# A PRACTICAL AND UNDETECTABLE WATERMARK FOR LANGUAGE MODELS USING PSEUDORANDOM CODES

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

The strongest quality notion for language model watermarks is *undetectability*, which ensures that the watermarked model is computationally indistinguishable from the original model. This notion is powerful, implying that the watermark preserves the model's performance on any task that can be run in polynomial time. However, until now all undetectable watermarks either lacked nontrivial robustness (Christ et al., 2024) or had no practical implementation (Christ & Gunn, 2024). The latter construction uses *pseudorandom codes* (PRCs), whose codeword length corresponds to the number of tokens required for detection. Achieving nontrivial robustness required infeasibly long codewords, on the order of thousands of tokens. In this work, we significantly reduce the number of tokens required for detection in the PRC watermarking scheme of Christ & Gunn (2024) by introducing a new *entropy-aware* detector. We show that on the Qwen family of models, reliable detection is possible given only $\sim$450 tokens. For the first time, this brings robust and undetectable LLM watermarks into the realm of practicality. Empirically, we find that the PRC watermark resists black-box detection and preserves quality, as measured by benchmark performance and output diversity. Compared with state-of-the-art baselines, our watermark offers greater resistance to the evaluated watermark-stealing attacks than SynthID-Text (Dathathri et al., 2024) and more diverse responses than TextSeal (Sander et al., 2026). Thus, we provide the first practical implementation of PRC watermarking for language models.

## 1 INTRODUCTION

Watermarks have emerged as a promising tool for identifying AI-generated content. A watermark influences the generation process, embedding a pattern that is visible to a corresponding detection algorithm. Watermarks for image models have already been widely deployed, by organizations including Google, OpenAI, Meta, and Amazon (Laurie Richardson, 2024; OpenAI, 2024; Nick Clegg, 2024; Antje Barth, 2025). However, text watermarks have lagged behind, adopted only by Google (Google DeepMind) and very recently by Anthropic (Claude Support, 2026).

Anthropic has explicitly stated that it implemented text watermarks in response to the EU AI Act, which requires AI-generated content including text to be marked as such. In order to comply, other organizations will likely follow suit, raising the need for the development of reliable text watermarks.

Two central challenges are *robustness*, the ability of the watermark to withstand modification of the text, and *quality*, how much the watermark alters the distribution of text generated by the model. While it might seem that watermarks must inherently alter the text, surprisingly, a very strong quality notion is possible. A watermark is *undetectable* if no polynomial-time algorithm can distinguish between the watermarked model and the original model, even if it is allowed to make adaptive queries. This implies, in particular, that the watermark does not affect performance on any experiment that terminates within our lifetime. Christ et al. (2024) introduced the first undetectable watermark for LLMs, which requires a modest amount of text to detect but lacks robustness: changing even 1% of the text removes the watermark.

Subsequent work (Christ & Gunn, 2024) proposed the first robust and undetectable LLM watermark, which remains even if *any constant fraction* of the tokens are changed. However, while this scheme's asymptotic guarantees are favorable in theory, it is folklore that the concrete parameters are poor



Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

when implemented exactly as proposed. In particular, the *number of tokens* required to detect the watermark is in the thousands, even with no robustness.

The watermarks adopted by Google and Anthropic, which use the SynthID watermark family (Dathathri et al., 2024), have far weaker quality guarantees. While neither has stated which version of SynthID they deployed, the highest-quality version in the corresponding research paper (Dathathri et al., 2024) achieves only *distortion-freeness*, which says that a *single* watermarked text is distributed identically to a single response from the original model. However, distortion-freeness says nothing about whether the watermark introduces correlations between multiple responses, and indeed it has been observed that distortion-free watermarks can worsen misalignment (Verma et al., 2026).

Therefore, an undetectable and robust LLM watermark with practical concrete parameters (in particular, detection from $\sim$1,000 tokens) would be a significant improvement. The Christ-Gunn approach has already been made practical for *image* watermarks, by Gunn et al. (2025).

We initiate the study of *practical* robust and undetectable watermarks for LLMs, following the approach of Christ & Gunn (2024). Roughly, their watermark works by planting a codeword from a pseudorandom error-correcting code in the generated text, where each token encodes a few bits of the codeword. The detector converts the text to a binary string and uses a secret key to check whether it is close to a codeword. Low-entropy tokens translate to errors imposed on the detector's string: the watermark has little freedom to embed the desired codeword bits without altering the distribution too much. The codeword length determines the number of tokens required to detect the watermark, and unfortunately the error-correcting codes of (Christ & Gunn, 2024) have a poor tradeoff between error tolerance and codeword length.

Our contributions:

**Entropy-aware detection.** Our key insight is that the detector can infer likely error locations by estimating the entropy of each token in the received text. This extra information is crucial: for the error-correcting codes used by Christ & Gunn (2024), our new detector achieves the same error tolerance with shorter codewords, decreasing the amount of text required for detection. We show empirically using the Qwen family of models that *as few as $\sim$450 tokens* suffice for detection at 90% TPR. Compared to the detector of Christ & Gunn (2024), we achieve 90.0% TPR versus 58.2% for our baseline set of parameters at 448 tokens, with both targeting a 0.1% FPR (Figure 1a). For the first time, this gives us an undetectable watermark robust to a constant fraction of token changes and detectable given a few paragraphs.

**Cross-model detectability.** Estimating the entropy requires some guesswork, as the detector does not know the model version used to generate the text nor the prompt. Even so, we show that other model versions, without knowledge of the prompt, still significantly improve detection. Performance is best when the entropy estimator is the same as the generating model. Meanwhile, we find that using a smaller model to estimate entropy retains most of the detection power, with an average drop of roughly 5 percentage points at lengths where matched-model detection reaches 90% TPR (Figure 2).

**Undetectability, quality, and diversity.** Consistent with the undetectability guarantees from its pseudorandom-code construction, our empirical evaluations show similar benchmark performance and output diversity to unwatermarked generation. Moreover, our watermark produces more diverse responses and better resists black-box detection than selected state-of-the-art baselines. Compared with TextSeal (Sander et al., 2026), our watermark produces more diverse responses, with Self-BLEU of 0.019 versus 0.047 at 1,024 tokens on Qwen3-8B-Base (Table 12). Neither of the black-box detection tests of Gloaguen et al. (2024) detects our watermark, while each of the three baseline watermarks, including SynthID-Text (Dathathri et al., 2024), is detected by at least one test (Table 13).

**Robustness.** Furthermore, our watermark is highly resistant to watermark-stealing attacks (Jovanović et al., 2024), with attack success rates of 0.2% versus 44.2% for SynthID-Text (Dathathri et al., 2024) under the strongest tested variants after 30,000 watermarked queries (Table 14). Our watermark also resists removal through random token replacement, retaining 94.6% detection after replacing 30% of the tokens in 4,096-token responses (Table 15a).

**False-positive guarantees and implementation.** In addition to our experiments, we prove an upper bound on the FPR of our new detector and observe false-positive rates consistent with this bound. We provide the first practical implementation of PRC watermarking for language models, available at `https://anonymous.4open.science/r/Pseudorandom-Code-Watermark-LLM/`.

## 2 RELATED WORKS

Following two influential works of Kirchenbauer et al. (2023) and Aaronson (2023), there has been a steadily growing body of literature on LLM watermarks.

Kirchenbauer et al. (2023); Aaronson (2023) take a similar approach, using pseudorandomness derived from the previous $k$ tokens to influence the choice of the next token. Detection requires rederiving this pseudorandomness and checking whether the next token matches; therefore, it relies on $k$-grams being preserved, limiting robustness. For example, if every other token is changed, all $k$-grams from the original response will be destroyed, removing the watermark. This approach also alters the quality of the text, as repeated $k$-grams will result in the same pseudorandomness, biasing their following tokens the same way each time.

Other schemes offer different combinations of quality and robustness. For example, Kuditipudi et al. (2023) construct a watermark that is *distortion-free*, in that a single watermarked response is distributed identically to a single response from the original model. Their scheme is also robust to a constant fraction of edits. However, their scheme does introduce correlated biases *across* responses.

The strongest notion of quality is called *undetectability* (Christ et al., 2024), and ensures that any polynomial number of watermarked responses are computationally indistinguishable from the original model. Christ et al. (2024) construct an undetectable watermarking scheme that, like Kirchenbauer et al. (2023); Aaronson (2023), use previous tokens to derive pseudorandomness biasing the next token. The key difference is that the number of previous tokens is chosen variably depending on their entropy, ensuring that the pseudorandomness is never reused. However, the scheme of Christ et al. (2024) is not robust if enough $k$-grams are altered.

A subsequent scheme of Christ & Gunn (2024) used a new tool called *pseudorandom codes* to achieve both undetectability and robustness to any constant fraction of token substitutions. However, while the scheme had favorable asymptotics, it was folklore knowledge that naive implementations resulted in an infeasible amount of text required for detection, on the order of thousands of tokens. A recent work (Wang et al., 2025) indeed found that naively implementing the scheme from the original PRC watermarking paper (Christ & Gunn, 2024) is *not* practical, requiring severely quality-degrading high temperatures for detection. In contrast, in this work, we'll show that the PRC watermark framework *can* in fact be made feasible in practice by introducing entropy-aware detection. Therefore, before this work, there was no practical undetectable and robust watermarking scheme. Deployed schemes have relatively weaker quality guarantees. These include SynthID, which is currently used to watermark Gemini (Dathathri et al., 2024), and Meta's TextSeal (Sander et al., 2026). Both are distortion-free but not undetectable. See Table 1 for a comparison of these schemes.

There are also a number of schemes prioritizing robustness to semantic-preserving transformations, such as paraphrasing (Ren et al., 2024; Liu et al., 2024). In contrast, we focus on watermarks with milder robustness and far stronger quality guarantees, namely undetectability.

