairs $(r^{(i)}, j)$ are unique, then the values of $u_j^{(i)}$ are pseudorandom. However, the detection algorithm needs to know $r^{(i)}$ to compute the detection score. If $r^{(i)}$ is a counter, then we would need to keep a global state to maintain it. Moreover, to use the detection algorithm we would need to enumerate over all possible counter values. If $r^{(i)}$ is

a long random string, no global state is needed, but the detection algorithm still needs to know $r^{(i)}$. While $r^{(i)}$ must be recoverable by the detection algorithm, it cannot simply be written in the output text, as we might as well just append "WATERMARK!" to it instead (which would obviously change the distribution of outputs). Our solution is to use real randomness to generate the first few tokens of each output, keeping track of how much *entropy* we used in the process. Once this entropy passes some specified threshold, we use the high-entropy prefix as $r^{(i)}$. Since these prefixes have high enough entropy, all choices of $r^{(i)}$ will be unique with all but negligible probability. The detection algorithm will test whether any prefix in the text, if used as $r^{(i)}$, will yield an unusually high score for the remainder of the text.

In the above sketch the detector needs the entire output from the model to detect the watermark. We use a modification of this scheme which is able to detect the watermark, even when it is given only an contiguous substring of the output with sufficiently high entropy. Essentially, this modification works the same except it "resets" the choice of $r^{(i)}$ whenever enough *new* pseudo-entropy is observed.

4.3. Pseudo-Code

Let $\mathsf{poly}_1(\cdot), \mathsf{poly}_2(\cdot)$ be polynomials. Let $F_{\mathsf{sk}} : \{0,1\}^{\mathsf{poly}_1(\lambda)} \to \{0,1\}^{\mathsf{poly}_2(\lambda)}$ be a PRF, where $\mathsf{sk} \in \{0,1\}^\lambda$. We wish to interpret the output of F_{sk} as a real number in $[0,1]$. We do so by letting z be the integer representation of the output and taking $\frac{z}{2^{\mathsf{poly}_2(\lambda)}}$. We consider poly_2 to be a large polynomial and ignore floating point errors. In the below algorithms, we allow F_{sk} to take strings of varying length as input; we assume that $\mathsf{poly}_1(\cdot)$ is chosen such that these strings are never too long, and if they are too short we pad them. In the algorithms we assume that the token alphabet is binary as discussed in Section 4.1. We let `done` denote the binary encoding of the "done" token, and we write $\texttt{done} \in (x_1, \ldots, x_k)$ if and only if the decoding of $(x_1, \ldots, x_k)$ in the original token alphabet includes `done`.

We let $\mathcal{W} = (\mathsf{Setup}, \mathsf{Wat}, \mathsf{Detect})$ denote the watermarking scheme where Wat is Algorithm 1, Detect is Algorithm 2, and $\mathsf{Setup}(1^\lambda)$ samples $\mathsf{sk} \leftarrow \{0,1\}^\lambda$. The correctness proofs are deferred to the full version.

5. Removability of Watermarks

A natural question is how robust an undetectable watermarking scheme can be to active attempts to remove it. Our watermark scheme, for example, is detectable as long as a long enough consecutive substring out of the LLM output remains intact. While we would ideally like to have an undetectable watermarking scheme that is robust to any efficient adversary attempting to remove a watermark, there are both practical and theoretical barriers to achieving this property. Intuitively, a well-resourced enough user that is able to completely paraphrase an LLM output should be able to remove any watermarks. This intuition can be substantiated both empirically (we discuss this further in the full version of the paper) and theoretically (e.g., Zhang et al. (2023) show that under some strong assumption on the ability to rephrase text, all watermarks are removable). We also provide an insight on the matter by showing that certain API choices (in particular, allowing a user to specify a prefix for the response) allow the user to provably remove undetectable watermarks.

