Watermarks for Language Models via Probabilistic Automata

Anonymous ACL submission

Abstract

Decoder-based watermarking embeds a secret key into language-model outputs at inference time. Traditional schemes rely on k-gram hashing to bias token sampling, while a recent scheme leverages edit distance by generating a noise sequence and measuring its distance to the produced text for detection. However, this method suffers from limited generation diversity and substantial computational overhead during detection. We study a broader class of watermarking schemes based on *probabilistic automata*, which generalizes prior edit-distance-based constructions. Within this framework, we present two instantiations: (i) a practical scheme that improves both generation diversity and detection efficiency, and (ii) a theoretical construction with formal undetectability guarantees under standard cryptographic assumptions. Experiments on LLaMA-3B and Mistral-7B demonstrate that our approach remains robust under edit-based corruption while substantially reducing detection cost.

1 Introduction

The rapid development of large-scale language models (LMs) has markedly improved AI's ability to generate textual content (Brown et al., 2020). Despite these advancements, apprehensions have arisen over authenticity, ownership, and potential misuse of such technologies (Zellers et al., 2019; Solaiman et al., 2019). Traditional AI detection methods, such as classifier-based detection, often fall short in terms of robustness. In contrast, text watermarking offers a potential solution to these problems. It works by embedding a private key within the text that can be detected by the key holder, thereby identifying and minimizing the abuse of AI-generated content.

A widely adopted watermarking method conditions the decoder on the preceding k generated tokens (Kirchenbauer et al., 2023; Aaronson, 2022). While effective, this approach can degrade LM's output quality by introducing noticeable distortions, such as biases toward certain k-grams. To address these distortions, distortion-free watermarking was introduced to preserve the LM's output distribution (Kuditipudi et al., 2024); however, it does not guarantee LM's generation diversity. More recently, undetectable watermarking has been explored (Christ et al., 2024), which prevents detection by adversaries and naturally maintains generation diversity. Despite these advancements, the relationship between distortion-freeness and undetectability has thus far never been clearly defined. To this end, we establish the connection between distortion-freeness and undetectability, and show that many existing watermarks are detectable. One interpretation of this fact is that the watermarking output distribution admits a simple automaton representation and can therefore be learned under the Probably Approximately Correct (PAC) framework.

Our work is closely related to the approach of Kuditipudi et al. (2024), which uses a cyclic key sequence as noise for unbiased decoding and leverages the *edit distance* (specifically, Levenshtein distance) metric to improve robustness against any edit-based attacks. However, this method suffers from two notable drawbacks: (1) it reduces generative diversity, often leading to deterministic outputs, and (2) it requires partitioning text into blocks with the time complexity scales quadratically with the block size, which creates a major computational bottleneck.

In this line of work, detection is formulated as computing the distance between the generated text and a noise distribution. In their setting, this distribution is defined over a finite set of cyclic sequences with uniform probability, limiting its expressiveness. In contrast, we consider a broader class of noise distributions defined via *probabilistic automata* (PA), which provide sufficient structure for efficient edit-distance computation. Building on this formulation, we introduce a new class of water-

marking schemes based on PA, with the following key contributions:

- The proposed PA-based watermarking framework generalizes the method of Kuditipudi et al. (2024), which can be viewed as a special case corresponding to a probabilistic automaton with a simple cyclic topology (see Figure 1).
- We demonstrate the expressiveness of this framework by constructing a watermarking scheme that achieves formal undetectability under suitable assumptions.
- Practically, we present a scheme that improves both generation diversity and detection efficiency over Kuditipudi et al. (2024): (i) generation diversity increases from $\Theta(\lambda)$ to $\Omega(\lambda d^n)$, where $d \geq 1$, λ is the key length, and n is the sequence length; and (ii) detection time complexity is reduced from $\Theta(\lambda n k^2)$ to $\Theta(\lambda n)$, where k denotes their block size.

2 Preliminaries

We denote the alphabet of an automaton by Σ and the vocabulary of a language model by $\mathcal{V}$ to explicitly distinguish them. A language is defined as a mapping $\Sigma^* \to [0,1]$ (or $\mathcal{V}^* \to [0,1]$, respectively). Given an automaton $\mathcal{M}$, we denote by $\mathcal{L}(\mathcal{M}) \subseteq \Sigma^*$ the language recognized by $\mathcal{M}$, i.e., the set of strings accepted by $\mathcal{M}$. We denote the size of the watermark key by λ. In undetectable watermarking settings, it also serves as the security parameter, which measures the strength of distortion-freeness and undetectability of a watermarking scheme. Further discussion on λ will follow in Section 4. The length of the generated sequence is given by $m = \mathcal{O}(\text{poly}(\lambda))$, where $\text{poly}(\cdot)$ denotes a polynomial function. For a sequence $\boldsymbol{x} = (x_1, \ldots, x_n)$, we write $\boldsymbol{x}_{i:} = (x_i, \ldots, x_n)$ for a suffix, and $\boldsymbol{x}_{i:j} = (x_i, \ldots, x_j)$ for a contiguous subsequence.

Definition 1 (Language Model)**.** An (autoregressive) language model is defined by a function $\mathsf{Model} : \mathcal{V}^* \to \Delta(\mathcal{V})$ that maps a sequence of tokens to a probability distribution over $\mathcal{V}$, where $\Delta(\cdot)$ denotes a probability distribution over a set. Given an initial sequence of tokens (a *prompt*) $\boldsymbol{x} \in \mathcal{V}^*$, the probability of a sequence $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)$ is defined as

$$p(\boldsymbol{y}) = \prod_{i=1}^{m} p(y_i \mid \boldsymbol{x}, \boldsymbol{y}_{1:i-1}), \tag{1}$$

[1]All subscripts are taken modulo λ.

where $\boldsymbol{y}_{l:h} = (y_l, y_{l+1}, \ldots, y_h)$ represents a subsequence of $\boldsymbol{y}$. Each conditional probability is modeled by

$$p(\cdot \mid \boldsymbol{x}, \boldsymbol{y}_{1:i-1}) = \mathsf{Model}(\boldsymbol{x}, \boldsymbol{y}_{1:i-1}). \tag{2}$$

We use the notation $\boldsymbol{y} \xleftarrow{\mathsf{AR}} \mathsf{Model}(\boldsymbol{x})$ to indicate that $\boldsymbol{y}$ is autoregressively generated by Model given $\boldsymbol{x}$, following Equation 1.

Definition 2 (Decoder-based Watermarking Scheme)**.** A decoder-based watermarking scheme is a triplet $\mathcal{W} := (\mathsf{Gen}, \mathsf{Model}^{\mathsf{wat}}, \mathsf{Detect})$ such that:

1. The key generation algorithm Gen is randomized and takes as input 1^λ to generate a secret key $\mathsf{sk} \in \mathcal{K}$: [2]

$$\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda). \tag{3}$$

2. The watermarking algorithm $\mathsf{Model}^{\mathsf{wat}}_{\mathsf{sk}} := (\mathsf{Model}, \Phi, \Gamma)$ is an autoregressive process that consists of an unwatermarked model Model, a noise generator $\Phi : \mathcal{K} \times \Xi^* \times \mathcal{V}^* \to \Delta(\Xi)$, and a decoder $\Gamma : \Xi \times \Delta(\mathcal{V}) \to \mathcal{V}$, where Ξ is the noise space. At the i-th decoding step, $\xi_i \in \Xi$ is sampled by

$$\xi_i \sim \Phi_{\mathsf{sk}}(\boldsymbol{\xi}_{1:i-1}, \boldsymbol{x}, \boldsymbol{y}_{1:i-1}), \tag{4}$$

then the next token y_i is produced by the decoder deterministically given the noise ξ_i:

$$y_i \leftarrow \Gamma(\xi_i, \mathsf{Model}(\boldsymbol{x}, \boldsymbol{y}_{1:i-1})). \tag{5}$$

3. The detection algorithm $\mathsf{Detect}_{\mathsf{sk}}$ takes as input sk and $\boldsymbol{y}$, and outputs true or false.

While the above definition presented seems abstract, it covers a wide range of existing watermarking frameworks. Specific examples are provided in Appendix E.

Ideally, $\mathsf{Detect}_{\mathsf{sk}}(\boldsymbol{y})$ should output true if $\boldsymbol{y}$ is generated by $\mathsf{Model}^{\mathsf{wat}}_{\mathsf{sk}}(\boldsymbol{x})$ for some $\boldsymbol{x}$, and output false if $\boldsymbol{y}$ is independent of sk. The former property is referred to as *completeness* and the latter *soundness*. As a special case, when Φ does not depend on the prefix of tokens $\boldsymbol{x}$ and $\boldsymbol{y}_{1:i-1}$, Equation 4 simplifies to

$$\xi_i \sim \Phi_{\mathsf{sk}}(\boldsymbol{\xi}_{1:i-1}). \tag{6}$$

[2]The input is given in unary notation to ensure polynomial runtime in λ. The structure of the key can be arbitrary, with specifics described in later sections.

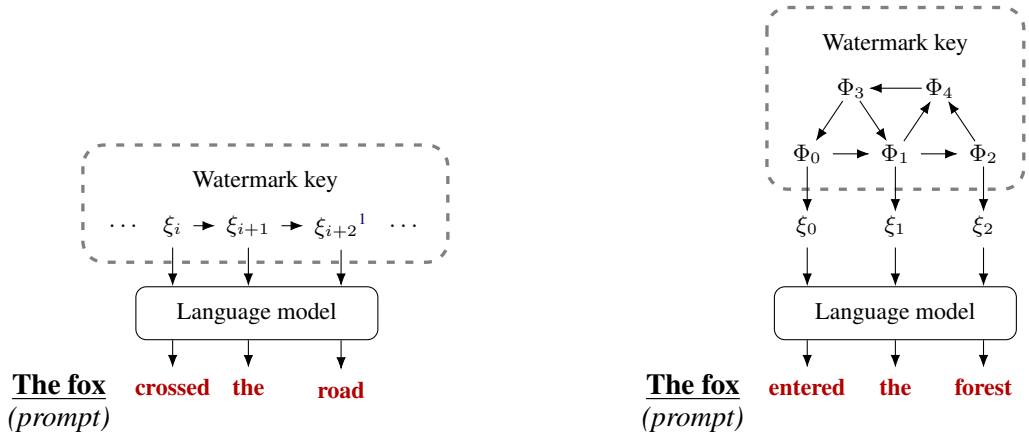

Figure 1: Comparison of generation of two watermarking schemes where the key follows a cyclic structure (Kuditipudi et al., 2024) on the left and a probabilistic automaton on the right. Each Φ specifies a probability distribution over ξ, with precise definitions provided in later sections.

We refer to this case as *model-agnostic* as the distribution of the noise can be decomposed autoregressively with the chain rule and does not depend on the specific model used and therefore can be precomputed before decoding.

All decoder-based watermarking schemes require sufficiently high text entropy; otherwise, the output tends to be deterministic and no watermarks can be embedded. For a detailed discussion of entropy effects, refer to Christ et al. (2024); Kuditipudi et al. (2024); we do not repeat these efforts here. In the sequel a watermarking scheme always denotes a decoder-based watermarking scheme.

The robustness of a watermarking scheme quantifies its ability to withstand edit-based corruptions to the watermarked data without losing the embedded watermark.

Definition 3 (Edit Distance)**.** The edit distance $d(\boldsymbol{s}_1, \boldsymbol{s}_2)$ between two sequences $\boldsymbol{s}_1, \boldsymbol{s}_2 \in \mathcal{V}^*$ is the minimum cost of transforming $\boldsymbol{s}_1$ into $\boldsymbol{s}_2$ through a sequence of single-position edit operations, including insertion, deletion, and substitution.

Definition 4 (Robustness)**.** A watermarking scheme is considered robust if, for any watermarked sequence $\boldsymbol{y}$ from $\mathsf{Model}^{\mathsf{wat}}_{\mathsf{sk}}(\boldsymbol{x})$ and any sequence $\boldsymbol{y}'$ with the edit distance bounded by $d(\boldsymbol{y}, \boldsymbol{y}') \leq \gamma \max(|\boldsymbol{y}|, |\boldsymbol{y}'|)$ for some $\gamma > 0$, the detection function reliably identifies the watermark:

$$\mathsf{Detect}_{\mathsf{sk}}(\boldsymbol{y}') = \mathsf{true}. \tag{7}$$

Note that a sequence detected as watermarked is not necessarily generated by the watermarked model. Robustness ensures that a watermarked sequence and its close neighbors remain detectable, but this does not compromise the property of soundness.