Table 1: Comparison of language model watermarking schemes.

| Scheme | Undetectable | Distortion-free (single response) | Nondeterministic outputs |
|---|---|---|---|
| KGW (Kirchenbauer et al., 2023) | ✗ | ✗ | ✓ |
| Gumbel (Aaronson, 2023) | ✗ | ✓* | ✗† |
| SynthID-Text (Dathathri et al., 2024) | ✗ | ✓‡ | ✓ |
| TextSeal (Sander et al., 2026) | ✗ | ✓¶ | ✓ |
| PRC (ours) | ✓§ | ✓ | ✓ |

*Provided that no $k$-gram is repeated. †With a fixed key, each prompt yields one deterministic response; context-window hashing restores variability at the cost of cross-response correlations. ‡In the non-distortionary configuration. ¶After both TextSeal keys have been used for a repeated context, subsequent occurrences use unwatermarked sampling. §Computational undetectability guarantee from Christ & Gunn (2024). See Appendix B.5 for discussion on undetectability in our practical implementation.

# 3 METHODS

We now describe our watermarking scheme based on pseudorandom error-correcting codes (PRCs).

A pseudorandom error-correcting code is a keyed error-correcting code whose encodings are computationally indistinguishable from uniform strings to an efficient observer without the key, while the key holder can recognize codewords despite noise. We use the zero-bit PRC construction introduced in Christ & Gunn (2024), which uses the Learning Parity with Noise (LPN) assumption for pseudorandomness and sparse parity checks for robustness. "Zero-bit" means the PRC only encodes the presence of a codeword and thus the presence of a watermark.

## 3.1 OVERVIEW OF THE PRC WATERMARKING SCHEME

At a high level, our watermarking scheme consists of the following three stages:

1. **Generate a PRC sequence.** A keyed PRC encoder produces a hidden binary sequence.
2. **Watermark the model by embedding the PRC sequence during generation.** Map the tokens vocabulary to two bins. Each hidden PRC bit influences which bin at that corresponding position is selected, while preserving the original model distribution.
3. **Detect the watermark in a candidate text.** The detector maps observed tokens to bins, assigns soft scores for the hidden bits, and combines these scores using the secret parity checks. It outputs watermarked if the combined score exceeds a specified threshold.

A key challenge in watermark detection is that the strength of the token-level watermark signal depends on the probability distribution at each position. Our entropy-aware detectors account for these variations in token probabilities to improve detection.

## 3.2 EMBEDDING THE WATERMARK

We first describe how to embed a PRC sequence to watermark a model. Let $x = (x_1, \ldots, x_T) \in \{0,1\}^T$ be the binary sequence to embed into the generated tokens $y = (y_1, \ldots, y_T)$. We partition the vocabulary into two bins, $V_0$ and $V_1$, and define $b_i = \textsc{Text-to-Bin}(y_i)$, where $\textsc{Text-to-Bin}(v) = b$ for $v \in V_b$. Let $\pi_i$ be the model's next-token distribution conditioned on the prompt and preceding tokens, and let $p_i = \sum_{v \in V_1} \pi_i(v)$.

The embedding rule of Christ & Gunn (2024) uses both $x_i$ and $p_i$ to randomly select a bin $b_i$, then samples $y_i$ from $\pi_i$ conditioned on that bin. The full procedure is given in Appendix B.1. This procedure has the following two properties which are useful for watermarking.

**Lemma 3.1** (Distribution preservation (Christ & Gunn, 2024))**.** *If the input bits $x_1, \ldots, x_T$ are independent and uniform, the embedding procedure produces exactly the same text distribution as ordinary sampling from the language model.*

The proof is given in Appendix A.1.

**Lemma 3.2** (Per-token watermark signal (Christ & Gunn, 2024))**.** *For independent uniform input bits, at each position $i$,*

$$\Pr[b_i = x_i \mid p_i] = \frac{1}{2} + \min(p_i, 1 - p_i). \tag{1}$$

The proof is given in Appendix A.3. This shows that positions with more balanced bin probabilities (higher entropy) carry stronger watermark signal. This motivates entropy-aware detection.

To watermark the output, we replace the independent uniform input bits with PRC codewords, described next.

## 3.3 ZERO-BIT PRC CONSTRUCTION

We use a zero-bit PRC, from Christ & Gunn (2024), which encodes only the presence of a watermark. Let $n$ be the block length, $r$ the number of parity checks, $t$ the number of coordinates in each check, and $\eta$ the noise rate.

The secret key fixes a full-row-rank parity-check matrix $P \in \mathbb{F}_2^{r \times n}$, with $t$ ones in each row, and a uniformly random pad $z \in \{0,1\}^n$ sampled independently of $P$. For each encoding, the encoder samples a clean codeword $c \in \{0,1\}^n$ satisfying $Pc = 0$ and independent noise bits $e_i \sim \mathrm{Ber}(\eta)$, then outputs $x = c \oplus z \oplus e$, where $\oplus$ denotes bitwise addition modulo two. The matrix $P$ and pad $z$ remain fixed across encodings.

Appendix B.2 gives the full construction of a zero-bit PRC with a fixed block-length. Online ("sampling on the fly") and multi-block extensions are described in Appendix B.3 and B.4, respectively.

### 3.4 UNDETECTABILITY

We show that our watermark scheme using the zero-bit PRC satisfies the computational undetectability guarantees in Christ & Gunn (2024, Lemma 19).

**Corollary 3.2.1** (Undetectability)**.** *Let $\lambda$ be the security parameter of the PRC construction, with key and code sizes chosen as functions of $\lambda$. Consider any polynomial-time adversary $A$ without access to the watermarking key or detector. Assume the PRC encoder satisfies the PRC pseudorandomness requirement of Christ & Gunn (2024, Definition 1).*

*For an efficiently samplable language model, write* $\mathsf{WM}$ *for our embedding procedure (Section 3.2), using fresh PRC encodings under one key, and* $\mathsf{Plain}$ *for ordinary model generation. Define $p_{\mathrm{wm}} = \Pr[A^{\mathsf{WM}} = 1]$ and $p_{\mathrm{plain}} = \Pr[A^{\mathsf{Plain}} = 1]$. These are the probabilities that the adversary outputs 1 after making polynomially many adaptive queries to the watermarked and ordinary models, respectively.*

*Then, for every constant $c > 0$ and all sufficiently large $\lambda$,*

$$|p_{\mathrm{wm}} - p_{\mathrm{plain}}| \leq \lambda^{-c}. \tag{2}$$

In other words, assuming PRC pseudorandomness and efficient model sampling, no polynomial-time adversary without the key or detector can distinguish watermarked from ordinary model responses with non-negligible advantage.

See Appendix A.2 for the proof. The relation between the undetectability guarantee of Corollary 3.2.1 and our experimental settings is discussed in Appendix B.5.

### 3.5 ENTROPY-AWARE PRC WATERMARK DETECTION

We now describe how to detect the PRC watermark in a generated text. A key contribution of our work is introducing entropy-aware detection for PRC watermarks.

We first describe detection for one complete PRC block. Given a candidate token sequence $y = (y_1, \ldots, y_T)$, the detector recovers the observed bins $b_i = \textsc{Text-to-Bin}(y_i)$. For entropy-aware detection, it also estimates the bin probabilities. We retain $p_i$ for the true probability mass of $V_1$ under the generation model and write $\tilde{p}_i$ for the detector's estimate. The detector obtains $\tilde{p}_i$ by evaluating a detector model on the observed token prefix $y_1, \ldots, y_{i-1}$ and summing its next-token probabilities over $V_1$. This requires a model forward pass but does not require the original prompt. The detector model may differ from the generation model.

We define a general soft-parity detector, followed by the three token-scoring rules that we evaluate.

#### 3.5.1 SOFT TOKEN AND PARITY SCORES

**Definition 3.1** (Soft token score)**.** For each position $i$, assign a soft token score $s_i \in [-1, 1]$ representing evidence about the hidden PRC bit $x_i$. We use the convention that $s_i > 0$ is evidence that $x_i = 0$, $s_i < 0$ is evidence that $x_i = 1$, and the magnitude $|s_i|$ represents the strength of this evidence. In particular, $s_i = 0$ means that the observation provides no net evidence for either value of $x_i$.

**Definition 3.2** (Soft parity score)**.** Let $\mathcal{S}_j = \{i : P_{j,i} = 1\}$ denote the support of parity check $j$. Each parity check has weight $|\mathcal{S}_j| = t$. Define its soft parity score as $q_j = \prod_{i \in \mathcal{S}_j} s_i$.

### 3.5.2 SOFT-PARITY PRC DETECTOR

**Definition 3.3** (Soft-parity PRC watermark detector)**.** Given observed bins $b$, estimated probabilities $\tilde{p}$, a token-scoring rule, parity-check matrix $P$, secret pad $z$, and target false-positive rate $\alpha$, the detector proceeds as follows.

1. Compute the soft token score $s_i$ for each token.
2. For each parity check $j$, compute the soft parity score $q_j$ from Definition 3.2 and correct its sign using the secret pad: $X_j = \prod_{i \in \mathcal{S}_j} ((-1)^{z_i} s_i) = (-1)^{\sum_{i \in \mathcal{S}_j} z_i} q_j$.
3. Sum over all $r$ parity checks to obtain $D = \sum_{j=1}^{r} X_j$ and $\nu = \sum_{j=1}^{r} q_j^2$.
4. If $\nu = 0$, output *not watermarked*. Otherwise, output *watermarked* if $D \geq \sqrt{2\nu \log(1/\alpha)}$.

### 3.5.3 NAIVE DETECTOR

**Definition 3.4** (Naive PRC watermark detector)**.** The naive PRC watermark detector uses the score $s_i^{\text{naive}} = 1 - 2b_i$. Thus, if $b_i = 0$, then $s_i^{\text{naive}} = +1$, and if $b_i = 1$, then $s_i^{\text{naive}} = -1$.

This detector treats the observed bin as its estimate of the hidden bit. It does not account for the model probability $p_i$ at each position.

**The problem with low-entropy tokens.** When $p_i$ is close to 0 or 1, the observed bin is only weakly correlated with the hidden bit. Low-entropy positions therefore carry little watermark signal. This motivates using the model probabilities when assigning token scores.