We say that a model is *prefix-specifiable* if the user can specify a prefix of the model's response. More formally, we require that for any PROMPT and text $x_1, \ldots, x_k$, the user can efficiently compute a new prompt PROMPT$'$ such that $\overline{\mathsf{Model}}(\text{PROMPT}')$ is distributed identically to $\overline{\mathsf{Model}}(\text{PROMPT})$

Algorithm 1: Substring-complete watermarking algorithm $\mathsf{Wat}_{\mathsf{sk}}$.
Data: A prompt (PROMPT), secret key sk, and parameter λ
Result: Watermarked text $x_1, \ldots, x_L$
$r \leftarrow \perp, H \leftarrow 0, i \leftarrow 1, \ell \leftarrow 1$
while $\texttt{done} \notin (x_1, \ldots, x_{i-1})$ **do**
  $p_i \leftarrow \mathsf{Model}(\text{PROMPT}, x_1, \ldots, x_{i-1})$
  if $r = \perp$ **then**
    `// Still sampling the first block`
    Sample $x_i \leftarrow p_i$
  else
    `// Embed the watermark`
    $x_i \leftarrow \mathbb{1}[F_{\mathsf{sk}}(r, \ell) \leq p_i(1)]$
  $H \leftarrow H - \log p_i(x_i)$
  if $H \geq \frac{2}{\ln 2}\lambda\sqrt{\ell}$ **then**
    `// Reassign` r
    $r \leftarrow (x_{i-\ell}, \ldots, x_i)$
    $H \leftarrow 0, \ell \leftarrow 0$
  end
  $i \leftarrow i + 1, \ell \leftarrow \ell + 1$
end

Algorithm 2: Substring-complete detector $\mathsf{Detect}_{\mathsf{sk}}$.
Data: Text $x_1, \ldots, x_L$ and a secret key sk
Result: true or false
for $i, \ell \in [L]$, $\ell < i$ **do**
  $r^{(i,\ell)} \leftarrow (x_{i-\ell}, \ldots, x_i)$
  $v_j^{(i,\ell)} \leftarrow x_j \cdot F_{\mathsf{sk}}(r^{(i,\ell)}, j - i - 1) + (1 - x_j) \cdot (1 - F_{\mathsf{sk}}(r^{(i,\ell)}, j - i - 1))$ for each $j \in [L]$
  for $k \in [i + 1, L]$ **do**
    if $\sum_{j=i+1}^{k} \ln(1/v_j^{(i,\ell)}) > (k - i) + \lambda\sqrt{k - i}$ **then**
      return true
    end
  end
end
return false

conditioned on the response's prefix matching $(x_1, \ldots, x_k)$. This property is also assumed in the definition of a language model in Kirchenbauer et al. (2023). For example, ChatGPT does not allow the user to specify prefixes of the response, but the OpenAI Playground allows the user to submit text under the "Assistant" role which the model will use as a prefix for its next response.

Theorem 13 *Let $\mathcal{W} = (\mathsf{Setup}, \mathsf{Wat}_{\mathsf{sk}}, \mathsf{Detect}_{\mathsf{sk}})$ be any undetectable watermarking scheme. Assume that the underlying model $\overline{\mathsf{Model}}$ is prefix-specifiable. Then there exists an efficient algorithm $\mathcal{A}$ making queries to $\mathsf{Wat}_{\mathsf{sk}}$ such that, for any* PROMPT *and a random* $\mathsf{sk} \leftarrow \mathsf{Setup}(1^\lambda)$*, the distributions $\mathcal{A}^{\mathsf{Wat}_{\mathsf{sk}}}(\text{PROMPT})$ and $\overline{\mathsf{Model}}(\text{PROMPT})$ are $\mathsf{negl}(\lambda)$-close in* statistical *distance. The number of queries made by $\mathcal{A}$ to $\mathsf{Wat}_{\mathsf{sk}}$ is exactly the length of text output by $\mathcal{A}$.*

We conclude that no undetectable watermarking scheme can be *completely* unremovable. Still, it might require significantly more resources for a user to generate unwatermarked text from the model. It remains an open problem to find the strongest model of editing to which undetectable watermarks can be robust.