3 Constructing watermarks through probabilistic automata

Noise distribution. In edit-distance-based watermarking, a noise sequence is first sampled given the secret key and then used during both generation and detection, where the edit distance measures how well the generated text aligns with the watermark. However, for a general distribution over noise sequences, computing such alignment scores can be computationally intractable. This challenge motivates our use of probabilistic automata (PA). Unlike standard automata, which characterize sets of sequences, a PA defines a probability distribution over sequences. We leverage this property to use PAs to model the noise distribution. Specifically, the PA enables us to sample noise sequence during generation while providing sufficient structure to compute edit-distance-based alignment scores efficiently.

We next introduce the relevant definitions that underlie our watermarking constructions.

Definition 5 (Probabilistic Automaton)**.** A probabilistic automaton (PA) is defined as a tuple $(Q, \Sigma, \delta, \pi_0, \pi_f)$, where (1) Q is a finite set of states; (2) Σ is a finite alphabet of input symbols; (3) $\delta : Q \times \Sigma \times Q \to [0, 1]$ is the transition probability function; (4) $\pi_0 : Q \to [0, 1]$ defines the initial probability of each state; (5) $\pi_f : Q \to [0, 1]$ defines the final probability of each state, such that, for every $q \in Q$, $\pi_f(q) + \sum_{a \in \Sigma} \sum_{q' \in Q} \delta(q, a, q') = 1$.

3.1 Watermarking schemes represented by probabilistic automata

In model-agnostic watermarking schemes, given a secret key sk, Φ_{sk} defines a distribution of the random variable $\boldsymbol{\xi} \in \Xi^*$ by applying Equation 6 autoregressively. One case of this distribution is the stochastic language recognized by a PA. We model this distribution using a hierarchical automaton. Specifically, for a PA $\mathcal{M} = (Q, \Sigma, \delta, \pi_0, \pi_f)$ with $\Sigma \subset \Delta(\Xi)$, the process starts from an initial state q_0. Each transition emits $\Phi_i \in \Sigma$, which is defined as a probability distribution over Ξ, from which noise $\xi_i \sim \Phi_i$ is sampled. The noise ξ_i, represented by a binary sequence, is used for decoding the next token y_i as described in Equation 5. Each probability distribution Φ_i is modeled by an inner PA with a binary alphabet. The hierarchical structure allows the PA to model the noise distribution represented by binary alphabet.

3.2 Constructing watermarks for language models

We now elaborate a specific construction of the watermarking scheme. We consider a decoder that uses exponential minimum sampling following Aaronson (2022); Kuditipudi et al. (2024). The decoder generates the next token based on a noise ξ and the model's output probabilities, which can be formally expressed as

$$\Gamma(\xi_i, \mathsf{Model}(\boldsymbol{y}_{1:i-1})) = \arg\min_{j \in \mathcal{V}} (\pi_j / \log(\mu_j)), \tag{8}$$

where π_j is the probability assigned by $\mathsf{Model}(\boldsymbol{y}_{1:i-1})$ to token $j \in \mathcal{V}$, and $\xi_i = (\mu_1, \ldots, \mu_{|\mathcal{V}|})$ with $\mu_j \overset{\text{i.i.d.}}{\sim} \text{Uniform}[0, 1]$. This decoder preserves the model's categorical distribution at each step.

We model the noise distribution using a hierarchical PA composed of two independent components. The outer PA governs the evolution of the watermark across decoding steps, capturing its progression over the sequence. The inner PA defines the token-level noise distribution used at each step. This decomposition is introduced purely for clarity, allowing the overall construction to be described and analyzed in a modular way.

The outer PA consists of λ states, denoted by $q_0, q_1, \ldots, q_{\lambda-1}$. Each state q_i transitions to $q_{i+1 \bmod \lambda}, \ldots, q_{i+d \bmod \lambda}$ with equal probability, thereby defining the distribution over positions in the watermark sequence.

The inner PA is defined over binary strings and produces the noise vector $\xi = (\mu_1, \ldots, \mu_{|\mathcal{V}|})$ used by the decoder for a single token. Intuitively, it encodes the random variables underlying exponential minimum sampling in a form that can be composed with the outer PA and handled by the detection algorithm. For each state, all outgoing transitions have equal probability. The full inner-PA construction, including the bit-level state layout, is deferred to Appendix H.1.

As a special case, when $d = 1$ and b, c are sufficiently large, this construction recovers the cyclic key sequence watermarking scheme of Kuditipudi et al. (2024). During generation, the PA sampling process is integrated into token generation as described in Algorithm 1.

We now proceed to describe the detection algorithm. We begin by defining a cost following Aaronson (2022) and Kuditipudi et al. (2024) as

$$d_0(y, \xi) = \log(1 - \mu_y), \tag{9}$$

where $\xi = (\mu_1, \mu_2, \ldots, \mu_{|\mathcal{V}|})$ and $\mu_i \in [0, 1]$ is represented by its binary expansion using the inner PA. For $\Phi \in \Delta(\Xi)$, the cost is defined as

$$d_0(y, \Phi) = \max_{\xi \in \text{supp}(\Phi)} \{d_0(y, \xi)\}. \tag{10}$$

Before we define the Levenshtein distance for PAs, we introduce the necessary definition.

Definition 6 (Support Automaton)**.** The support automaton of a PA $\mathcal{M} = (Q, \Sigma, \delta, \pi_0, \pi_f)$ is a non-deterministic finite automaton (NFA) $\underline{\mathcal{M}} = (Q, \Sigma, \delta, Q_0, Q_f)$, where Q_0 (respectively Q_f) denotes the set of initial (respectively final) states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition function defined as $(q, a, q') \in \delta \Leftrightarrow \phi(q, a, q') > 0$.

Algorithm 1 Watermarking Schemes Represented by PAs

Input: Model, PA $\mathcal{M}$, decoder Γ
Output Watermarked sequence $\boldsymbol{y}$

1: Sample an initial state q from $\mathcal{M}$.
2: **for** each decoding step i **do**
3: Transition q to the next state and obtain $\Phi_i \in \Delta(\Xi)$.
4: Sample $\xi_i \sim \Phi_i$.
5: Select the next token y_i using ξ_i via Equation 8.
6: **return** $\boldsymbol{y}$

Definition 7 (Generalized Levenshtein Distance)**.** For a sequence $\boldsymbol{y} \in \mathcal{V}^*, \boldsymbol{\Phi} \in \Delta^*(\Xi), \gamma_i, \gamma_d$, the Levenshtein distance between $\boldsymbol{y}$ and $\boldsymbol{\Phi}$ is defined recursively as

$$d_L(\boldsymbol{y}, \boldsymbol{\Phi}) = \min \begin{cases} \gamma_d|\boldsymbol{y}| + \gamma_i|\boldsymbol{\Phi}|, \text{ if } |\boldsymbol{y}| = 0 \\ \qquad\qquad\qquad \text{or } |\boldsymbol{\Phi}| = 0, \\ d_L(\boldsymbol{y}_{2:}, \boldsymbol{\Phi}) + \gamma_d, \\ d_L(\boldsymbol{y}, \boldsymbol{\Phi}_{2:}) + \gamma_i, \\ d_L(\boldsymbol{y}_{2:}, \boldsymbol{\Phi}_{2:}) + d_0(y_1, \Phi_1). \end{cases} \tag{11}$$

For a PA $\mathcal{M}$ defining a support language $\mathcal{L}(\underline{\mathcal{M}}) \subseteq \Delta^*(\Xi)$, the Levenshtein distance between $\boldsymbol{y}$ and $\mathcal{M}$ is defined as

$$d_L(\boldsymbol{y}, \mathcal{M}) = \min_{\boldsymbol{\Phi} \in \mathcal{L}(\underline{\mathcal{M}})} \{d_L(\boldsymbol{y}, \boldsymbol{\Phi})\}. \tag{12}$$

In Equation 11, the behavior of $\boldsymbol{y}$ and $\boldsymbol{\Phi}$ is not symmetric, as the length of $\boldsymbol{\Phi}$ can be infinite. Therefore, we assign different costs to insertion (γ_i) and deletion (γ_d). Meanwhile, the generalized Levenshtein distance is different from the design in Kuditipudi et al. (2024). In their algorithm, the lengths of the key sequence and the text are constrained to be equal, and it may disrupt their alignment when insertions or deletions are involved. In contrast, our definition allows flexible alignment despite difference in lengths.

The naive dynamic programming computes Equation 12 in $\mathcal{O}(dm\lambda \log \lambda))$ time (Wagner, 1974), whereas our PA construction reduces it to $\mathcal{O}(m\lambda)$. The dynamic programming algorithm is detailed in Appendix C.1.

Empirical p-value. We quantify the alignment between an observed sequence $\boldsymbol{y}$ and the watermarked model $\mathcal{M}_{\mathsf{sk}}$ by the statistic $\psi = d_L(\boldsymbol{y}, \mathcal{M}_{\mathsf{sk}})$, and treat as null hypothesis H_0 that $\boldsymbol{y}$ is independent of sk. We sample N independent keys $\{\mathsf{sk}_i\}_{i=1}^N$ and compute $\psi_i = d_L(\boldsymbol{y}, \mathcal{M}_{\mathsf{sk}_i})$ $(i = 1, \ldots, N)$. The empirical p-value is then

$$\hat{p} = \frac{1 + \sum_{i=1}^N \mathbf{1}[\psi_i \leq \psi]}{N+1}. \tag{13}$$

A small p-value indicates that $\boldsymbol{y}$ aligns unusually well with the true key (suggesting a watermark), whereas a larger p-value is consistent with unwatermarked noise.

An upper bound of p-value via Vysochanskij–Petunin Inequality. Let μ and σ^2 denote the sample mean and variance of $\{\psi_i\}_{i=1}^N$, given by $\mu = \frac{1}{N}\sum_{i=1}^N \psi_i$ and $\sigma^2 = \frac{1}{N-1}\sum_{i=1}^N (\psi_i - \overline{\psi})^2$, and define the standardized statistic

$$z = (\psi - \mu)/\sigma. \tag{14}$$

Under the mild assumption that the null distribution of d_L is unimodal, the one-sided Vysochanskij–Petunin inequality (Mercadier and Strobel, 2021) gives the following upper bound:

$$\Pr[Z \leq z] \leq \begin{cases} \dfrac{4}{9(z^2+1)} & \text{if } |z| \geq \sqrt{5/3}, \\ \dfrac{4}{3(z^2+1)} - \dfrac{1}{3} & \text{otherwise.} \end{cases} \tag{15}$$

We refer to this scheme as WEPA, for Watermarking schemE through Probabilistic Automata. The design of WEPA makes it robust and highly efficient for dynamic programming. As observed, disregarding bitwidth constraints, the generation diversity of WEPA is $\Theta(\lambda d^m)$, which is a substantial improvement over the construction of Kuditipudi et al. (2024).

4 Expressiveness of PA-based watermarking schemes

In this section, we draw connections between distortion-freeness, undetectability, and security levels in watermarking schemes. The security level is measured by the security parameter λ that determines the key size and reflects the computational hardness of breaking cryptographic schemes. It is assumed to be known to adversaries, and their running time is expressed as functions of λ rather than concrete values. An adversary is considered efficient if it can be modeled as a probabilistic algorithm with running time bounded by a polynomial function of λ. A function is called *negligible* if it becomes asymptotically smaller than the inverse of any polynomial as λ grows, which represents a probability that is practically negligible for large λ. Negligible functions can be expressed as $\mathrm{negl}(\lambda) = \mathcal{O}(\frac{1}{\mathrm{poly}(\lambda)})$ (i.e., functions that vanish faster than any inverse polynomial). While prior work briefly mentions the relationship between these notions (Golowich and Moitra, 2024), we provide a detailed analysis here and defer formal statements to Appendix D.

Distortion-freeness. Distortion-freeness ensures that the distribution of the unwatermarked model closely resembles the joint distribution of the watermarked model and the key. However we are

not interested to require them to be exactly identical, but the two distributions are similar enough that distinguishing between them requires computational effort beyond feasible limits. As a result, the quality of the unwatermarked model remains unchanged. An informal definition is given below, with the formal game-based definition deferred to Appendix D.1.