### 3.5.4 ENTROPY-WEIGHTED DETECTOR

**Definition 3.5** (Entropy-weighted detector)**.** Let $H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$ denote binary entropy. The entropy-weighted detector uses $s_i^{\text{ent}} = (1 - 2b_i) H(\tilde{p}_i)$.

This detector assigns the same sign as the naive detector and a magnitude equal to the binary entropy. It therefore gives smaller weight to low-entropy positions.

However, the two observed bins can provide different amounts of evidence about $x_i$. For example, when $p_i = 0.1$, the embedding rule in Appendix B.1 gives $\Pr[b_i = 1 \mid x_i = 0, p_i = 0.1] = 0$ and $\Pr[b_i = 1 \mid x_i = 1, p_i = 0.1] = 0.2$. Thus, observing $b_i = 1$ determines that $x_i = 1$, whereas observing $b_i = 0$ gives weaker evidence for $x_i = 0$. The entropy-weighted detector assigns these two observations the same magnitude, motivating a score derived directly from the posterior distribution of the hidden bit.

### 3.5.5 POSTERIOR-MEAN DETECTOR

**Definition 3.6** (Posterior-mean detector)**.** Under the embedding model with a uniform prior on the input bit $x_i$, we define the posterior-mean score as

$$s_i^{\text{post}} = \Pr[x_i = 0 \mid b_i, p_i] - \Pr[x_i = 1 \mid b_i, p_i] = \mathbb{E}[(-1)^{x_i} \mid b_i, p_i]. \tag{3}$$

If the observed bit is $b_i = 0$, then $s_i^{\text{post}} = p_i/(1-p_i)$ when $p_i \leq 1/2$, and $s_i^{\text{post}} = 1$ when $p_i > 1/2$. If the observed bit is $b_i = 1$, then $s_i^{\text{post}} = -1$ when $p_i \leq 1/2$, and $s_i^{\text{post}} = -(1-p_i)/p_i$ when $p_i > 1/2$.

Thus, an observation from the lower-probability bin is assigned score magnitude 1, while an observation from the higher-probability bin is assigned score magnitude $\min(p_i, 1-p_i)/\max(p_i, 1-p_i)$. When the detector uses an estimated probability $\tilde{p}_i$, we apply the same scoring rule with $\tilde{p}_i$ in place of $p_i$. The derivation is given in Appendix A.4.

## 3.6 STATISTICAL FALSE-POSITIVE RATE GUARANTEE FOR PRC WATERMARK DETECTION

The following theorem provides a statistical upper bound on the false-positive rate of our PRC watermark detector. It applies to all three scoring methods.

**Theorem 3.3** (False-positive rate guarantee for PRC watermark detection)**.** *Let $P \in \mathbb{F}_2^{r\times n}$ be a full-row-rank parity-check matrix and let $z \leftarrow$ Unif$(\{0,1\}^n)$ be the secret pad. Consider any unwatermarked candidate text whose resulting token scores $s_1, \ldots, s_n$ are independent of $z$. Let $q_j = \prod_{i\in\mathcal{S}_j} s_i$ and let $D$ be the detector statistic from Definition 3.3. Assume $\sum_{j=1}^r q_j^2 > 0$. Then, for any target false-positive rate $\alpha \in (0,1)$, $\Pr_z\left[D \geq \sqrt{2\log(1/\alpha)\sum_{j=1}^r q_j^2}\right] \leq \alpha$.*

See Appendix A.5 for the proof.

Note that, for the entropy-aware detectors which require estimates of the bin probabilities (or entropies) at each position, errors in these estimates can reduce TPR, but do not affect the FPR guarantee, which holds for any token scores independent of the secret pad.

## 4 Experimental results

Following the approach of Kirchenbauer et al. (2023) and Kuditipudi et al. (2023), we evaluate our watermark on continuations from the RealNewsLike subset of C4 Raffel et al. (2020). For our primary detection experiments, we randomly sample 500 prompts and generate completions using Qwen3 models Yang et al. (2025) with the implementation provided by Raschka (2024). Since these are text completions for given prefixes, we disable the chat template. We set the temperature to 1 unless otherwise stated.

We implement the PRC watermark described in Section 3.1. For detection, we use the analytical threshold from Theorem 3.3 with a target FPR $\alpha = 10^{-3}$. We evaluate detection, as measured by TPR@FPR, across different detector scoring methods, PRC noise rates $\eta$, parity-check weights $t$, fixed-block vs. online PRC watermark constructions, model sizes, and detector models. We also directly compare our entropy-aware detector to the naive implementation of the original Christ-Gunn watermark detector from (Wang et al., 2025; Christ & Gunn, 2024).

Additionally, we compare our watermark with the state-of-the-art watermarking methods TextSeal (Sander et al., 2026) and SynthID-Text (Dathathri et al., 2024), alongside Gumbel-Max as a baseline (Aaronson, 2023). We evaluate detection, repetition within individual responses, and diversity across responses to the same prompt. We also evaluate undetectability via black-box detection tests and robustness to watermark-stealing attacks and random token substitutions.

In the sections below, we present the main observations from our experiments. Additional experimental details are provided in Appendix D.

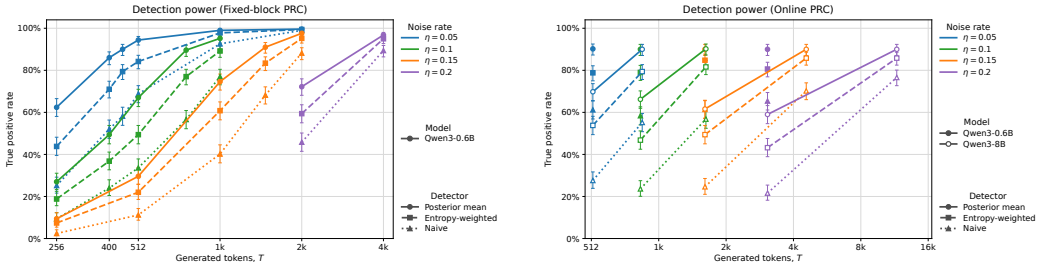
### 4.1 Detectability

**Entropy-aware detection substantially improves detection power.** Across all evaluated noise rates and token lengths, and for both fixed-block and online PRC, entropy-aware detection improves upon the naive detector — in particular, the posterior-mean detector achieves the highest TPR, followed by the entropy-weighted detector, then naive detector in all settings (Figure 1). For fixed-block PRC with Qwen3-0.6B-Base, $\eta = 0.05$, and $T = n = 448$, the three detectors achieve $90.0\%$, $79.4\%$, and $58.2\%$ TPR, respectively, at target FPR $10^{-3}$. Thus, accounting for token-level entropy improves detection even without increasing response length.

**Sparse parity checks and low noise rates favor shorter detection lengths.** For fixed-block PRC with $t = 3$, the shortest evaluated lengths reaching at least $90\%$ TPR are 448 tokens at $\eta = 0.05$ and 4,096 tokens at $\eta = 0.20$, both using the posterior-mean detector (Table 3). Meanwhile, increasing the parity check weight from $t = 3$ to $t = 5$ drops TPR from $97.2\%$ to $28.0\%$ at $n = 400$ and $\eta = 0.05$, for the fixed-block PRC with sequence length $T = 2n$ (Appendix D.4). Thus, we use $t = 3$ in our main experiments as it is the most practical setting.[1]

**Entropy-aware detection also significantly improves detection on Wang et al. (2025).** On identical watermarked responses, our posterior-mean detector achieves better detection than Wang et al.'s implementation of the Christ–Gunn detector Wang et al. (2025); Christ & Gunn (2024) across all evaluated settings (or identical detection, once both reach 100% TPR). For instance, at temperature

[1] See Appendix B.5 for a discussion on the implications for undetectability.




(a) Fixed-block PRC with Qwen3-0.6B.

(b) Online PRC with Qwen3-0.6B and Qwen3-8B.

Figure 1: **Detection power across detectors, response lengths, and PRC noise rates.** (1) Posterior-mean detection achieves the highest observed TPR across both constructions and all noise rates. (2) Higher noise rates require longer responses. For online PRC, Qwen3-8B needs approximately 1.7–3.8 times as many tokens as Qwen3-0.6B to reach 90% observed posterior-mean TPR at the evaluated lengths. The horizontal axis uses a logarithmic scale. Error bars show pointwise 95% Wilson confidence intervals from 500 responses per point, conditional on the evaluated keys and settings. All settings use temperature 1, $t = 3$, and target FPR 0.1%.

1.2, our posterior-mean detector obtains 97.50% TPR compared to 0% with their baseline detector, with an observed FPR of 0.0049%. See Table 11 and Appendix D.9 for the full comparison.

**Observed false-positive rates are consistent with the Hoeffding bound.** Observed false positives are zero in nearly every configuration at target FPR $\alpha = 10^{-3}$, and a sweep over larger target FPRs $\alpha \in \{10^{-2}, 5{\cdot}10^{-2}, 10^{-1}\}$ shows observed FPR almost always falls below the target. Full results are given in Table 8.

**Larger models require longer responses.** Holding the parameters of the PRC watermark fixed, larger models require more tokens to reach the same detection power. Qwen3-8B-Base requires approximately 1.7–3.8× as many tokens as Qwen3-0.6B-Base to reach at least 90% observed TPR at the evaluated lengths, with the posterior detector and online PRC (Figure 1b). The respective lengths are 864 versus 520 tokens at $\eta = 0.05$, and 11,856 versus 3,136 tokens at $\eta = 0.20$ (Table 5). This pattern is consistent with the lower token and binary-partition entropy measured for the larger model (Table 6), which leaves less sampling uncertainty through which to embed the watermark.

**Watermarks remain detectable with a smaller model.** For responses generated by Qwen3-8B-Base, using a smaller model, Qwen3-0.6B-Base, to recover token probabilities retains much of the detection power. At practically-relevant response lengths where posterior-mean detection by the matched 8B model achieves ∼90% TPR, the smaller 0.6B model achieves 83.2–86.0% TPR. Longer responses generally reduce this gap, although matched-model detection remains more effective across all evaluated settings. Full results are provided in Appendix D.2.