References

Scott Aaronson. My AI Safety Lecture for UT Effective Altruism. https://scottaaronson.blog/?p=6823, November 2022. Accessed May 2023. 3

Sahar Abdelnabi and Mario Fritz. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 121–140. IEEE, 2021. 3

Daria Beresneva. Computer-generated text detection using machine learning: A systematic review. In *Natural Language Processing and Information Systems: 21st International Conference on Applications of Natural Language to Information Systems, NLDB 2016, Salford, UK, June 22-24, 2016, Proceedings 21*, pages 421–426. Springer, 2016. 3

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *CoRR*, abs/2306.09194, 2023a. doi: 10.48550/ARXIV.2306.09194. URL https://doi.org/10.48550/arXiv.2306.09194. 4

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. *IACR Cryptol. ePrint Arch.*, page 763, 2023b. URL https://eprint.iacr.org/2023/763. 4

Geoffrey A. Fowler. We tested a new chatgpt-detector for teachers. it flagged an innocent student. *The Washington Post*, April 2023. 2

Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. Gltr: Statistical detection and visualization of generated text. *arXiv preprint arXiv:1906.04043*, 2019. 3

Dmitriy Genzel and Eugene Charniak. Entropy rate constancy in text. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 199–206, 2002. 6

Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986. 4

Johan Håstad, Russell Impagliazzo, Leonid A Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999. 4

Nicholas J. Hopper, Luis von Ahn, and John Langford. Provably secure steganography. *IEEE Trans. Computers*, 58(5):662–676, 2009. doi: 10.1109/TC.2008.199. URL https://doi.org/10.1109/TC.2008.199. 9

Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks VS Lakshmanan. Automatic detection of machine generated text: A critical survey. *arXiv preprint arXiv:2011.01314*, 2020. 3

Kayla Jimenez. Professors are using ChatGPT detector tools to accuse students of cheating. But what if the software is wrong? *USA Today*, April 2023. 2

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html. 1, 3, 4, 12

Jan Hendrik Kirchner, Lama Ahmad, Scott Aaronson, and Jan Leike. New ai classifier for indicating AI-written text. https://openai.com/blog/new-ai-classifier-for-indicating-ai-written-text, January 2023. Accessed May 2023. 3

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023. 3

Thomas Lavergne, Tanguy Urvoy, and François Yvon. Detecting fake content with relative entropy scoring. *PAN*, 8:27–31, 2008. 3

Weixin Liang, Mert Yuksekgonul, Yining Mao, Eric Wu, and James Zou. GPT detectors are biased against non-native english writers. *arXiv preprint arXiv:2304.02819*, 2023. 2

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. DetectGPT: Zero-shot machine-generated text detection using probability curvature. *CoRR*, abs/2301.11305, 2023. doi: 10.48550/arXiv.2301.11305. URL https://doi.org/10.48550/arXiv.2301.11305. 3

Travis Munyer and Xin Zhong. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint arXiv:2305.05773*, 2023. 3

OpenAI. tiktoken repository. https://github.com/openai/tiktoken, 2023. Accessed April 2023. 9

Jipeng Qiang, Shiyu Zhu, Yun Li, Yi Zhu, Yunhao Yuan, and Xindong Wu. Natural language watermarking via paraphraser-based lexical substitution. *Artificial Intelligence*, page 103859, 2023. 3

Yaqian Shi and Lei Lei. Lexical richness and text length: An entropy-based perspective. *Journal of Quantitative Linguistics*, 29(1):62–79, 2022. 6

Edward Tian. gptzero update v1. https://gptzero.substack.com/p/gptzero-update-v1, January 2023. Accessed May 2023. 1, 3

KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. Robust natural language watermarking through invariant features. *arXiv preprint arXiv:2305.01904*, 2023. 3

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. Defending against neural fake news. *Advances in neural information processing systems*, 32, 2019. 3

Hanlin Zhang, Benjamin L 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. 11

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for AI-generated text. *arXiv preprint arXiv:2306.17439*, 2023. 9