Definition 8 (Distortion-Freeness, Informal)**.** A watermarking scheme is *distortion-free* if, given a single sequence sampled from either the original model or the corresponding watermarked model, no polynomial-time adversary can distinguish the two induced distributions with non-negligible advantage (in the security parameter).

Distortion-freeness admits a stronger notion, *perfect distortion-freeness*, which intuitively requires that watermarking preserves the output distribution exactly. In this case, the randomness of the unwatermarked model can be viewed as arising from first sampling a secret key and then generating text conditioned on that key, without changing the resulting distribution. This guarantees that watermarking introduces no statistical distortion whatsoever. However, this requirement is often too strict and rarely satisfied in practice; for instance, prior constructions fail to meet it once the generated text exceeds the key length. Consequently, we focus instead on computational indistinguishability, which captures the practical goal that any detectable distortion remains beyond feasible computation. A formal statement and proof of the equivalence underlying perfect distortion-freeness are deferred to Appendix D.1.

Undetectability. While distortion-freeness ensures that watermarking does not alter the output distribution in a single query, it does not prevent detection when an adversary can make multiple, adaptive queries. Undetectability captures a stronger requirement: even with oracle access to a watermarked model, no efficient adversary should be able to reliably determine whether a given output was generated with or without a watermark.

Definition 9 (Undetectability, Informal)**.** A watermarking scheme is *undetectable* if, even after interacting with the watermarked model, no polynomial-time adversary can distinguish outputs generated by the watermarked model from those generated by the original model with non-negligible advantage (in the security parameter).

The formal definition is deferred to Appendix D.2. By definition it is clear that all undetectable watermarking schemes are distortion-free due to the stronger capabilities of adversaries. This implies that undetectable watermarking maintains the output distribution of the watermarked model unchanged. Conversely, any watermarking scheme that introduces distortion is also detectable. This further implies that the watermarks in Kirchenbauer et al. (2023); Zhao et al. (2024); Aaronson (2022); Kuditipudi et al. (2024) are all detectable.

The following theorem highlights the *expressiveness* (i.e., the extent to which the automaton represents various sequences) of PAs for constructing watermarking schemes.

Theorem 1. *(Abridged) Under certain conditions, there exists an undetectable watermarking scheme that can be represented by a PA if sparse LPN is hard.*

We defer the the full statement and proof in Appendix G. Within this framework, the cyclic key sequence watermarking scheme (Kuditipudi et al., 2024) appears as a special case with a simple cyclic automaton structure; we defer the explicit construction and example to Appendix H. Our approach, by contrast, uses more general PA constructions with substantially greater expressiveness. This observation highlights the broader expressiveness of PA-based watermarking schemes, as the automaton representation is sufficiently rich to capture watermarking schemes that achieve undetectability.

5 Empirical evaluation

We conducted an empirical evaluation of WEPA's statistical properties and robustness on LLaMA-3.2-3B (Dubey et al., 2024) and Mistral-7B (v0.3) (Jiang et al., 2023). Following prior work (Kirchenbauer et al., 2023; Kuditipudi et al., 2024), we generated watermarked continuations from the news-like subset of C4 (Raffel et al., 2020). We compared WEPA with three baselines: cyclic key sequence watermarking with exponential minimum sampling (EXP) (Kuditipudi et al., 2024), unigram green-red set watermarking (KGW) (Kirchenbauer et al., 2023), and unbiased watermarking (Unbiased) (Hu et al., 2024). KGW is not directly comparable because it may introduce noticeable distortion. For each experiment, we report the $1/3$, $1/2$ (median), and $2/3$ quantiles of p-values across 200 watermarked samples. Other details and hyperparameters are deferred to Appendix B.2. Additional empirical results are collected in Appendix A.

5.1 Varying text lengths

We varied the generation length of watermarked text from 4 to 20 tokens, as shown in Figure 2. The results highlight differences between the two language models due to variations in their generation entropy. Among the methods, WEPA ($d = 1$) and EXP consistently achieve the strongest detection performance. WEPA ($d = 2$) performs slightly worse, followed by KGW. The Unbiased method lags behind, especially on shorter sequences. We further report ROC-AUC and TPR@1%FPR in Appendix A.3. Notably, watermarking short text is inherently more challenging than long text. For completeness, we also include experiments on longer sequences (up to 200 tokens) in Appendix A.1.

5.2 Robustness to edit-based attacks

We evaluated the robustness against three types of edit-based attacks: (1) substitution, (2) deletion, and (3) insertion, each applied by randomly corrupting a fraction of tokens. For substitution and insertion, replacement tokens were sampled uniformly from the vocabulary. No padding was introduced, nor was truncation applied following deletions or insertions. All experiments were conducted on sequences of length 50 tokens.

Figure 3 reports results on LLaMA-3B, while we defer the results on Mistral-7B in Appendix A.6. WEPA ($d = 1$) performs slightly better than the other methods across all attack types, likely due to its flexible alignment mechanism. In contrast, EXP performs slightly worse, as it requires exact alignment between the generated text and a fixed-length key sequence. KGW shows reduced robustness, likely due to its unigram-based design. The Unbiased method is not robust under these attacks, as it was not designed to handle edit-based perturbations. We also evaluate a stronger semantic-preserving paraphrasing attack based on round-trip translation in Appendix A.7, where the edit-distance-based schemes remain more resilient than KGW.

5.3 Efficiency

Efficiency is the primary bottleneck of EXP. Its detection procedure is inherently expensive, as it requires repeatedly shifting the key sequence and recomputing the Levenshtein distance λ times. In contrast, our framework avoids this redundancy by extending the edit distance formulation to the PA, which allows the computation to be performed only once. This yields a speedup in a factor of λ. In addition, we incorporate an optimization of token discretization, which further provides a substantial improvement in runtime practically.

We compared the runtime of the detection algorithms for WEPA ($d = 1$), WEPA ($d = 2$) and EXP. For WEPA ($d = 2, b = 6$), we choose $b = 6$ as the bitwidth does not affect efficiency. The results are presented in Table 1. For EXP, we implemented an optimization with token discretization. Each algorithm was evaluated on a sample from C4 dataset with text length of 256 using LLaMA's tokenizer. WEPA is significantly faster than EXP, which is consistent with their theoretical time complexities.

6 Related work

Watermarking of language models. Text watermarking aims at blending a private "key" in text generation so that it can be detected by key holders. Early approaches relied on subtle text modifications using heuristics Atallah et al. (2001, 2002); Topkara et al. (2005). Following the autoregressive nature of language models (LMs), recent watermarking methods start to condition the token generation by the key and k prior tokens (Kirchenbauer et al., 2023; Aaronson, 2022; Zhao et al., 2024). However, these methods can significantly alter the underlying LM's distribution, for instance, by introducing biases for certain k-grams.

Watermarking without changing the next token distribution of the LM on a *single text sample* is defined as *distortion-free* watermarking (Kuditipudi et al., 2024; Hu et al., 2024). For example, Kuditipudi et al. (2024) uses a cyclic key sequence of noise for unbiased decoding and was the first to employ the *edit distance* metric as the alignment between text and the key sequence for detection. While this method enhances the robustness of watermarking against edit-based attacks, it suffers from two drawbacks of lacking generation diversity, and it relies on partitioning the text into blocks and repeatedly shifting the key sequence to compute edit distances multiple times, posing significant efficiency challenges.

Recent works have constructed *undetectable* watermarks theoretically that require watermarked texts to be indistinguishable across *multiple queries*, yet none of them remains practical. For instance, Christ et al. (2024)'s construction is based on hash functions and not robust to edit-based attacks. Christ and Gunn (2024) assume a binary symmetric channel model for LMs, which is clearly

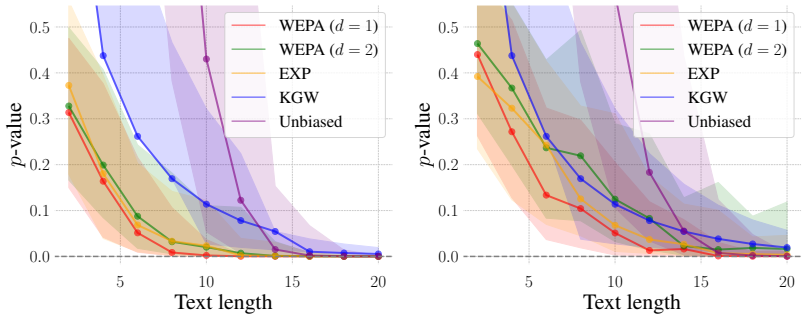

Figure 2: Median p-values varying text lengths on LLaMA-3B (left) and Mistral-7B (right).

Scheme	**Time** (s)
WEPA ($d = 1$)	7.81
WEPA ($d = 2$)	8.22
WEPA ($d = 1, b = 6$)	10.60
EXP (our impl.)	2039.56
EXP	49024.57

Table 1: Detection efficiency comparison. Unless otherwise specified, `float32` is used for b in WEPA.

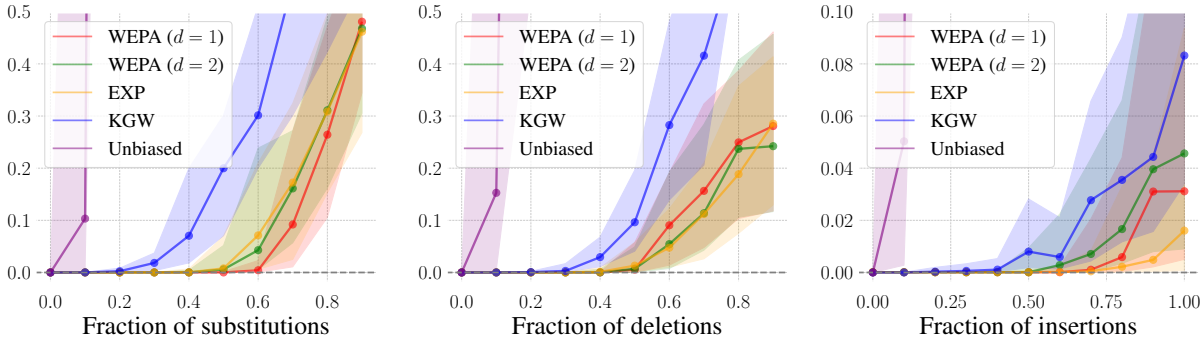

Figure 3: Median p-values under substitution (left), deletion (middle), and insertion (right) attacks.

unrealistic. Golowich and Moitra (2024) make an assumption that the vocabulary size scales polynomially with the security parameter (i.e., the size of the "key"), which does not typically hold in practice. Notably, all of these undetectable watermarking schemes rely on the construction of pseudorandom functions, yet they are based on disparate assumptions and lack a unified framework.

Probabilistic automata. Probabilistic automata (PA) are widely studied in computational linguistics that describe distributions with latent variables over finite sequences of symbols. In this paper, we use PA to denote the general probabilistic automaton model used to represent distributions over strings. The learnability of such automata has gained profound theoretical interest and practical relevance, particularly in modeling distributions over strings.

Kearns et al. (1994) explored the complexity of learning probabilistic automata within the Probably Approximately Correct (PAC) framework. In particular, they related the learnability of restricted automaton classes over a two-letter vocabulary to the hardness of learning noisy parities, which is believed to be computationally hard.

Edit distance and error correction. Edit distance quantifies the similarity between two strings by counting the minimum number of operations required to transform one into the other. This concept extends to formal languages, where the edit distance is defined as the minimum distance between any pair of sequences within the languages. This metric is particularly valuable in error correction, where it helps identify the closest valid string to a given input string. Wagner (1974) introduced an error-correcting algorithm that constructs a finite state automaton to recognize a set of strings. Their approach uses dynamic programming to compute the edit distance between a string and a regular language.

7 Conclusion

We introduced a class of watermarking schemes constructed through probabilistic automata. Within this framework, we instantiated WEPA, a practical watermarking method that achieves both improved generation diversity and more efficient detection. Empirical results further demonstrate its effectiveness and efficiency. Furthermore, by extending to probabilistic non-deterministic finite automata, we established an undetectable watermarking scheme.

References

Scott Aaronson. 2022. My ai safety lecture for ut effective altruism. Accessed May 2023.