### 4.2 Quality and Undetectability

Table 2: Quality benchmark scores for watermarked and unwatermarked Qwen3-0.6B generation with the chat template and temperature 1. The gap (unwatermarked − watermarked, computed before rounding) is near zero ($< 0.01$) on all benchmarks, consistent with the undetectability guarantee of the PRC watermark.

| Benchmark | Watermarked | Unwatermarked | Δ |
|---|---|---|---|
| ARC-Easy | 0.837 | 0.827 | −0.009 |
| GSM8K | 0.748 | 0.753 | 0.005 |
| HellaSwag | 0.404 | 0.406 | 0.002 |
| MMLU | 0.514 | 0.515 | 0.001 |
| IFEval | 0.559 | 0.567 | 0.007 |

Undetectability implies that watermarking should not degrade (or even improve) the quality of outputs. We evaluate whether our PRC watermark affects model quality by comparing watermarked and unwatermarked Qwen3-0.6B outputs on public benchmarks, including ARC-Easy (Clark et al., 2018), GSM8K (Cobbe et al., 2021), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021), and IFEval (Zhou et al., 2023), which cover reasoning, knowledge, and instruction following. Our evaluations use the chat template and temperature 1.

**Our PRC watermark preserves the benchmark performance of the original model.** Across these five benchmarks, the absolute score difference is at most 0.93 percentage points (Table 2). These small observed differences between the quality of watermarked and unwatermarked generations support the practical utility and undetectability of our PRC watermark.

**Our PRC watermark preserves the diversity of unwatermarked text.** Responses sampled with our watermark show no statistically significant differences from unwatermarked responses on the three diversity metrics across the evaluated response lengths. These include Self-BLEU for diversity between responses to the same prompt, and repeated 4-grams and Distinct-3 for repetition within individual responses (Table 12).

**Our PRC watermark yields greater between-response diversity than selected SOTA baselines.** Responses sampled with our PRC watermark have significantly lower Self-BLEU (higher between-sequence diversity) than those sampled with TextSeal, SynthID-Text (depth 30), and Gumbel-Max at evaluated response lengths, even when repeat handling is enabled for these baselines (Appendix D.10).

**Black-box tests of undetectability flag all three baselines but do not flag our watermark.** We apply the Red-Green and Fixed-Sampling tests of Gloaguen et al. (2024) to evaluate whether generated responses reveal watermarking *without* access to the secret key or detector. On Qwen3-0.6B, using 18,000 responses per watermark scheme for Red-Green and 1,000 for Fixed-Sampling, neither test flags the PRC watermark or unwatermarked generation. Meanwhile, KGW-2.0 and SynthID-Text are flagged by the Red-Green test, and EXP by the Fixed-Sampling test (Appendix D.11).

### 4.3 Robustness

**Our PRC watermark is harder to spoof while retaining useful detection.** These attacks learn token preferences from watermarked responses and try to produce new text that passes watermark detection without the secret key. We evaluate adaptations of Jovanović et al. (2024) on Qwen3-0.6B-Base, using one watermarking key per scheme. At 30,000 watermarked queries, KGW-2.0 and EXP are easily spoofed, with success rates of 100% and 98.4%, respectively, while SynthID-Text reaches 44.2% (Table 14). Our watermark, on the other hand, is highly resistant in the tested settings: 15 of 17,500 attack texts (0.086%) trigger detection and pass the quality filter. This uses an empirically calibrated threshold targeting a 0.1% FPR, which detects 91.0% of genuine watermarked query responses. The analytical threshold instead flags one attack text and detects 83.8% of the same genuine responses. See details in Appendix D.12.

**Random token substitutions reduce detection power.** We evaluate uniformly random token substitutions on Qwen3-0.6B-Base, following Kuditipudi et al. (2023). For 4096-token responses, our watermark retains very high 94.6% detection even when 30% of tokens are substituted. At the same length and substitution rate, EXP and KGW-2.0 retain 100% detection, while SynthID-Text retains 99.0%. All schemes use thresholds targeting a nominal 0.1% FPR. For shorter lengths, however, PRC detection falls severely with higher substitution rates. See Appendix D.13.

## 5 Conclusion

We present the first practical implementation and empirical study of pseudorandom code (PRC)-based undetectable watermarking for language models. We show that entropy-aware detection substantially reduces the text required to detect PRC watermarks, making it practical for outputs as short as $\sim$450 tokens. Consistent with the undetectability guarantees from its construction, the PRC watermark preserves diversity of outputs and performance on LLM benchmarks, and is resistant to black-box detection. Our watermark is also resistant to stealing and removal attacks in evaluated settings. Thus, this work opens a new practical route to undetectable and robust watermarks for language models using pseudorandom codes.

### ACKNOWLEDGMENTS

We thank the Modal for Academics program for providing us the compute credits for running the experiments in this study.

### AI USE STATEMENT

We used generative AI tools for coding the implementation and analyses, as well as for refining proofs, finding related literature, and drafting and polishing the manuscript. The central idea of applying entropy-aware detection to PRC watermarks was developed by us. We have carefully reviewed and revised the AI-assisted material, and we take responsibility for the final content.

### ETHICS STATEMENT

This work aims to support transparency and accountability by making PRC watermark detection more practical. Our attack evaluations help assess vulnerabilities and inform future defenses. We acknowledge the risks of false detections and misuse of attack methods, and recommend interpreting detection results alongside other evidence, particularly when informing consequential decisions.

### REPRODUCIBILITY STATEMENT

We provide details of our PRC watermark scheme in Appendix B, proofs in Appendix A, and experimental settings in Appendix C. Our implementation is available at `https://anonymous.4open.science/r/Pseudorandom-Code-Watermark-LLM/`.

## REFERENCES

Scott Aaronson. Watermarking of large language models. Talk at the Simons Institute for the Theory of Computing, Large Language Models and Transformers Workshop, August 2023. URL https://simons.berkeley.edu/talks/scott-aaronson-ut-austin-openai-2023-08-17. August 17, 2023.

Antje Barth. Amazon Titan Image Generator and watermark detection API are now available in Amazon Bedrock. https://aws.amazon.com/blogs/aws/amazon-titan-image-generator-and-watermark-detection-api-are-now-available-in-amazon-bedrock/, 2025.

Miranda Christ and Sam Gunn. Pseudorandom Error-Correcting Codes. In *Advances in Cryptology – CRYPTO 2024: 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2024, Proceedings, Part VI*, pp. 325–347, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-68390-9. doi: 10.1007/978-3-031-68391-6_10. URL https://doi.org/10.1007/978-3-031-68391-6_10. event-place: Santa Barbara, CA, USA.

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable Watermarks for Language Models. In *Proceedings of Machine Learning Research*, volume 196, 2024. URL https://proceedings.mlr.press/v247/christ24a/christ24a.pdf.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Claude Support. How claude marks ai-generated content. https://support.claude.com/en/articles/16266773-how-claude-marks-ai-generated-content, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi

Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, Ilia Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, Oct 2024. ISSN 1476-4687. doi: 10.1038/s41586-024-08025-4. URL https://doi.org/10.1038/s41586-024-08025-4.

Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Black-box detection of language model watermarks. In *ICML 2024 Workshop on Foundation Models in the Wild*, 2024. URL https://openreview.net/forum?id=SgrEUyZia6.

Google DeepMind. SynthID – A tool to watermark and identify content generated through AI. https://deepmind.google/science/synthid/.

Sam Gunn, Xuandong Zhao, and Dawn Song. An Undetectable Watermark for Generative Image Models. 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1331202ea3bb0a53ae897af0bb16e309-Paper-Conference.pdf.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.03300.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

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 (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, July 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *CoRR*, abs/2307.15593, 2023. doi: 10.48550/ARXIV.2307.15593. URL https://doi.org/10.48550/arXiv.2307.15593.

Laurie Richardson. How we're increasing transparency for gen AI content with the C2PA. https://blog.google/technology/ai/google-gen-ai-content-transparency-c2pa/, 2024.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 6499–6519, 2024.

Nick Clegg. Labeling AI-Generated Images on Facebook, Instagram and Threads. https://about.fb.com/news/2024/02/labeling-ai-generated-images-on-facebook-instagram-and-threads/, 2024.

OpenAI. Understanding the source of what we see and hear online. https://openai.com/index/understanding-the-source-of-what-we-see-and-hear-online/, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Sebastian Raschka. *Build A Large Language Model (From Scratch)*. Manning, 2024. ISBN 978-1633437166. URL https://www.manning.com/books/build-a-large-language-model-from-scratch.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. A robust semantics-based watermark for large language model against paraphrasing. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 613–625, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.40. URL https://aclanthology.org/2024.findings-naacl.40/.

Tom Sander, Hongyan Chang, Tomáš Souček, Tuan Tran, Valeriu Lacatusu, Sylvestre-Alvise Rebuffi, Alexandre Mourachko, Surya Parimi, Christophe Ropers, Rashel Moritz, Vanessa Stark, Hady Elsahar, and Pierre Fernandez. Textseal: A localized llm watermark for provenance & distillation protection. *arXiv preprint arXiv:2605.12456*, 2026.

Apurv Verma, Hai Phan, and Shubhendu Trivedi. Watermarking degrades alignment in language models: Analysis and mitigation. *Trans. Mach. Learn. Res.*, 2026, 2026. URL https://openreview.net/forum?id=w2ATKQcfWx.

Tianrui Wang, Anyu Wang, Tianshuo Cong, Delong Ran, Jinyuan Liu, and Xiaoyun Wang. Cryptanalysis of LDPC-based pseudorandom error-correcting codes. Cryptology ePrint Archive, Paper 2025/2283, 2025. URL https://eprint.iacr.org/2025/2283.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019. URL https://aclanthology.org/P19-1472/.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023. URL https://arxiv.org/abs/2311.07911.

# A ADDITIONAL PROOFS

## A.1 PROOF OF DISTRIBUTION PRESERVATION

*Proof of Lemma 3.1.* Fix a generation position $i$ and condition on the previously generated tokens $y_1, \ldots, y_{i-1}$.

If $p_i \leq 1/2$, then $\Pr[b_i = 1] = \frac{1}{2}(0) + \frac{1}{2}(2p_i) = p_i$. If $p_i > 1/2$, then $\Pr[b_i = 1] = \frac{1}{2}(2p_i - 1) + \frac{1}{2}(1) = p_i$. Thus the probability of selecting each bin is exactly its probability mass under the original model distribution.

For any $v \in V_1$,

$$\Pr[y_i = v \mid y_1, \ldots, y_{i-1}] = p_i \frac{\pi_i(v)}{p_i} = \pi_i(v). \tag{4}$$

The same argument holds for $v \in V_0$.

Thus, conditioned on every possible prefix $y_1, \ldots, y_{i-1}$, the distribution of the next token is the same as under MODEL. Therefore the joint distribution of the entire generated response $y$ is identical to ordinary sampling from MODEL. □

## A.2 PROOF OF UNDETECTABILITY

*Proof of Corollary 3.2.1.* We follow the argument of Christ & Gunn (2024, Lemma 19). All probabilities are averaged over key generation and the random choices made by the encoder, token sampler, and adversary $A$. Note that the same key is reused across responses, while each encoding uses fresh randomness.

We construct a test for PRC pseudorandomness using $A$. The test gets bitstrings from an oracle $\mathcal{O}$, which either returns fresh PRC encodings under a fixed key, or it returns independent uniform strings of the same length. For each prompt from $A$, the test requests fresh strings from its oracle and generates a response using our embedding procedure (Section 3.2), with independent token-sampling randomness. The test returns $A$'s output and runs in polynomial time.

When the supplied strings are fresh PRC encodings under one key, the responses have exactly the same distribution as those produced by WM. When they are independent uniform strings, Lemma 3.1 shows that the responses have exactly the same distribution as those produced by Plain. The unused uniform bits remain independent of the preceding interaction, so this argument also holds when $A$ chooses prompts based on earlier responses.

The resulting test therefore distinguishes PRC encodings from uniform strings with advantage $|p_{\mathrm{wm}} - p_{\mathrm{plain}}|$. By the assumed PRC pseudorandomness (Christ & Gunn, 2024, Definition 1), this advantage is at most $\lambda^{-c}$ for every constant $c > 0$ and all sufficiently large $\lambda$, thus proving Equation 2. □

## A.3 PROOF OF THE PER-TOKEN WATERMARK SIGNAL

*Proof of Lemma 3.2.* Suppose first that $p_i \leq 1/2$. If $x_i = 0$, then $\Pr[b_i = 0 \mid x_i = 0] = 1$. If $x_i = 1$, then $\Pr[b_i = 1 \mid x_i = 1] = 2p_i$. Therefore,

$$\Pr[b_i = x_i] = \frac{1}{2}(1) + \frac{1}{2}(2p_i) = \frac{1}{2} + p_i. \tag{5}$$

Now suppose that $p_i > 1/2$. If $x_i = 1$, then $\Pr[b_i = 1 \mid x_i = 1] = 1$. If $x_i = 0$, then $\Pr[b_i = 0 \mid x_i = 0] = 2(1 - p_i)$. Therefore,

$$\Pr[b_i = x_i] = \frac{1}{2}(1) + \frac{1}{2}\big(2(1 - p_i)\big) = \frac{1}{2} + (1 - p_i). \tag{6}$$

Combining the two cases gives Eq. (1). □

### A.4 Derivation of the posterior-mean detector

Suppose first that the detector knows the true bin probability $p_i = p$. The hidden bit is uniform, $\Pr[x_i = 0] = \Pr[x_i = 1] = 1/2$.

For $p \leq 1/2$, the generation rule gives $\Pr[b_i = 1 \mid x_i = 0] = 0$ and $\Pr[b_i = 1 \mid x_i = 1] = 2p$. Thus, if $b_i = 1$, then $x_i = 1$ with probability one, so $s^{\text{post}}(1, p) = -1$.

If $b_i = 0$, Bayes' rule gives

$$\Pr[x_i = 0 \mid b_i = 0, p] = \frac{1}{2(1-p)} \tag{7}$$

and

$$\Pr[x_i = 1 \mid b_i = 0, p] = \frac{1-2p}{2(1-p)}. \tag{8}$$

Therefore,

$$\Pr[x_i = 0 \mid b_i = 0, p] - \Pr[x_i = 1 \mid b_i = 0, p] = \frac{p}{1-p}. \tag{9}$$

The case $p > 1/2$ follows symmetrically, giving the scoring rules in Definition 3.6. Thus, when $\tilde{p}_i = p_i$, the score is exactly $\mathbb{E}[(-1)^{x_i} \mid b_i, p_i]$.

When the detector uses an estimated probability $\tilde{p}_i$, we evaluate the same scoring function at $\tilde{p}_i$. In this case the score need not equal the true posterior mean; errors in $\tilde{p}_i$ can therefore reduce the true-positive rate.

### A.5 Proof of the false-positive rate guarantee

*Proof of Theorem 3.3.* Fix the candidate text and, for an entropy-aware detector, the estimated bin probabilities $\tilde{p}_1, \ldots, \tilde{p}_n$. This fixes every token score $s_i$ and therefore every soft parity score $q_j = \prod_{i \in \mathcal{S}_j} s_i$.

For each parity check, define

$$X_j = \prod_{i \in \mathcal{S}_j} ((-1)^{z_i} s_i) = q_j (-1)^{(Pz)_j}. \tag{10}$$

Because the candidate text and token scores are independent of the secret pad $z$, conditioning on them does not change the fact that $z \sim \text{Unif}(\{0,1\}^n)$.

Since $P \in \mathbb{F}_2^{r \times n}$ has full row rank, it is a surjective map from $\mathbb{F}_2^n$ to $\mathbb{F}_2^r$. Therefore,

$$Pz \sim \text{Unif}(\{0,1\}^r). \tag{11}$$

In particular, the coordinates $(Pz)_1, \ldots, (Pz)_r$ are independent $\text{Ber}(1/2)$ random variables. Hence the signs $(-1)^{(Pz)_1}, \ldots, (-1)^{(Pz)_r}$ are independent and uniform over $\{-1, +1\}$. Therefore $\mathbb{E}[X_j] = 0$ and $X_j \in [-|q_j|, |q_j|]$.

Applying the one-sided Hoeffding inequality,

$$\Pr\left[\sum_{j=1}^{r} X_j \geq u\right] \leq \exp\left(-\frac{u^2}{2\sum_{j=1}^{r} q_j^2}\right). \tag{12}$$

Recall that $D = \sum_{j=1}^{r} X_j$. Choose $u = \sqrt{2\log(1/\alpha)\sum_{j=1}^{r} q_j^2}$ as threshold, which gives

$$\Pr_z\left[D \geq \sqrt{2\log(1/\alpha)\sum_{j=1}^{r} q_j^2}\right] \leq \alpha. \tag{13}$$

This completes the proof. □

### A.6 PER-CHECK WATERMARK SIGNAL

To isolate how token-level signal and PRC noise affect a parity check, we consider a simplified model. The bin probabilities $p_1, \ldots, p_n$ and detector estimates are fixed in advance, independently of the clean codeword, secret pad, PRC noise, and embedding randomness. The embedding rule uses independent randomness at each position, and each score $s_i$ depends only on the observed bin $b_i$ and the fixed detector estimate.

Let $\mathcal{S}_j$ denote the support of parity check $j$. For a soft token score $s_i$, define

$$\mu_i = \frac{1}{2}\left(\mathbb{E}[s_i \mid x_i = 0, p_i] - \mathbb{E}[s_i \mid x_i = 1, p_i]\right). \tag{14}$$

For the naive score, Lemma 3.2 gives $\mu_i = 2\min(p_i, 1 - p_i)$.

Recall that $x_i = c_i \oplus z_i \oplus e_i$. Averaging over the uniformly random pad bit $z_i$ and the embedding randomness gives

$$\mathbb{E}\left[(-1)^{z_i} s_i \mid c_i, e_i, p_i\right] = (-1)^{c_i \oplus e_i} \mu_i. \tag{15}$$

Under the conditional-independence assumption above, the expectation of the sign-corrected score for parity check $j$ therefore factorizes:

$$\mathbb{E}[X_j \mid c, p] = \prod_{i \in \mathcal{S}_j} \mathbb{E}\left[(-1)^{z_i} s_i \mid c_i, p_i\right] \tag{16}$$

$$= (-1)^{\sum_{i \in \mathcal{S}_j} c_i} (1 - 2\eta)^t \prod_{i \in \mathcal{S}_j} \mu_i. \tag{17}$$

Because the clean codeword satisfies $\bigoplus_{i \in \mathcal{S}_j} c_i = 0$ for every parity check,

$$\mathbb{E}[X_j \mid c, p] = (1 - 2\eta)^t \prod_{i \in \mathcal{S}_j} \mu_i. \tag{18}$$

The expectation averages over the secret pad, PRC noise, and embedding randomness. It characterizes the mean parity-check score. This shows how weak token-level signal compounds across a parity check. If several positions have $p_i$ close to 0 or 1, their small values of $\mu_i$ reduce the signal of the full check. The factor $(1 - 2\eta)^t$ likewise shows how PRC noise compounds across the $t$ positions in a check. Increasing $t$ therefore makes detection more sensitive to both PRC noise and low-signal positions.

## B DETAILS ON PRC WATERMARK CONSTRUCTIONS

### B.1 EMBEDDING A BINARY SEQUENCE INTO THE LANGUAGE-MODEL GENERATION

We first describe how a binary sequence $x$ is embedded into the language-model output.

#### B.1.1 LANGUAGE MODEL AND BINARY VOCABULARY MAPPING

**Definition B.1** (Language model, adapting Christ & Gunn (2024))**.** A MODEL is a deterministic algorithm that takes as input the prompt and the previously generated tokens $y_1, \ldots, y_{i-1}$ and outputs a probability distribution $\pi_i$ over the next token $y_i$. That is, $\pi_i = \text{MODEL}(\text{prompt}, y_1, \ldots, y_{i-1})$. For each vocabulary token $v \in V$, $\pi_i(v)$ denotes the probability assigned to $v$.