Mikhail J Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. 2001. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Information Hiding: 4th International Workshop, IH 2001 Pittsburgh, PA, USA, April 25–27, 2001 Proceedings 4*, pages 185–200. Springer.

Mikhail J Atallah, Victor Raskin, Christian F Hempelmann, Mercan Karahan, Radu Sion, Umut Topkara, and Katrina E Triezenberg. 2002. Natural language watermarking and tamperproofing. In *International workshop on information hiding*, pages 196–212. Springer.

Andrej Bogdanov and Alon Rosen. 2017. Pseudorandom functions: Three decades later. In *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, pages 79–158. Springer.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Miranda Christ and Sam Gunn. 2024. Pseudorandom error-correcting codes. In *Annual International Cryptology Conference*, pages 325–347. Springer.

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

Alexander Clark and Franck Thollard. 2004. Pac-learnability of probabilistic deterministic finite state automata. *Journal of Machine Learning Research*, 5(May):473–497.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Noah Golowich and Ankur Moitra. 2024. Edit distance robust watermarks via indexing pseudorandom codes. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2024. Unbiased watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Michael Kearns, Yishay Mansour, Dana Ron, Ronitt Rubinfeld, Robert E Schapire, and Linda Sellie. 1994. On the learnability of discrete distributions. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 273–282.

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

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

Mathieu Mercadier and Frank Strobel. 2021. A one-sided vysochanskii-petunin inequality with financial applications. *European Journal of Operational Research*, 295(1):374–377.

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

Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, Gretchen Krueger, Jong Wook Kim, Sarah Kreps, and 1 others. 2019. Release strategies and the social impacts of language models. *arXiv preprint arXiv:1908.09203*.

Mercan Topkara, Cuneyt M Taskiran, and Edward J Delp III. 2005. Natural language watermarking. In *Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 441–452. SPIE.

Robert A Wagner. 1974. Order-n correction for regular languages. *Communications of the ACM*, 17(5):265–268.

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

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2024. Provable robust watermarking for AI-generated text. In *The Twelfth International Conference on Learning Representations*.

Limitations

Our work has several limitations that suggest directions for future research. First, the lack of tight analytical bounds on the p-value remains a challenge, as the edit distance metric is inherently difficult to analyze. Second, the detection algorithm requires knowledge of the private key, which limits detection to only the key holder. Developing an asymmetric decoding and detection system could improve security.

Ethical considerations

This study showcases recent advances in the field of Machine Learning, aimed at preventing the misuse of AI-generated content. Our technique endeavors to improve the security of text generation systems, thereby reducing the risk of misleading information and contributing to the establishment of ownership and copyright of AI-generated content. From an ethical perspective, this technology plays an important role in protecting the integrity of information dissemination on digital platforms and helps strengthen public trust in AI applications. While we recognize that these technologies may be used to restrict freedom of information or enforce censorship in some environments, we believe that their benefits in reducing misinformation outweigh the potential risks.

Appendix A. Additional empirical results

A.1 Varying text lengths on long texts

We evaluated WEPA and EXP on longer text sequences, as presented in Figure 4. For cases where the empirical p-values became exponentially small, direct computation was infeasible; instead, we report upper bounds estimated via Equation 14. Due to the looseness of this bound, these values are not directly comparable to KGW. Nonetheless, the results reveal consistent trends with those in Figure 2. Notably, WEPA ($d = 1$) achieves the strongest performance, while WEPA ($d = 2$) exhibits slightly lower accuracy compared to EXP.

A.2 Perplexity evaluation

Figure 5 presents the perplexity distributions across watermarking methods. KGW noticeably alters the text, while WEPA and Unbiased maintain perplexity levels comparable to unwatermarked outputs.

A.3 Varying text lengths with supplemental evaluation metrics

Figure 6 presents ROC-AUC results across varying text lengths, while Figure 7 reports the corresponding TPR@1%FPR results. Overall, WEPA ($d = 1$) achieves performance comparable to EXP on both metrics, with WEPA ($d = 2$) performing even better. Although the Unbiased method attains the highest scores on these metrics, it is not robust to edit-based attacks.

A.4 Varying key length

We varied the key length (λ) from 2^4 to 2^{12} while keeping the text length fixed at 8, as shown in Figure 8. Consistent with the findings of Kuditipudi et al. (2024), we observe that the p-value increases with λ.

A.5 Varying bitwidth

We varied the bitwidth (b) of WEPA from 2 to 12 while keeping the text length fixed at 8, as shown in Figure 9. The results indicate that a higher bitwidth improves performance with sacrifice of text diversity. The performance stabilizes when $b \geq 6$.

A.6 Robustness to edit-based attack on Mistral-7B

We evaluated robustness against edit-based attacks on Mistral-7B, following the setup in Section 5.2, with results shown in Figure 10. Under a lower-entropy model, KGW demonstrates greater stability. Overall, these schemes exhibit performance similar to the LLaMA-3B model.

A.7 Robustness against paraphrasing via translation attacks

Following Kuditipudi et al. (2024), we evaluate a round-trip translation attack by translating English watermarked texts of varying lengths into another language and then back into English using a machine translation model. We consider French and Russian as the intermediate languages. This attack is substantially stronger than the edit-based perturbations studied in Section 5.2, since it can alter lexical choice and local sentence structure while preserving the original semantics. We do not include Unbiased in this experiment, as it is not robust to such attacks and typically yields a p-value of 1, indicating complete failure of detection. The corresponding p-value and ROC-AUC results are shown in Figures 11 and 12, respectively.

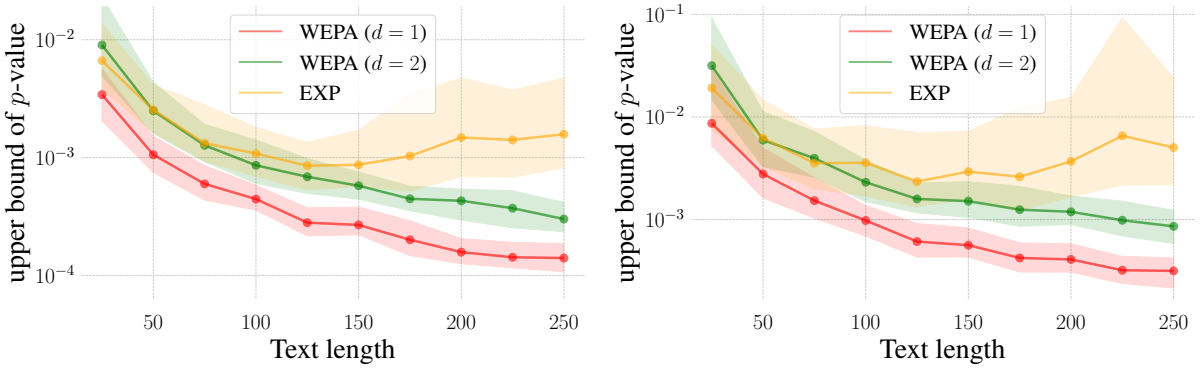

Figure 4: Median p-values varying text lengths on LLaMA-3B (left) and Mistral-7B (right).

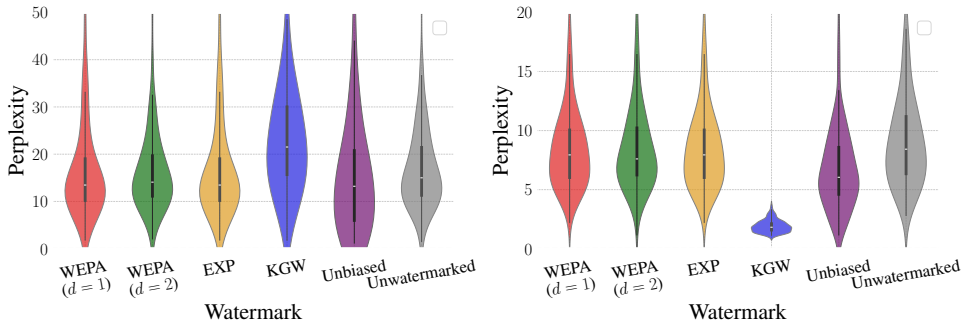

Figure 5: Perplexity distribution of each watermarking scheme on LLaMA-3B (left) and Mistral-7B (right).

Overall, we observe similar trends under both French and Russian round-trip translation attacks: WEPA and EXP consistently outperform KGW in terms of p-value, while WEPA ($d = 1$) achieves the strongest ROC-AUC. This suggests that, although translation-based paraphrasing weakens detection performance for all methods, the edit-distance-based schemes remain more resilient than unigram-based watermarking under this type of semantic-preserving perturbation.

Appendix B. Experiment Setup

B.1 Computation resources

Experiments were run on an internal cluster with dual AMD EPYC 7453 28-core CPUs (56 cores total), 1 TiB RAM, and 8 NVIDIA GeForce GPUs (24 GiB each, CUDA 11.6). Peak GPU memory usage reached approximately 15 GiB per device. The system provided ample memory and storage, with over 800 GiB RAM available during runs. However, our experiments primarily rely on GPU resources for language model inference, and do not require high-performance CPUs.

B.2 Hyperparameter settings

We detail the hyperparameter configurations used for all methods evaluated in our experiments. For WEPA, we set the key length λ to 256. For the cyclic key sequence watermarking baseline (EXP), we used a key sequence length of 256 and applied exponential minimum sampling with a soft Levenshtein cost parameter $\gamma = 0$. For the green-red set watermarking baseline (KGW) Kirchenbauer et al. (2023), we the green set fraction is set at 0.25 and the logit bias $\delta = 2$. For the unbiased watermarking method of Hu et al. (2024), we used the δ-reweight setting and report the upper bound on p-values described in that work, rather than the same Monte Carlo estimate used for WEPA and EXP.

For WEPA, unless specified, we used the key length of λ 256 and `float32` type to mimic the behavior when b is sufficiently large. For Levenshtein distance, we set the deletion cost $\gamma_d = 0$ and insertion cost $\gamma_i = 2$. Notably, γ_i cannot be too small as a lower insertion cost allows tokens to align with arbitrary states with minimal penalty. For WEPA and EXP, p-values were estimated via Equation 13 using a Monte Carlo sample size of 10,000. In the main experiments, we report quantiles over 200

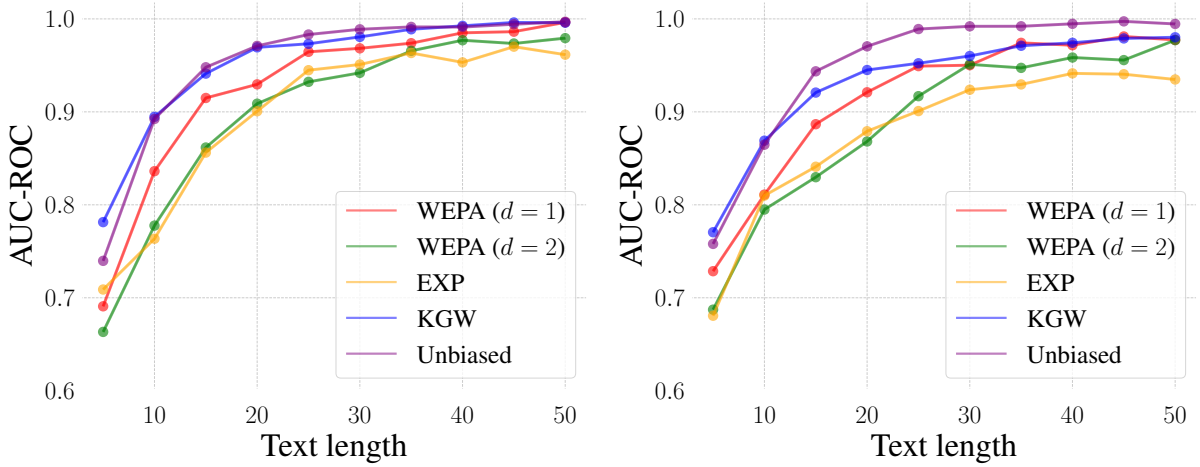

Figure 6: ROC-AUC results varying text lengths on LLaMA-3B (left) and Mistral-7B (right).