**Definition B.2** (Binary vocabulary mapping)**.** Let $V$ be the language-model vocabulary. Fix a partition $V = V_0 \sqcup V_1$, where $V_0$ and $V_1$ have approximately equal size. Define TEXT-TO-BIN : $V \to \{0, 1\}$ by TEXT-TO-BIN$(v) = b$ if $v \in V_b$.

For a token sequence $y = (y_1, \ldots, y_T)$, write TEXT-TO-BIN$(y) = (b_1, \ldots, b_T)$, where $b_i =$ TEXT-TO-BIN$(y_i)$. At generation position $i$, define $p_i = \sum_{v \in V_1} \pi_i(v)$ to be the total probability mass that the model assigns to bin $V_1$.

In our implementation, the partition into the two bins $V_0, V_1$ is chosen once using a seeded random permutation of $V$. Exactly half of the vocabulary tokens are placed in each bin. The bins therefore have equal cardinality, but their probability masses $1 - p_i$ and $p_i$ need not be equal.

### B.1.2 LANGUAGE-MODEL GENERATION CONDITIONED ON THE PRC SEQUENCE

Ordinary language-model generation repeatedly queries MODEL and samples a token from the resulting next-token distribution. To embed the binary sequence $x$, we use the following generation algorithm from Christ & Gunn (2024).

**Definition B.3** (GENERATE algorithm for watermarked model output)**.** The GENERATE algorithm takes as input the prompt and a binary sequence $x = (x_1, \ldots, x_T) \in \{0,1\}^T$. For each $i = 1, \ldots, T$:

1. Query $\pi_i = \text{MODEL}(\text{prompt}, y_1, \ldots, y_{i-1})$ and compute $p_i = \sum_{v \in V_1} \pi_i(v)$.
2. Sample a binary bin $b_i$ according to $x_i$ and $p_i$:
   - If $p_i \leq 1/2$, sample $b_i \leftarrow \text{Ber}(2x_i p_i)$.
   - If $p_i > 1/2$, sample $b_i \leftarrow \text{Ber}\,(1 - 2(1 - x_i)(1 - p_i))$.
3. Sample the actual language-model token from the original model distribution conditioned on the selected bin, $y_i \sim \pi_i(\cdot \mid V_{b_i})$.

The algorithm outputs the generated token sequence $y = (y_1, \ldots, y_T)$.

### B.1.3 DISTRIBUTION PRESERVATION

**Lemma B.1** (Distribution preservation of GENERATE (Christ & Gunn, 2024))**.** *Suppose the input bits are independent and uniform, $x_i \sim$ Ber$(1/2)$. Then* GENERATE *produces exactly the same distribution as ordinary sampling from* MODEL*.*

The proof is given in Appendix A.1. To watermark the response output by GENERATE, we replace the uniformly random sequence $x$ with a sequence generated by a pseudorandom code.

### B.1.4 PER-TOKEN WATERMARK SIGNAL

Although averaging over a uniformly random $x_i$ preserves the model distribution, the observed bin $b_i$ is correlated with the corresponding hidden bit $x_i$.

**Lemma B.2** (Per-token watermark signal (Christ & Gunn, 2024))**.** *Suppose $x_i$ is uniform. Then*

$$\Pr[b_i = x_i] = \frac{1}{2} + \min(p_i, 1 - p_i). \tag{19}$$

*The advantage of $b_i$ over random guessing for predicting $x_i$ is therefore* $\min(p_i, 1 - p_i)$.

The proof is given in Appendix A.3. This shows why the amount of watermark signal varies across tokens. When $p_i$ is close to $1/2$, the observed bin carries substantial information about $x_i$. When $p_i$ is close to 0 or 1, the observed bin carries much less information about $x_i$.

## B.2 CONSTRUCTING A ZERO-BIT PSEUDORANDOM CODE

This expands upon Section 3.3 in the main text. We first describe the fixed-block PRC construction used to generate the binary sequence $x$. The online "sampling-on-the-fly" construction and the multiblock fixed-length construction are found in Appendix B.3 and B.4, respectively.

**Definition B.4** (Fixed-block zero-bit PRC)**.** Fix a PRC block length $n$, number of parity checks $r$, parity-check weight $t$, and PRC noise rate $\eta$. Let $d = n - r$.

The PRC key-generation algorithm constructs a generator matrix $G \in \mathbb{F}_2^{n \times k}$, a parity-check matrix $P \in \mathbb{F}_2^{r \times n}$, and a uniformly random secret pad $z \in \{0,1\}^n$. The dimension $k$ is internal to the zero-bit PRC encoder.

1. Sample the first $d$ rows $G_1, \ldots, G_d$ uniformly from $\mathbb{F}_2^k$.
2. For each remaining index $j = d+1, \ldots, n$, sample $t-1$ distinct parent indices $i_1, \ldots, i_{t-1} < j$ and set $G_j = G_{i_1} \oplus \cdots \oplus G_{i_{t-1}}$. This introduces a weight-$t$ parity check with support $\mathcal{S}_j = \{i_1, \ldots, i_{t-1}, j\}$.

3. Record the resulting $r$ parity relations as the rows of $P$. Thus every clean codeword $c = Gu$ satisfies $Pc = 0$. Before the final permutation, each parity-check row contains a distinct newest coordinate $j$, so $\mathrm{rank}(P) = r$.

4. Sample a uniform random permutation of the $n$ coordinates and apply the same permutation to the rows of $G$ and the columns of $P$.

5. Sample a uniformly random secret pad $z \leftarrow \mathrm{Unif}(\{0,1\}^n)$.

To sample a PRC sequence, sample $u \leftarrow \mathrm{Unif}(\mathbb{F}_2^k)$ and compute the clean codeword $c = Gu$. Then sample independent noise bits $e_i \sim \mathrm{Ber}(\eta)$ and output $x_i = c_i \oplus z_i \oplus e_i$.

The clean codeword $c$ provides the parity structure, the noise $e$ is the PRC noise, and the secret pad $z$ randomizes the parity pattern. The secret pad also gives the statistical false-positive guarantee used by our detector.

### B.3 "SAMPLING-ON-THE-FLY" ONLINE ZERO-BIT PRC

In this setting, we do not predetermine the block length of the PRC. Instead, it grows with the generated token length.

The scheme is largely the same as the fixed-length case presented in the main text. We now require that the parity at each location of the codeword depends only on previous positions. The fixed-block construction satisfies this requirement before its final global permutation, so in the online setting we omit this permutation.

Additionally, since we do not know the block length $n$ in advance, we need to ensure that the number of parity checks at a current length $L$ stays close to the desired parity-check density. Let $\rho$ be the desired fraction of positions that are parity positions. In our experiments, $\rho = 0.99$.

Because a weight-$t$ parity check requires $t-1$ earlier positions, we assign the first $t-1$ positions as free. For each prefix length $L$, let $r_L = 0$ for $L < t-1$, and for $L \geq t-1$ define

$$r_L = \min\big(\mathrm{round}(\rho L), L - t + 1\big). \tag{20}$$

For each subsequent position $i$, we assign position $i$ as a parity position if $r_i > r_{i-1}$. Otherwise it is a free position. Thus parity positions are added as often as necessary to keep their fraction close to $\rho$, while ensuring that each parity check only depends on earlier positions.

**Definition B.5** (Online pseudorandom code generator)**.** Let $t$ be the parity-check weight and $\eta$ the PRC noise rate. Fix a keyed rule for choosing parity-check supports and a keyed pad sequence.

The online PRC-GENERATE produces a binary sequence $x_1, x_2, \ldots$. At each position $i$:

1. Sample a clean codeword bit $c_i$ as follows.
   - If position $i$ is free, sample $c_i \leftarrow \mathrm{Unif}(\{0,1\})$.
   - If position $i$ is a parity position, choose $t-1$ earlier positions $j_1, \ldots, j_{t-1} < i$ according to the keyed support rule and set $c_i = c_{j_1} \oplus \cdots \oplus c_{j_{t-1}}$. This creates the weight-$t$ parity check
   $$c_{j_1} \oplus \cdots \oplus c_{j_{t-1}} \oplus c_i = 0. \tag{21}$$
2. Sample a noise bit $e_i \sim \mathrm{Ber}(\eta)$.
3. Generate the pad bit $z_i$ from the keyed pad sequence.
4. Output $x_i = c_i \oplus e_i \oplus z_i$.

After generating a sequence of length $L$, the parity checks introduced so far form a parity-check matrix $P_L \in \mathbb{F}_2^{r_L \times L}$, and the generated clean codeword satisfies $P_L c_{1:L} = 0$.

The matrix $P_L$ has full row rank: each parity-check row has a distinct newest coordinate $i$, while all other nonzero coordinates in that row occur strictly earlier.

For the online construction, the pad sequence can be generated from a finite secret key $K_z$. At position $i$, compute $z_i = \mathrm{PRF}_{K_z}(i)$. This allows the pad sequence to grow together with the generated text. For a PRF-generated pad, the corresponding false-positive guarantee is computational, up to the distinguishing advantage of the PRF.

### B.4 Multiblock fixed-block PRC

The fixed-block PRC construction in the main text produces a PRC sequence of length $n$. If we want to generate a sequence of length $T > n$, we can extend the construction by sampling a fresh PRC encoding after each block of $n$ positions.

Suppose for simplicity that $T = Bn$ for an integer $B$. For each block $\ell = 1, \ldots, B$, sample a fresh PRC sequence $x^{(\ell)} \in \{0,1\}^n$. The full sequence is the concatenation $x = x^{(1)} \| \cdots \| x^{(B)}$.

For detection, apply the PRC detector separately to each block and output *watermarked* if any block is detected.

To maintain an overall target false-positive rate $\alpha$, use target false-positive rate $\alpha/B$ for each block. By Theorem 3.3, each block has false-positive rate at most $\alpha/B$. Therefore, by a union bound, $\Pr[\text{at least one block produces a false positive}] \leq \sum_{\ell=1}^{B} \frac{\alpha}{B} = \alpha$.

### B.5 Practical undetectability

PRC pseudorandomness requires polynomially many fresh encodings under one key to be computationally indistinguishable from independent uniform strings (Christ & Gunn, 2024, Definition 1). Christ and Gunn establish this guarantee for their zero-bit PRC constructions as the codeword length $n$ grows, under their cryptographic assumptions and parameter choices, including $t = \Theta(\log n)$ (Christ & Gunn, 2024, Theorems 1 and 2). Under this PRC pseudorandomness assumption, Corollary 3.2.1 establishes asymptotic computational undetectability for our watermark. This rules out polynomial-time distinguishers with non-negligible advantage that lack access to the watermarking key or detector, including tests based on text quality or diversity.

Our main experiments instead use $t = 3$ to improve detection at practical response lengths. We therefore assess generation behavior empirically at these parameter settings. Our evaluations show similar measured quality and diversity to unwatermarked generation (Table 2 and Appendix D.10). Neither of the two evaluated black-box detection tests detects the PRC watermark (Appendix D.11).

Together, these findings provide empirical support for undetectability in the tested settings, but do not guarantee undetectability against a motivated adversary targeting these concrete parameter choices.

Recent work (Wang et al., 2025) introduces targeted attacks on PRC watermarks, including key recovery, distinguishing, and noise-overlay removal. However, these attacks require access to the encoding key. Our scheme uses a secret-key deployment in which the model provider retains the encoding key, while adversaries may query the watermarked model and observe its outputs. Therefore, the attacks from this work do not directly apply to our PRC watermarking scheme.

Future work should assess distinguishing attacks beyond those evaluated here, including attacks tailored to the PRC construction that do not require access to the encoding key, and develop concrete security guarantees for practical parameter choices.

## C Details on implementation and experimental setup

Unless otherwise stated, all results use 500 C4 RealNewsLike prompts per configuration with the proven Hoeffding threshold $\tau = \sqrt{2\nu \log(1/\alpha)}$ and target FPR $\alpha = 10^{-3}$. We use check weight $t = 3$, a single length-$n$ code block with $T = n$ and $r = \mathrm{round}(0.99n)$.

With 500 trials, 95% binomial confidence intervals are approximately $\pm 4.4$ percentage points at 50% TPR and $\pm 2.6$ percentage points at 90% TPR.

### C.1 BINARY VOCABULARY PARTITION

As described in Section B.1.1, we use binary bucketing to map the language-model vocabulary to $\{0, 1\}$. We take a seeded random permutation of the vocabulary and assign approximately half of the tokens to each of $V_0$ and $V_1$.

The two bins have approximately equal cardinality, but their probability masses need not be equal. At generation position $i$, the probability mass on $V_1$ is $p_i = \sum_{v \in V_1} \pi_i(v)$.

We use binary bucketing instead of assigning multiple bits to each vocabulary token and selecting several bits at a time. Using more bins makes the individual bin probabilities smaller and more uneven. In preliminary experiments, using two or more bits per token severely degraded the detection performance. We therefore use one binary observation per generated token throughout our experiments.

### C.2 GENERATION AND DETECTION MODELS

For the primary detection experiments, we use Qwen3-0.6B-Base and Qwen3-8B-Base (Yang et al., 2025) to generate continuations from the RealNewsLike subset of C4 (Raffel et al., 2020), following Kirchenbauer et al. (2023); Kuditipudi et al. (2023). We disable the chat template and use temperature 1 with the full vocabulary unless otherwise stated.

For entropy-aware detection at positions $i \geq 2$, we teacher-force the detector model on the observed prefix $y_1, \ldots, y_{i-1}$, without the original prompt. Writing $\tilde{\pi}_i$ for its next-token distribution, we obtain $\tilde{p}_i = \sum_{v \in V_1} \tilde{\pi}_i(v)$. The first token has no preceding generated context, so we set its soft score $s_1 = 0$ without assigning a probability estimate $\tilde{p}_1$. We retain its position in the PRC sequence. Any parity check containing it therefore contributes zero to the detector.

Unless otherwise stated, generation and probability recovery use the same model. We also evaluate Qwen3-0.6B-Base as the detector model for Qwen3-8B-Base generations (Appendix D.2).

For the quality benchmarks in Table 2, we use Qwen3-0.6B with its chat template and thinking enabled. The base models perform very poorly on several of these benchmarks, often close to random, so the scores and differences between them would not be meaningful. We also note that standard pre-training benchmarks based on logits would not be meaningful for evaluating the watermark because the watermark does not alter the model logits. It only modifies the sampling procedure. The black-box detection tests also use Qwen3-0.6B but with thinking disabled (Appendix D.11).

# D ADDITIONAL EXPERIMENTAL RESULTS

See Section C for details on the experiment setup.

In the section we present the details and results of the following experiments.

| Section | Additional results |
|---|---|
| D.1 | Full detectability sweep |
| D.2 | Detector model choice |
| D.3 | Generation model size |
| D.4 | Parity-check weight |
| D.5 | PRC noise rate |
| D.6 | Observed false-positive rates |
| D.7 | Single vs. multi-block PRC |
| D.8 | Fixed-block vs. online PRC |
| D.9 | Comparison with Wang et al. (2025) |
| D.10 | Diversity and detectability across watermarks |
| D.11 | Black-box detection tests |
| D.13 | Robustness to random token substitutions |
| D.12 | Robustness to watermark-stealing attacks |
| D.14 | Examples of watermarked generations |

## D.1 FULL DETECTABILITY SWEEP

Table 3: Detection performance of the fixed-block PRC watermark (single length-$n$ code block, $T = n$, $r \approx 0.99n$, $t = 3$, Qwen3-0.6B-Base). Cells report TPR and observed FPR (%) over 500 watermarked and 500 unwatermarked completions from C4 RealNewsLike prompts per setting, using the Hoeffding threshold with target FPR $10^{-3}$. Posterior-mean detection achieves the highest TPR in every configuration.

| | | TPR (%) | | | FPR (%) | | |
|---|---|---|---|---|---|---|---|
| $\eta$ | $n$ | Posterior mean | Entropy weighted | Naive | Posterior mean | Entropy weighted | Naive |
| 0.05 | 256 | 62.4 | 43.8 | 25.2 | 0.0 | 0.0 | 0.0 |
| 0.05 | 400 | 86.0 | 71.0 | 52.0 | 0.0 | 0.0 | 0.0 |
| 0.05 | 416 | 88.2 | 73.6 | 51.0 | 0.0 | 0.0 | 0.0 |
| 0.05 | 448 | 90.0 | 79.4 | 58.2 | 0.0 | 0.0 | 0.0 |
| 0.05 | 512 | 94.4 | 84.2 | 68.8 | 0.0 | 0.0 | 0.0 |
| 0.05 | 1024 | 99.0 | 97.8 | 92.6 | 0.0 | 0.2 | 0.0 |
| 0.10 | 256 | 27.0 | 18.8 | 9.4 | 0.0 | 0.0 | 0.0 |
| 0.10 | 400 | 49.4 | 36.8 | 24.0 | 0.0 | 0.0 | 0.0 |
| 0.10 | 512 | 67.0 | 49.4 | 33.6 | 0.0 | 0.0 | 0.0 |
| 0.10 | 768 | 89.6 | 77.0 | 56.6 | 0.0 | 0.0 | 0.2 |
| 0.10 | 1024 | 95.2 | 89.2 | 77.0 | 0.0 | 0.0 | 0.0 |
| 0.15 | 256 | 9.4 | 7.4 | 2.4 | 0.0 | 0.0 | 0.0 |
| 0.15 | 400 | 17.6 | 9.8 | 5.8 | 0.0 | 0.2 | 0.2 |
| 0.15 | 512 | 29.6 | 22.0 | 11.2 | 0.0 | 0.0 | 0.0 |
| 0.15 | 1024 | 74.6 | 60.8 | 40.2 | 0.0 | 0.0 | 0.0 |
| 0.15 | 1504 | 91.0 | 83.4 | 68.2 | 0.0 | 0.0 | 0.0 |
| 0.15 | 2048 | 97.6 | 95.2 | 88.2 | 0.0 | 0.0 | 0.0 |
| 0.20 | 2048 | 72.2 | 59.4 | 45.8 | 0.0 | 0.0 | 0.0 |
| 0.20 | 4096 | 96.8 | 95.0 | 89.4 | 0.2 | 0.2 | 0.0 |

### D.2 Effect of model choice for entropy-aware detection

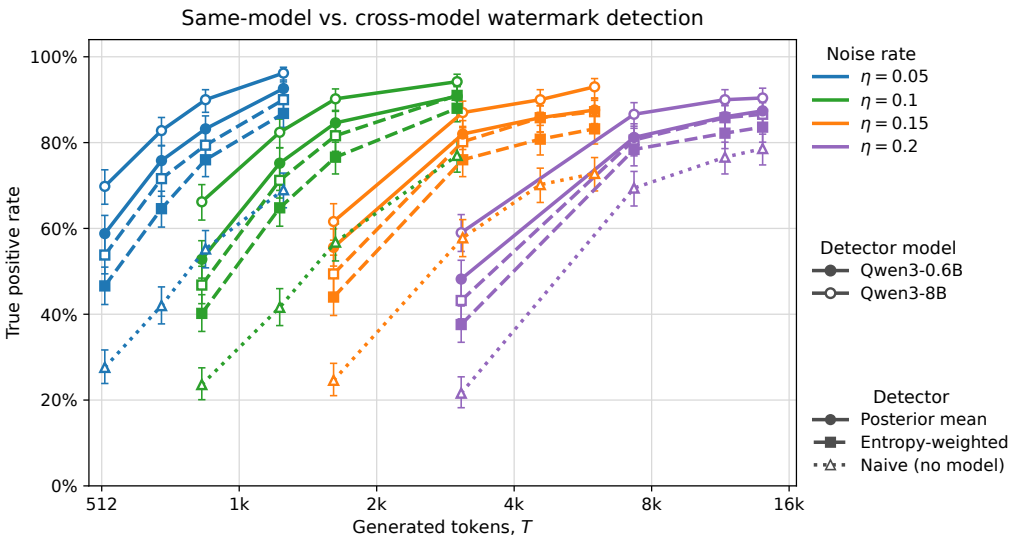



Figure 2: **Effect of the detection model on detection power.** We compare detection power using Qwen3-8B-Base and Qwen3-0.6B-Base as the number of generated tokens $T$ increases. The horizontal axis uses a logarithmic scale. Both models evaluate the same responses generated by Qwen3-8B-Base. Naive detection does not require a language model and is shown once per noise rate. Error bars show pointwise 95% Wilson confidence intervals, conditional on the evaluated keys and settings. Settings: online PRC, $t = 3$, temperature 1, target FPR $0.1\%$, and 500 watermarked responses per setting.