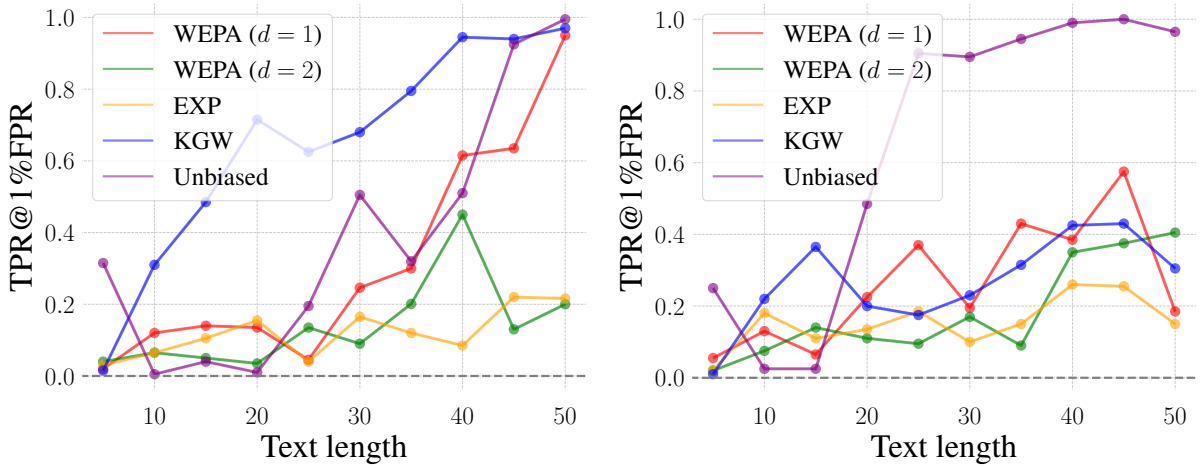

Figure 7: TPR@1%FPR results varying text lengths on LLaMA-3B (left) and Mistral-7B (right).

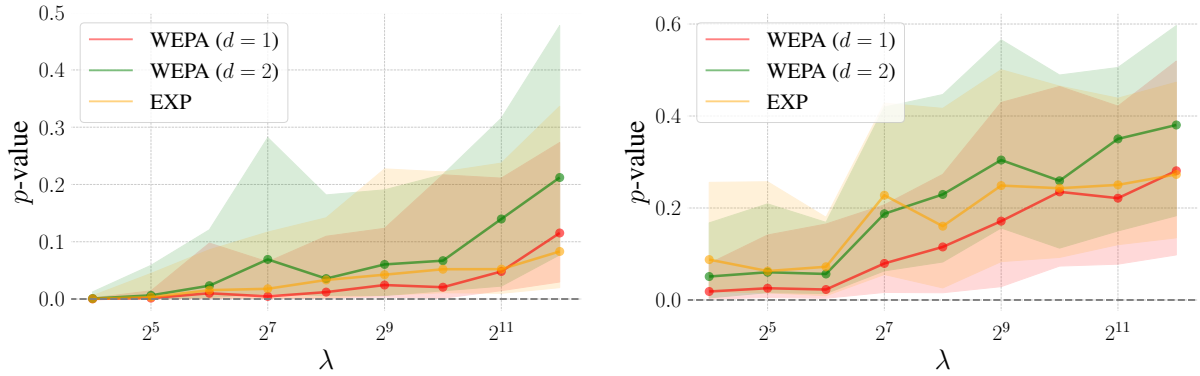

Figure 8: Median p-values varying λs on LLaMA-3B (left) and Mistral-7B (right).

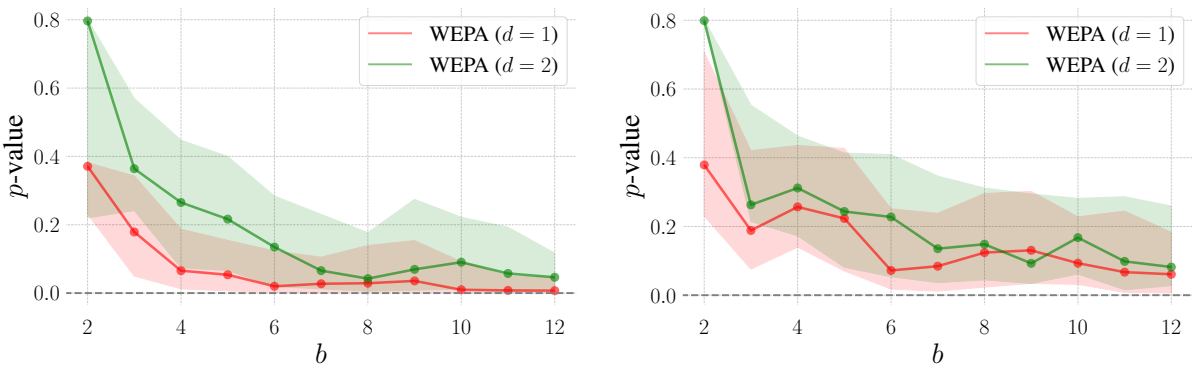

Figure 9: Median p-values varying bitwidths on LLaMA-3B (left) and Mistral-7B (right).

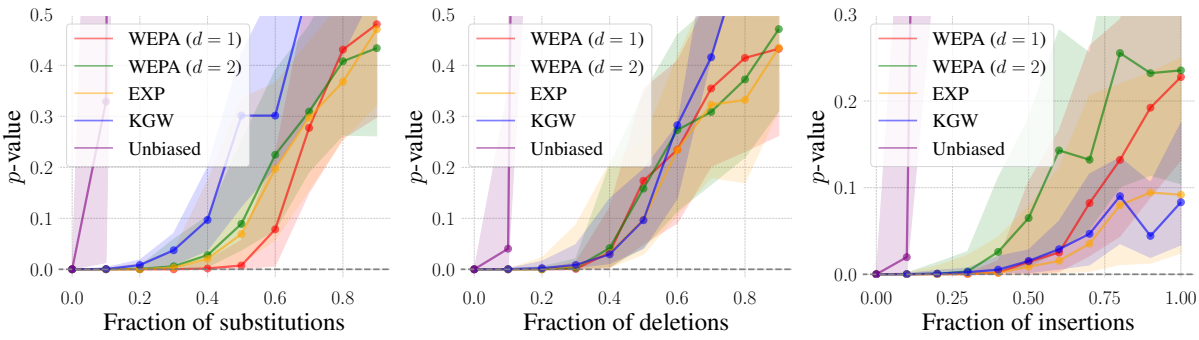

Figure 10: Median p-values under random substitution (left), deletion (middle), and insertion (right) attacks.

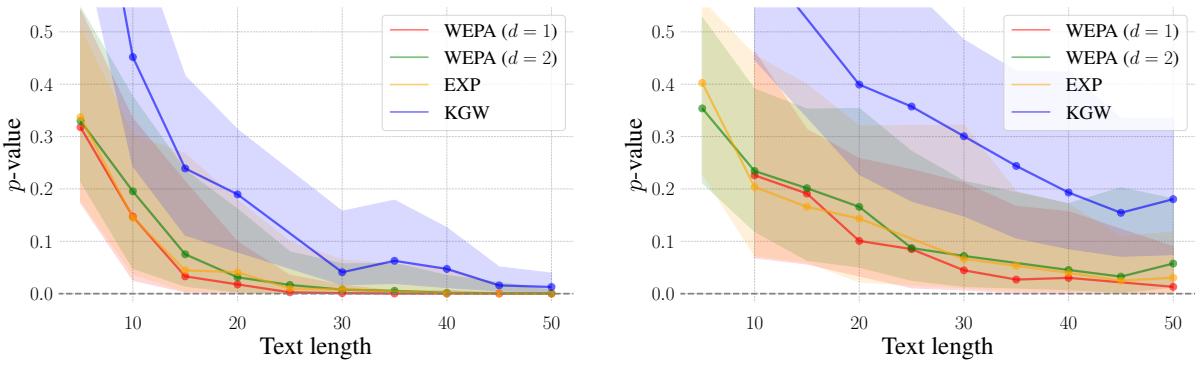

Figure 11: Median p-values under round-trip translation attacks with French (left) and Russian (right) as the intermediate languages.

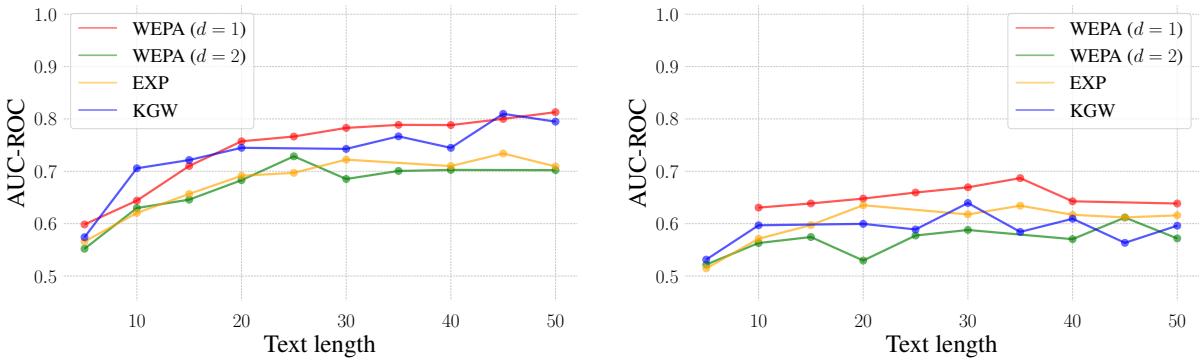

Figure 12: ROC-AUC under round-trip translation attacks with French (left) and Russian (right) as the intermediate languages.

watermarked samples.

Appendix C. Implementation details

C.1 Implementation of Levenshtein distance calculation

Algorithm 2 presents the dynamic programming for computing the Levenshtein distance used in WEPA. The $\text{cost}(\cdot, \cdot)$ array stores the precomputed costs between substates and tokens defined by Equation 10.

While the innermost loop can be optimized using a monotonic queue for a complexity of $\mathcal{O}(m\lambda)$, we do not carry out this optimization in our implementation in practice since the degree d is usually small.

Appendix D. Formal definitions and analysis of distortion-freeness and undetectability

In this section, we provide a formal analysis of the notions introduced in Section 4, including the formal definitions of distortion-freeness and undetectability, as well as the associated theorems.

D.1 Distortion-freeness.

We begin the definition by considering an adversary who has oracle access to the unwatermarked model (i.e., can make unlimited queries) and observes a single instance of watermarked data, and its goal is to determine whether the observed data is watermarked or not. While we make assumptions about the adversary's computational capabilities, we make no assumptions about its strategy, which means the definition provides protection against any computationally bounded adversary. The idea is that the adversary should be unable to extract any partial information that distinguishes watermarked data from unwatermarked data.

In addition, we do not make assumptions about the adversary's prompting strategy. The adversary can manipulate the model's behavior by prompting so that the model outputs arbitrary distribution. This ensures that the security definition remains robust against adversarial control over the model's output distribution. Since the adversary can shape the output distribution arbitrarily, the specific input to the model becomes irrelevant—choosing a prompt equates to choosing a model. Therefore, we can safely omit the model's input in our analysis.

The definition is formalized using the indistinguishability against one-time attacks ($\mathsf{IND\text{-}OT}$) game, defined for a watermarking scheme $\mathcal{W}$, an adversary $\mathcal{A}$, and the security parameter λ:

1. A secret key is generated as $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$.
2. The adversary $\mathcal{A}$ is given 1^λ and chooses a model $\mathsf{Model}(\cdot)$.[3]
3. A uniform bit $b \in \{0, 1\}$ is sampled. A sequence $\boldsymbol{y} \xleftarrow{\mathsf{AR}} f(\cdot)$ is generated and given to $\mathcal{A}$, where $f = \mathsf{Model}$ if $b = 0$, and $f = \mathsf{Model}^{\mathsf{wat}}_{\mathsf{sk}}$ otherwise.
4. The adversary $\mathcal{A}$ outputs a bit b'.

The advantage of $\mathcal{A}$ in this game is defined as

$$\text{Adv}^{\mathsf{IND\text{-}OT}}_{\mathcal{W}}(\mathcal{A}) := \left| \Pr\left[b = b'\right] - \frac{1}{2} \right|. \tag{16}$$

Definition 10 (Distortion-freeness)**.** A watermarking scheme $\mathcal{W}$ is *distortion-free* if it is indistinguishable against one-time attacks. Formally, for any polynomial-time $\mathcal{A}$,

$$\text{Adv}^{\mathsf{IND\text{-}OT}}_{\mathcal{W}}(\mathcal{A}) \leq \text{negl}(\lambda). \tag{17}$$

Moreover, a watermarking scheme is considered *perfectly distortion-free* if $\text{Adv}^{\mathsf{IND\text{-}OT}}_{\mathcal{W}}(\mathcal{A}) = 0$.

The following theorem provides an equivalent definition of perfect distortion-freeness.

Theorem 2. *A watermarking scheme is perfectly distortion-free iff for every language model that defines language distribution $\mathcal{L}$, every $\boldsymbol{y} \in \mathcal{L}$, it follows that*

$$\Pr[Y = \boldsymbol{y}] = \mathbb{E}_{\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)}\left[\Pr[Y = \boldsymbol{y} \mid K = \mathsf{sk}]\right], \tag{18}$$

where K is the random variable of the secret key.

We defer this proof to Appendix F.1. Theorem 2 indicates that the randomness in generating an output from the unwatermarked model can be equivalently represented by first sampling a key from the key space and then generating the sequence conditioned on that key. Consequently, the original distribution of the model's outputs is preserved. However, this condition is often too strict, and even Kuditipudi et al. (2024)'s construction is not perfectly distortion-free if the text length is greater than λ. Instead, our definition emphasizes *computational indistinguishability*, which is more attainable and aligns with the limitations of real-world systems.

[3] We omit the input $\boldsymbol{x}$ to Model, as choosing a prompt is equivalent to choosing a model.

Algorithm 2 Levenshtein Distance Calculation with Dynamic Programming

```
procedure LEVENSHTEINDISTANCE(y, λ, d, cost(·,·), γ_d, γ_i)
    m ← |y|
    f, g ← array of size λ initialized to 0                ▷ Assume f and g are cyclic.
    for i ← 1..m do
        for u ← 0..λ − 1 do
            d_0 ← cost(u, y_{i−1})
            f_u ← g_u + γ_d                                 ▷ Deletion
            for v ← u − d..u − 1 do
                f_u ← min(f_u, g_v + d_0)                   ▷ Substitution
        u* ← arg min(f_:)
        for u ← u* + 1..λ − 1 do
            for v ← u − d..u − 1 do
                f_u ← min(f_u, f_v + γ_i)                   ▷ Insertion
        for u ← 0..u* − 1 do
            for v ← u − d..u − 1 do
                f_u ← min(f_u, f_v + γ_i)                   ▷ Insertion
        f, g ← g, f
    return min(f_:)
```

D.2 Undetectability.

Distortion-freeness ensures that the quality of the language model is not changed by preserving the distribution in one query. It is nonetheless detectable with multiple queries, and a higher level of security is desired. Consider the following indistinguishability against chosen prompt attack (IND-CPA) game, defined for a watermarking scheme $\mathcal{W}$, an adversary $\mathcal{A}$, and the security parameter λ:

1. A secret key is generated as $\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)$.
2. The adversary $\mathcal{A}$ is given 1^λ and oracle access to $\mathsf{Model}'^{\mathsf{wat}}_{\mathsf{sk}}(\cdot)$ for any choice of Model', and then selects a model $\mathsf{Model}(\cdot)$.
3. A uniform bit $b \in \{0, 1\}$ is sampled. A sequence $\boldsymbol{y} \xleftarrow{\mathsf{AR}} f(\cdot)$ is generated and given to $\mathcal{A}$, where $f = \mathsf{Model}$ if $b = 0$, and $f = \mathsf{Model}^{\mathsf{wat}}_{\mathsf{sk}}$ otherwise.
4. The adversary $\mathcal{A}$ outputs a bit b'.

The advantage of $\mathcal{A}$ in this game is defined as

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{W}}(\mathcal{A}) := \left| \Pr\left[b = b'\right] - \frac{1}{2} \right|. \quad (19)$$

Definition 11 (Undetectability)**.** A watermarking scheme $\mathcal{W}$ is *undetectable* if, for all polynomial-time adversaries, there exists a negligible function negl such that

$$\mathrm{Adv}^{\mathsf{IND\text{-}CPA}}_{\mathcal{W}}(\mathcal{A}) \leq \mathrm{negl}(\lambda). \quad (20)$$

Appendix E. Additional analysis of existing watermarking schemes

In this section, we provide the alternative analysis of existing watermarking schemes through the lens of probabilistic automata (PA). In the main text, PA refers to the general probabilistic automaton model, namely the object that is often called a probabilistic nondeterministic finite automaton. For the detectability analysis in this appendix, we additionally define probabilistic deterministic finite automata (PDFA) as a restricted subclass of PA.

Definition 12 (Probabilistic Deterministic Finite Automaton)**.** Let $\mathcal{M} = (Q, \Sigma, \delta, \pi_0, \pi_f)$ be a PA. We say that $\mathcal{M}$ is a probabilistic deterministic finite automaton (PDFA) if: (1) $\exists q_0 \in Q$ such that $\pi_0(q_0) = 1$ and $\forall q \in Q \setminus \{q_0\}$, $\pi_0(q) = 0$, and (2) $\forall q \in Q, \forall a \in \Sigma$, there exists at most one state $q' \in Q$ such that $\delta(q, a, q') > 0$.

E.1 Watermarking schemes under uniform models

We assume a model that follows a uniform distribution over $\mathcal{V}$ so that $\mathsf{Model}(\cdot) = \mathrm{Cat}(1/|\mathcal{V}|, \ldots, 1/|\mathcal{V}|)$, where Cat denotes the categorical distribution. Then we proceed to analyze the output distribution of existing watermarking schemes under uniform models.

k-gram-based watermarking. k-gram-based watermarking has been introduced in many works

(Kirchenbauer et al., 2023; Zhao et al., 2024; Aaronson, 2022). Generally speaking, the core concept is to condition the next token on a window of k prior tokens. A hash function $h : \mathcal{K} \times \mathcal{V}^k \to \mathbb{Z}$ generates the noise based on the context of the k prior tokens. Then Φ computes the hash and alters the distribution and generates a variable:

$$\xi_i \sim \Phi(\boldsymbol{y}_{i-k:i-1}) = \Phi'(h_{\mathsf{sk}}(\boldsymbol{y}_{i-k:i-1})), \quad (21)$$

where $\Phi' : \mathbb{Z} \to \Delta(\Xi)$. The noise, ξ_i may be modeled as a uniform variable from $[0, 1]$ used for inverse transform sampling with a partition over the vocabulary (Kirchenbauer et al., 2023; Zhao et al., 2024), or a Gumbel variable (Aaronson, 2022).

Under uniform models, the output of a watermarking scheme can be represented by a PA $\mathcal{M} = (Q, \Sigma, \delta, \pi_0, \pi_f)$, with the state space $Q = \mathcal{V}^k$ and $\Xi = \mathcal{V}$, the transition function $\delta : Q \times \Sigma \times Q \to [0, 1]$ is defined as

$$\delta(\boldsymbol{y}_{i-k:i-1}, y_i, \boldsymbol{y}_{i-k+1:i}) = \mathbb{P}(Y = y_i \mid \boldsymbol{y}_{i-k:i-1}), \quad Y \sim \Phi(\boldsymbol{y}_{i-k:i-1}). \quad (22)$$

As each $\boldsymbol{y}_{i-k:i-1}$ and y_i uniquely determines $\boldsymbol{y}_{i-k+1:i}$, $\mathcal{M}$ is a PDFA.

Cyclic key sequence watermarking. Kuditipudi et al. (2024) introduced a watermarking scheme that uses a cyclic sequence of noise $\ldots \xi_{\lambda-1}\xi_0\xi_1 \ldots \xi_{\lambda-1}\xi_0 \ldots$ starting from random position in watermarks within generated text, where each $\xi_i \in [0, 1]$ for inverse transform sampling or $\xi_i \in [0, 1]^{|\mathcal{V}|}$ for exponential minimum sampling. The noise sequence is referred to as the key sequence.

Since the key sequence is cyclic, the output under uniform models also follows a repeated structure and can be recognized by a PDFA with at most 4λ states, which can be constructed with simliar ideas as suffix automata. We defer the construction in Appendix H.2.

E.2 Detectability of watermarking schemes

Definition 13 (KL-PAC Learnability)**.** Given a class of stochastic languages or distributions $\mathcal{C}$ over Σ^*, an algorithm $\mathcal{A}$ *KL-Probably Approximately Correctly (KL-PAC)-learns* $\mathcal{C}$ if there exists a polynomial q such that for all $c \in \mathcal{C}$, all $\epsilon > 0$ and $\delta > 0$, $\mathcal{A}$ is given a sample S_m and produces a hypothesis H satisfying

$$\mathbb{P}\left[D_{\mathsf{KL}}(c||H) > \epsilon\right] < \delta \quad (23)$$

whenever $m > q(1/\epsilon, 1/\delta, |c|)$, where $|c|$ is some measure of the complexity of the target. The algorithm runs in time polynomial in m plus the total length of the strings in S_m.

Theorem 3. *For any $\mu > 0$, any watermarking scheme the output of which under a uniform model represented by a μ-distinguishable PDFA with polynomial many states is detectable.*

Theorem 3 provides an alternative perspective to show that the watermarks proposed by Kirchenbauer et al. (2023); Zhao et al. (2024); Aaronson (2022); Kuditipudi et al. (2024) are all detectable and even spoofable.

Appendix F. Proof of technical results

F.1 Proof of Theorem 2

Proof. We begin we noticing that

$$\mathrm{Adv}_{\mathcal{W}}^{\mathsf{IND\text{-}OT}}(\mathcal{A}) = \left|\Pr\left[b = b'\right] - \frac{1}{2}\right| = 0 \quad \Leftrightarrow \quad \Pr\left[b = b'\right] = \frac{1}{2}. \tag{24}$$

Necessity. Fix an arbitrary output $\boldsymbol{y} \in \mathcal{L}$ for a given key $\mathsf{sk} \in K$. Suppose an adversary $\mathcal{A}$ attempts to distinguish between the original and watermarked models. The adversary may partition the output space $\mathcal{L}$ into two disjoint sets:

$$\mathcal{L}_0, \quad \mathcal{L}_1, \tag{25}$$

where $\mathcal{L}_0 \cap \mathcal{L}_1 = \varnothing$ and $\mathcal{L}_0 \cup \mathcal{L}_1 = \mathcal{L}$. The adversary's strategy is to output $b' = 0$ if $\boldsymbol{y} \in \mathcal{L}_0$ and $b' = 1$ otherwise. The probability of correctly guessing b is then given by:

$$\begin{aligned}\Pr[b = b'] &= \Pr[b' = 0 \mid b = 0]\Pr[b = 0] + \Pr[b' = 1 \mid b = 1]\Pr[b = 1] \\ &= \frac{1}{2}\left(\Pr[\boldsymbol{y} \in \mathcal{L}_0 \mid K = \mathsf{sk}, b = 0] + \Pr[\boldsymbol{y} \in \mathcal{L}_1 \mid K = \mathsf{sk}, b = 1]\right).\end{aligned} \tag{26}$$

Since the watermarking scheme $\mathcal{W}$ is assumed to be perfectly distortion-free, we have:

$$\Pr[Y = \boldsymbol{y}] = \mathbb{E}_{\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)}\Big[\Pr[Y = \boldsymbol{y} \mid K = \mathsf{sk}]\Big]. \tag{27}$$

Taking expectations over the key sk, it follows that:

$$\Pr[\boldsymbol{y} \in \mathcal{L}_0] = \mathbb{E}_{\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)}\left[\Pr[\boldsymbol{y} \in \mathcal{L}_0 \mid K = \mathsf{sk}]\right], \tag{28}$$

with a similar expression for $\mathcal{L}_1$. By the law of total probability, we obtain:

$$\Pr[\boldsymbol{y} \in \mathcal{L}_0] + \Pr[\boldsymbol{y} \in \mathcal{L}_1] = 1. \tag{29}$$

Substituting into the expression for $\Pr[b = b']$ and noting that b is independent of sk, we conclude:

$$\Pr[b = b'] = \frac{1}{2}\left(\Pr[\boldsymbol{y} \in \mathcal{L}_0] + \Pr[\boldsymbol{y} \in \mathcal{L}_1]\right) = \frac{1}{2}. \tag{30}$$

Since the adversary cannot achieve a probability of success greater than $\frac{1}{2}$, we conclude:

$$\mathrm{Adv}_{\mathcal{W}}^{\mathsf{IND\text{-}OT}}(\mathcal{A}) = \left|\Pr[b = b'] - \frac{1}{2}\right| = 0. \tag{31}$$

Thus, the adversary gains no distinguishing advantage, proving that the watermarking scheme $\mathcal{W}$ is perfectly indistinguishable.

Sufficiency. Conversely, assume that the watermarking scheme $\mathcal{W}$ is *not* perfectly distortion-free. Then there exists a language model with output distribution $\mathcal{L}$ and some $\boldsymbol{y} \in \mathcal{L}$ such that:

$$\Pr[Y = \boldsymbol{y}] \;\neq\; \mathbb{E}_{\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)}\big[\Pr[Y = \boldsymbol{y} \mid K = \mathsf{sk}]\big]. \tag{32}$$

Define the disjoint sets:

$$\mathcal{L}_0 \;=\; \{\boldsymbol{y} \in \mathcal{L} \mid \Pr[Y = \boldsymbol{y}] \;>\; \mathbb{E}_{\mathsf{sk} \leftarrow \mathsf{Gen}(1^\lambda)}\big[\Pr[Y = \boldsymbol{y} \mid K = \mathsf{sk}]\big]\}, \quad \mathcal{L}_1 \;=\; \mathcal{L} \setminus \mathcal{L}_0. \tag{33}$$

Consider an adversary $\mathcal{A}$ that outputs $b' = 0$ when $\boldsymbol{y} \in \mathcal{L}_0$ and $b' = 1$ when $\boldsymbol{y} \in \mathcal{L}_1$. The probability of a correct guess is given by:

$$\Pr[b = b'] = \frac{1}{2}\left(\Pr[\boldsymbol{y} \in \mathcal{L}_0 \mid b = 0] + \Pr[\boldsymbol{y} \in \mathcal{L}_1 \mid b = 1]\right). \tag{34}$$

Since $\boldsymbol{y} \in \mathcal{L}_0$ appears more frequently in the original model's distribution than in the watermarked model's, and vice versa for $\mathcal{L}_1$, it follows that:

$$\Pr[b = b'] > \frac{1}{2}. \tag{35}$$

Thus, the adversary gains a nonzero advantage, contradicting the assumption that $\mathcal{W}$ is perfectly distortion-free, which completes the proof. □

F.2 Proof of Theorem 3

Proof. Since any μ-distinguishable PDFA with a polynomial number of states and a bound on the expected length of strings generated from any state is KL-PAC-learnable (Clark and Thollard, 2004). That is, given sufficiently many samples from the distribution induced by a μ-distinguishable PDFA, an efficient learning algorithm can approximate the distribution arbitrarily well.

Suppose, for the sake of contradiction, that there exists a watermarking scheme $\mathcal{W}$ that produces outputs indistinguishable from those of an unwatermarked uniform model represented by a μ-distinguishable PDFA $\mathcal{M}$. That is, for any adversary $\mathcal{A}$,

$$\mathrm{Adv}^{\mathrm{ind}}_{\mathcal{W}}(\mathcal{A}) = \left|\Pr[b = b'] - \frac{1}{2}\right| = 0. \tag{36}$$

This implies that no efficient adversary can distinguish between the original model $\mathcal{M}$ and the watermarked model $\mathcal{M}'$ with non-trivial probability.

However, since $\mathcal{M}$ is μ-distinguishable, it is KL-PAC-learnable. Thus, there exists a polynomial-time algorithm $\mathcal{L}$ that, given polynomially many samples from the output distribution of either $\mathcal{M}$ or $\mathcal{M}'$, can learn a hypothesis h that approximates the true distribution up to any desired accuracy ϵ.

Consider the following adversary $\mathcal{A}$ that attempts to distinguish between $\mathcal{M}$ and $\mathcal{M}'$:

1. Draw $m = \mathrm{poly}(1/\epsilon, 1/\delta)$ samples from the given black-box model.
2. Run the KL-PAC-learning algorithm $\mathcal{L}$ to obtain a hypothesis h that approximates the underlying distribution.
3. Compute the likelihood of the observed samples under both the learned approximation of $\mathcal{M}$ and $\mathcal{M}'$.
4. Output $b' = 0$ if the samples are closer to the expected distribution of $\mathcal{M}$ and $b' = 1$ otherwise.

Since $\mathcal{M}$ is μ-distinguishable, every pair of states in $\mathcal{M}$ induces suffix distributions that differ by at least μ in ℓ_∞ norm, and $\mathcal{M}$ terminates with $m = \mathrm{poly}(\lambda)$ steps. If the watermarking scheme $\mathcal{W}$ alters the output distribution, then by definition of μ-distinguishability, it must introduce a statistically significant difference that is detectable given polynomially many samples.

Therefore, the KL-PAC-learning algorithm $\mathcal{L}$ enables the adversary $\mathcal{A}$ to distinguish between the original and watermarked models with probability strictly greater than $\frac{1}{2} + \gamma$ for some $\gamma > 0$. This contradicts our initial assumption that the watermarking scheme is perfectly indistinguishable.

Hence, we conclude that for any $\mu > 0$, any watermarking scheme applied to a uniform model represented by a μ-distinguishable PDFA with a polynomial number of states is necessarily detectable. □

Since all watermarking schemes proposed in Kirchenbauer et al. (2023); Zhao et al. (2024); Aaronson (2022); Kuditipudi et al. (2024) can be represented as a PDFA, and each pair of states in these automata is distinguishable (except in the unlikely case where two randomly chosen partitions are identical or the corresponding uniform random variables are nearly indistinguishable) it follows that all such watermarking schemes are necessarily detectable.

Appendix G. Undetectable watermark construction

Before presenting the watermark construction, we introduce the necessary definitions and assumptions.

Definition 14 (PAC Learnability)**.** Let $\mathcal{C}$ be a concept class consisting of Boolean functions $c : \mathcal{X} \to \{0, 1\}$ over some instance space $\mathcal{X}$. We say that an algorithm $\mathcal{A}$ *Probably Approximately Correctly (PAC)-learns* $\mathcal{C}$ if there exists a polynomial q such that for all target concepts $c \in \mathcal{C}$, all distributions $\mathcal{D}$ over $\mathcal{X}$, and all

$\epsilon, \delta > 0$, the algorithm $\mathcal{A}$, given access to i.i.d. examples drawn from $\mathcal{D}$ labeled by c, outputs a hypothesis h such that

$$\Pr\left[\ \Pr_{x\sim\mathcal{D}}\left[h(x) \neq c(x)\right] > \epsilon\ \right] < \delta \tag{37}$$

whenever the number of examples $m > q(1/\epsilon, 1/\delta, |c|)$, where $|c|$ denotes a measure of the complexity of the target concept. The algorithm runs in time polynomial in m plus the size of the input examples.

Assumption 4 (Sparsely Learning Parities with Noise)**.** Let $\mathcal{F} = (F_n)_{n\in\mathbb{N}}$ be the class of parity functions $F_n = \{f_s(x) = \bigoplus_{i\in[n]} x_i s_i \mid s \in \{0,1\}^n,\ \|s\|_1 = \log n\}$, where $x \in \{0,1\}^n$. Each f_s computes the parity of a logarithmic-sized subset of the input, defined by the support of s. No polynomial-time algorithm can PAC learn $\mathcal{F}$ under classification noise rate $q = 1/3$.

Definition 15 (Entropy Bound of Language Model)**.** Let $(y_1, y_2, \ldots, y_n)$ be a sequence of tokens generated by a language model over a finite vocabulary $\mathcal{V}$. The model assigns a conditional distribution $p(y_i \mid \boldsymbol{y}_{1:i-1})$ at each position i. We define the entropy bound of a language model as

$$\underline{H}(\mathsf{Model}) = \inf\left\{\mathbb{E}_{i\sim\mathcal{U}[n]}\left[H\left(y_i \mid \boldsymbol{y}_{1:i-1}\right)\right]\right\}, \tag{38}$$

where $H(y_i \mid \boldsymbol{y}_{1:i-1})$ denotes the conditional entropy and the expectation is taken over a uniformly random index $i \in [n]$.

Definition 16 (Weak Pseudorandom Function Family)**.** A family of functions $\mathcal{F} = \{f_s : \{0,1\}^n \to \{0,1\}\}_{s\in\{0,1\}^n}$ is a *weak pseudorandom function (PRF) family* if for every probabilistic polynomial-time adversary $\mathcal{A}$, the distinguishing advantage

$$\left|\Pr_{s\sim\{0,1\}^n}[\mathcal{A}^{f_s}(1^n) = 1] - \Pr_{f\sim\mathcal{U}}[\mathcal{A}^{f}(1^n) = 1]\right| \leq \mathrm{negl}(n), \tag{39}$$

where $\mathcal{U}$ denotes the uniform distribution over all functions $f : \{0,1\}^n \to \{0,1\}$, and the oracle queries to $\mathcal{A}$ are drawn uniformly at random from $\{0,1\}^n$.

Proposition 5 (From PAC Unlearnability to Weak Pseudorandomness)**.** *Let $\mathcal{F} = (\mathcal{F}_n)_{n\in\mathbb{N}}$ be a class of functions that is not PAC learnable under classification noise rate q. Then, for $n(\lambda) = \lambda$, the sequence of functions $(\mathcal{F}_{n(\lambda)})_\lambda$ constitutes a weak pseudorandom function family with noise level q.*

Proposition 5 is well known (see, for example, (Bogdanov and Rosen, 2017)), and we present it here in the same formulation as in (Golowich and Moitra, 2024).

G.1 Full Statement and Proof of Theorem 1

Reduction to a binary vocabulary. For analytical convenience, we reduce the vocabulary of $\mathcal{V}$ to a binary vocabulary $\{0,1\}$. Any categorical distribution over $\mathcal{V}$ with $|\mathcal{V}|$ can be equivalently represented by a binary distribution over $\{0,1\}^{\lceil\log_2|\mathcal{V}|\rceil}$ via entropy-preserving encoding. That is, given logits $p \in \Delta^{|\mathcal{V}|-1}$, we define a mapping to binary sequences where each symbol $v \in \mathcal{V}$ is assigned a unique bitstring $b(v) \in \{0,1\}^{\lceil\log_2|\mathcal{V}|\rceil}$, and generation proceeds bit by bit using the induced marginal distributions. This reduction preserves perplexity and generation quality up to negligible statistical error. We therefore assume, without loss of generality, that the model outputs binary tokens.

We present the full statement of Theorem 1 as follows.

Theorem 6. *There exists an undetectable watermarking scheme with generation length $n = \Omega(\lambda^{3+c}\log\lambda)$ for some $c > 0$, entropy bounded by $\underline{H}(\mathsf{Model}) > 2 + \log_2(1-q) - \frac{3-4q}{4(1-q)}\log_2(3-4q)$ that can be represented by a PA, if sparse LPN is hard with noise level of $0 < q < \frac{1}{2}$.*

Proof. Let $f_{\boldsymbol{s}}(\boldsymbol{x}) = \boldsymbol{s}\cdot\boldsymbol{x} \bmod 2$ be a parity function with noise level of q on the support of a secret key $\boldsymbol{s} \in \{0,1\}^\lambda$, and suppose a language model with binary output alphabet $\{0,1\}$.

At each token position i, we generate a pair of Gumbel noise variables (μ_0, μ_1) sampled from $U[0,1]^2$, whose binary representations are given by Equation 49.

If $(\lambda+1) \nmid i$, the pair (μ_0, μ_1) is left unmodified. Otherwise, when $(\lambda+1) \mid i$, we extract a watermark bit x_i via the indicator

$$x_i = \mathbf{1}\left[\mu_0 < \mu_1\right], \tag{40}$$

compute the parity bit $b = f_{\boldsymbol{s}}(\boldsymbol{x})$ over the current watermark buffer $\boldsymbol{x}$, and enforce $x_i = b$ by outputting (μ_0, μ_1) with probability $1 - q$ if $b = 0$ or with probability q and $b = 1$, otherwise outputting (μ_1, μ_0). This is easily accomplished by a PA by counting the number of bits of $\boldsymbol{s}$. The decoder of Equation 8 is used for each step to generating y_i.

The detector identifies all positions i such that $(\lambda + 1) \mid i$ and reconstructs the corresponding Gumbel pairs (μ_0, μ_1) used during generation. It computes the bit

$$x_i = \mathbf{1}[\mu_0 < \mu_1], \tag{41}$$

and compares it against the output token $y_i \in \{0, 1\}$. For each such position, let $z_i = \mathbf{1}[x_i = y_i]$ denote a match indicator. Over $t = \Omega(n/\lambda)$ such comparisons, the detector computes the empirical match rate:

$$\hat{p} = \frac{1}{t} \sum_{i=1}^{t} z_i. \tag{42}$$

The detector accepts (i.e., concludes the presence of a watermark) if $\hat{p} \geq \frac{1}{2} + \theta$ for some threshold $\theta = \Omega(\lambda^{-1})$, and rejects otherwise.

Undetectability. Let D_S denote the distribution over input-output pairs induced by our construction, where the input $\boldsymbol{x} \in \{0, 1\}^\lambda$ is uniformly random and the output bit is given by $f_{\boldsymbol{s}}(\boldsymbol{x})$ with noise rate q. This corresponds to the standard *noisy parity* distribution: with probability $1 - q$, the output is $f_{\boldsymbol{s}}(\boldsymbol{x})$, and with probability q, it is flipped.

Suppose there exists an algorithm $\mathcal{A}$ that KL-PAC-learns the class of such distributions, i.e., for any $\epsilon > 0$ and $\delta > 0$, given m samples drawn from D_S, the algorithm produces a hypothesis distribution $\hat{D}$ such that

$$\Pr\left[D_{\mathsf{KL}}(D_S \,\|\, \hat{D}) > \epsilon\right] < \delta \tag{43}$$

whenever $m > q(1/\epsilon, 1/\delta, |D_S|)$ for some polynomial q.

Now consider a random input $\boldsymbol{x} \in \{0, 1\}^\lambda$. Since the output bit is either $f_{\boldsymbol{s}}(\boldsymbol{x})$ or its complement, the two candidate outputs are $\boldsymbol{x}0$ and $\boldsymbol{x}1$. A hypothesis $\hat{D}$ that approximates D_S in KL-divergence necessarily assigns higher likelihood to the correct label in expectation. Therefore, by comparing the probabilities $\hat{D}(\boldsymbol{x}0)$ and $\hat{D}(\boldsymbol{x}1)$, one can predict $f_{\boldsymbol{s}}(\boldsymbol{x})$ with probability at least $1 - \epsilon$.

This yields an efficient algorithm that predicts noisy parity with non-negligible advantage, contradicting the Noisy Parity assumption, and thus also the hardness of sparse LPN.

Finally, by Proposition 5, the parity function $f_{\boldsymbol{s}}$ can be computed by a probabilistic automaton, and the corresponding output distribution D_S serves as a pseudorandom function. In particular, since D_S is not KL-PAC-learnable under the hardness of sparse LPN, it is indistinguishable from a uniform distribution by any efficient adversary.

Therefore, no efficient detector can distinguish watermarked text from unwatermarked text with non-negligible advantage.

Completeness. Let $p_0 \in (0, 1/2]$ denote the smallest possible bias such that the binary entropy of y_i satisfies

$$H(y_i) = -p_0 \log_2 p_0 - (1 - p_0) \log_2 (1 - p_0). \tag{44}$$

Consider the probability that the sampled watermark bit x_i matches the LM output y_i when $i \mid (\lambda + 1)$. Since $x_i = b$ with probability $1 - q$, and is flipped with probability q (due to the noise in $f_{\boldsymbol{s}}$), we have the following bound:

$$\begin{aligned} \Pr\left[x_i = y_i\right] &\geq \left[2p_0 \left(\frac{1}{2}\right)^{\log \lambda} (1 - q) + \frac{1}{2}\left(1 - \left(\frac{1}{2}\right)^{\log \lambda}\right)\right] \\ &= \frac{1}{2} + \lambda^{-1} \cdot \left(2p_0 - 2qp_0 - \frac{1}{2}\right) \\ &= \frac{1}{2} + c\lambda^{-1}, \end{aligned} \tag{45}$$

where $c := 2p_0 - 2qp_0 - \frac{1}{2} > 0$ by the entropy bound.

Then we have the following Chernoff bound:

$$\Pr\left[\hat{p} < \frac{1}{2} + \theta\right] \leq \exp\left(-2t(c\lambda^{-1} - \theta)^2\right) = \exp\left(-\frac{tc^2}{2\lambda^2}\right) = \text{negl}(\lambda). \tag{46}$$

Soundness. For unwatermarked text, assuming the Gumbel samples and tokens are independent (i.e., no watermarking mechanism was used), the match indicators z_i are i.i.d. unbiased coin flips:

$$\Pr[x_i = y_i] = \frac{1}{2}. \tag{47}$$

Then by Hoeffding's inequality,

$$\begin{aligned} \Pr\left[\hat{p} \geq \tfrac{1}{2} + \theta\right] &\leq \exp(-2t\theta^2) \\ &\leq \exp(-\lambda^c \log \lambda) \\ &= \text{negl}(\lambda), \end{aligned} \tag{48}$$

which indicates that the detector rejects unwatermarked text with overwhelming probability.

□

Appendix H. Construction details

H.1 Inner-PA construction

To complement the high-level description in Section 3.2, we give the detailed construction of the inner probabilistic automaton (inner PA) used in our watermarking framework, together with an illustration. The inner PA is responsible for generating the binary noise vectors $\mu_i \in \{0,1\}^c$ across vocabulary positions, as defined in the main text.

Since the inner PA operates over binary strings, the underlying continuous random variables are represented through binary approximations. In particular, any real number $z \in [0,1)$ can be approximated to precision c by

$$z \approx \frac{1}{2^c} \sum_{i=0}^{c-1} 2^i \sigma_i, \quad \sigma_i \in \{0,1\}, \tag{49}$$

where σ_i denotes the i-th bit in the binary expansion of z.

The inner PA is defined over binary strings and outputs a binary string of length $c|\mathcal{V}|$, which represents the vector $\xi = (\mu_1, \ldots, \mu_{|\mathcal{V}|})$ used by the decoder for a single token, where each $\mu_i \in \{0,1\}^c$. It begins at an initial state q_0 and terminates at a final state q_f, progressing through $|\mathcal{V}|$ layers, one for each vocabulary position. Within layer i, the automaton first traverses bitwise states $\sigma_{i,1}, \ldots, \sigma_{i,b}$ that encode the first b bits of μ_i. It then branches into two Boolean paths, denoted by $\iota_{i,j}$ and $\hat{\iota}_{i,j}$, which represent bit values 0 and 1 respectively and continue the remaining $c - b$ bits. The terminal states of layer i connect to the initial state of layer $i+1$, and after the final layer the automaton transitions to q_f.

More concretely, the construction is as follows. The first layer starts with $q_0 \to \sigma_{1,1}$. For each $1 \leq i \leq |\mathcal{V}|$ and $1 \leq j < b$, the automaton transitions as $\sigma_{i,j} \to \sigma_{i,j+1}$, where $\sigma_{i,j}$ encodes the j-th bit of μ_i. At $\sigma_{i,b}$, the automaton branches into two paths:

$$\sigma_{i,b} \to \iota_{i,b+1}, \qquad \sigma_{i,b} \to \hat{\iota}_{i,b+1}.$$

For $b \leq j < c$, these paths continue according to

$$\iota_{i,j} \to \iota_{i,j+1}, \quad \iota_{i,j} \to \hat{\iota}_{i,j+1}, \quad \hat{\iota}_{i,j} \to \iota_{i,j+1}, \quad \hat{\iota}_{i,j} \to \hat{\iota}_{i,j+1}.$$

Between layers, transitions connect

$$\iota_{i,c} \to \sigma_{i+1,1}, \qquad \hat{\iota}_{i,c} \to \sigma_{i+1,1},$$

and after the final layer the automaton concludes with $\iota_{|\mathcal{V}|,c}, \hat{\iota}_{|\mathcal{V}|,c} \to q_f$. For each state, all outgoing transitions have equal probability. When b and c are sufficiently large, the resulting binary strings can be interpreted as floating-point approximations of the random variables used in exponential minimum sampling.

As shown in Figure 13, the inner PA progresses from the initial state q_0 through $|\mathcal{V}|$ layers, each corresponding to a token in the vocabulary. The Boolean branching ensures exponential variability in possible noise sequences, while the overall number of automaton states remains polynomially bounded. This property is essential for both efficient sampling and tractable computation of edit distance during watermark detection.

H.2 Automaton construction for a cyclic string

We provide the construction of an automaton for a cyclic string in Algorithm 3. Given the input string s, the automaton has at most $4|s|$ states. Figure 14 illustrates an example PDFA that recognizes a substring of the repeated string of "abaa".

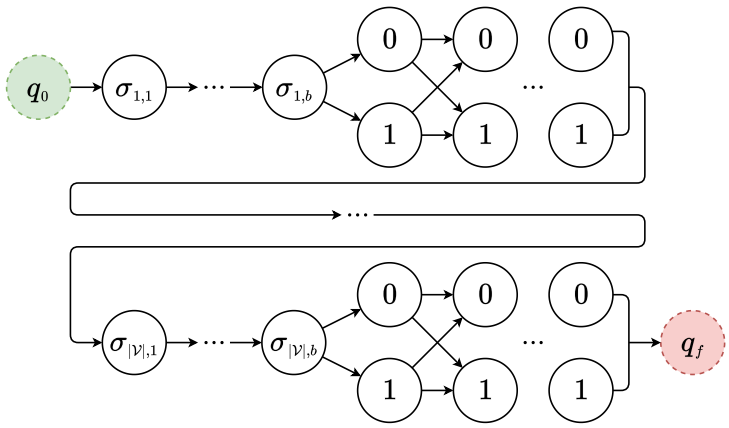

Figure 13: Illustration of the inner probabilistic automaton. Each layer encodes a binary vector μ_i, beginning from the initial state q_0 and terminating at q_f. Transitions within each layer encode bitwise states, branching into parallel Boolean paths $\iota_{i,j}$ and $\hat{\iota}_{i,j}$, which represent 0 and 1 respectively. Inter-layer transitions connect the terminal states of layer i to the initial states of layer $i+1$.

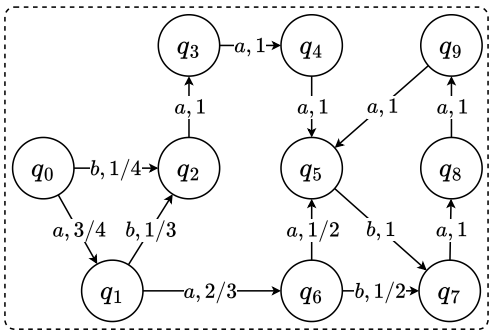

Figure 14: An example of a PDFA with probability that recognizes a substring of repeated string “abaa”.

Algorithm 3 Automaton Construction for a Cyclic String

```
Initialize: Let S be a set of states representing the suffix automaton, where each state S_i has:
  • S_i.length: The length of the longest string ending at S_i.
  • S_i.link: The suffix link, pointing to the longest proper suffix of the corresponding string that is
    also in the automaton.
  • S_i.next: A transition function mapping each character c to the corresponding state.
Create the initial state S_0 with S_0.length = 0 and S_0.link = −1.
last ← 0                                                    ▷ Tracks the last added state
function EXTEND(c)
    p ← last
    Create a new state S_cur with S_cur.length = S_p.length + 1
    while p ≠ −1 and c ∉ S_p.next do
        S_p.next[c] ← cur
        p ← S_p.link
    if p = −1 then
        S_cur.link ← 0
    else
        q ← S_p.next[c]
        if S_p.length + 1 = S_q.length then
            S_cur.link ← q
        else
            Create a new state S′ with S′.length = S_p.length + 1
            S′.next ← S_q.next
            S′.link ← S_q.link
            while p ≠ −1 and S_p.next[c] = q do
                S_p.next[c] ← S′
                p ← S_p.link
            S_q.link ← S′
            S_cur.link ← S′
    last ← cur
function BUILDAUTOMATON(s)
    for each character c in s do
        EXTEND(c)
    idx ← index of the newest state in the automaton
    for each character c in s do
        EXTEND(c)
    S_last.next[s_0] ← idx
```