Below are the main takeaways from the results of Figure 2 and Table 4: **Watermarks remain detectable with a smaller model.** In practically relevant regimes where the matched 8B posterior-mean detector achieves approximately 90% TPR, the 0.6B detector achieves 83.2–86.0% TPR, a reduction of 4.0–6.8 percentage points (Table 4). Nevertheless, the matched 8B model achieves higher TPR than the mismatched 0.6B model in all 32 comparisons. All differences are significant under two-sided exact McNemar tests with Holm correction across the 32 comparisons (adjusted $p \leq 0.00342$).

**Longer responses reduce the TPR loss from using a smaller model.** For both detectors and every noise level, the mismatch penalty is smaller at the longest evaluated length than at the shortest. However, it remains 3.0–5.4 percentage points for posterior-mean detection and 3.0–4.0 points for entropy-weighted detection at the longest lengths (Figure 2).

Table 4: **Effect of the detection model on detection power.** We compare matched (8B) and mismatched (0.6B) detection on the same Qwen3-8B-Base responses at every length plotted in Figure 2. Subscripts and superscripts give the lower and upper bounds of pointwise 95% Wilson confidence intervals, conditional on the evaluated keys and settings. Bolded rows show $T_{90}$, the shortest evaluated token length at which the posterior-mean detector reaches at least 90% TPR. Naive detection is model-independent. The $\Delta$ columns report 0.6B minus 8B TPR in percentage points. Settings: online PRC, $t = 3$, temperature 1, nominal target FPR $\alpha = 10^{-3}$, and 500 watermarked completions per setting.

| | | Matched (8B) | | Mismatched (0.6B) | | | $\Delta$ TPR (pp) | |
|---|---|---|---|---|---|---|---|---|
| $\eta$ | $T$ | Posterior mean | Entropy weighted | Posterior mean | Entropy weighted | Naive | Posterior mean | Entropy weighted |
| 0.05 | 520 | $69.8^{73.7}_{65.6}$ | $53.8^{58.1}_{49.4}$ | $58.8^{63.0}_{54.4}$ | $46.6^{51.0}_{42.3}$ | $27.6^{31.7}_{23.9}$ | $-11.0$ | $-7.2$ |
| | 692 | $82.8^{85.9}_{79.2}$ | $71.6^{75.4}_{67.5}$ | $75.8^{79.3}_{71.9}$ | $64.6^{68.7}_{60.3}$ | $42.0^{46.4}_{37.8}$ | $-7.0$ | $-7.0$ |
| | **864** | $\mathbf{90.0}^{92.3}_{87.1}$ | $\mathbf{79.4}^{82.7}_{75.6}$ | $\mathbf{83.2}^{86.2}_{79.7}$ | $\mathbf{76.0}^{79.5}_{72.1}$ | $\mathbf{55.2}^{59.5}_{50.8}$ | **−6.8** | **−3.4** |
| | 1 280 | $96.2^{97.6}_{94.1}$ | $90.0^{92.3}_{87.1}$ | $92.6^{94.6}_{90.0}$ | $86.8^{89.5}_{83.6}$ | $69.0^{72.9}_{64.8}$ | $-3.6$ | $-3.2$ |
| 0.10 | 848 | $66.2^{70.2}_{61.9}$ | $46.8^{51.2}_{42.5}$ | $52.8^{57.1}_{48.4}$ | $40.2^{44.6}_{36.0}$ | $23.6^{27.5}_{20.1}$ | $-13.4$ | $-6.6$ |
| | 1 256 | $82.4^{85.5}_{78.8}$ | $71.2^{75.0}_{67.1}$ | $75.2^{78.8}_{71.2}$ | $64.8^{68.9}_{60.5}$ | $41.6^{46.0}_{37.4}$ | $-7.2$ | $-6.4$ |
| | **1 664** | $\mathbf{90.2}^{92.5}_{87.3}$ | $\mathbf{81.6}^{84.8}_{78.0}$ | $\mathbf{84.6}^{87.5}_{81.2}$ | $\mathbf{76.6}^{80.1}_{72.7}$ | $\mathbf{56.8}^{61.1}_{52.4}$ | **−5.6** | **−5.0** |
| | 3 072 | $94.2^{95.9}_{91.8}$ | $91.0^{93.2}_{88.2}$ | $90.8^{93.0}_{87.9}$ | $88.0^{90.6}_{84.9}$ | $77.0^{80.5}_{73.1}$ | $-3.4$ | $-3.0$ |
| 0.15 | 1 648 | $61.6^{65.8}_{57.3}$ | $49.4^{53.8}_{45.0}$ | $55.6^{59.9}_{51.2}$ | $44.0^{48.4}_{39.7}$ | $24.6^{28.6}_{21.0}$ | $-6.0$ | $-5.4$ |
| | 3 160 | $87.0^{89.7}_{83.8}$ | $80.2^{83.5}_{76.5}$ | $82.0^{85.1}_{78.4}$ | $76.0^{79.5}_{72.1}$ | $57.8^{62.1}_{53.4}$ | $-5.0$ | $-4.2$ |
| | **4 672** | $\mathbf{90.0}^{92.3}_{87.1}$ | $\mathbf{85.8}^{88.6}_{82.5}$ | $\mathbf{85.8}^{88.6}_{82.5}$ | $\mathbf{80.8}^{84.0}_{77.1}$ | $\mathbf{70.2}^{74.0}_{66.0}$ | **−4.2** | **−5.0** |
| | 6 144 | $93.0^{94.9}_{90.4}$ | $87.2^{89.8}_{84.0}$ | $87.6^{90.2}_{84.4}$ | $83.2^{86.2}_{79.7}$ | $72.8^{76.5}_{68.7}$ | $-5.4$ | $-4.0$ |
| 0.20 | 3 136 | $59.0^{63.2}_{54.6}$ | $43.2^{47.6}_{38.9}$ | $48.2^{52.6}_{43.9}$ | $37.6^{41.9}_{33.5}$ | $21.6^{25.4}_{18.2}$ | $-10.8$ | $-5.6$ |
| | 7 496 | $86.6^{89.3}_{83.3}$ | $80.6^{83.8}_{76.9}$ | $81.2^{84.4}_{77.5}$ | $78.4^{81.8}_{74.6}$ | $69.4^{73.3}_{65.2}$ | $-5.4$ | $-2.2$ |
| | **11 856** | $\mathbf{90.0}^{92.3}_{87.1}$ | $\mathbf{85.8}^{88.6}_{82.5}$ | $\mathbf{86.0}^{88.8}_{82.7}$ | $\mathbf{82.2}^{85.3}_{78.6}$ | $\mathbf{76.6}^{80.1}_{72.7}$ | **−4.0** | **−3.6** |
| | 14 336 | $90.4^{92.7}_{87.5}$ | $86.6^{89.3}_{83.3}$ | $87.4^{90.0}_{84.2}$ | $83.6^{86.6}_{80.1}$ | $78.6^{82.0}_{74.8}$ | $-3.0$ | $-3.0$ |

### D.3 Effect of Model Size

Table 5: **Effect of model size on detection power (%).** Numerical data for Figure 1b. For each noise rate $\eta$, the two rows show the shortest evaluated token lengths $n$ at which posterior-mean detection reaches at least 90% TPR for Qwen3-0.6B-Base and Qwen3-8B-Base, respectively. The 8B model requires longer text to reach this TPR, with the gap increasing at higher noise rates. All three detectors are compared at the selected lengths. Settings: online PRC, $t = 3$, target FPR $\alpha = 10^{-3}$, and 500 prompts per configuration. – denotes unevaluated settings.

| | | Qwen3-0.6B-Base | | | Qwen3-8B-Base | | |
|---|---|---|---|---|---|---|---|
| $\eta$ | $n$ | Posterior-Mean | Entropy-Weighted | Naive | Posterior-Mean | Entropy-Weighted | Naive |
| 0.05 | 520 | 90.2 | 78.8 | 61.2 | 69.8 | 53.8 | 27.6 |
| 0.05 | 864 | – | – | – | 90.0 | 79.4 | 55.2 |
| 0.10 | 848 | 90.0 | 79.0 | 58.4 | 66.2 | 46.8 | 23.6 |
| 0.10 | 1,664 | – | – | – | 90.2 | 81.6 | 56.8 |
| 0.15 | 1,648 | 90.0 | 84.8 | 61.4 | 61.6 | 49.4 | 24.6 |
| 0.15 | 4,672 | – | – | – | 90.0 | 85.8 | 70.2 |
| 0.20 | 3,136 | 90.0 | 80.6 | 65.4 | 59.0 | 43.2 | 21.6 |
| 0.20 | 11,856 | – | – | – | 90.0 | 85.8 | 76.6 |

Table 6: **Effect of model size on token and binary bucket entropy.** $H_{\text{tok}}$ measures uncertainty over the full vocabulary. $H_{\text{bucket}} = H_2(p_i)$ measures uncertainty between the two watermark token groups, where $p_i$ is the total probability assigned to one group. Both entropies are measured in bits. The final column reports the percentage of token positions with $H_{\text{bucket}} < 0.1$ bits. For each model, statistics are computed over the first 1,808 tokens of each of 500 unwatermarked continuations, using the generation model with the original prompt and preceding generated tokens as context.

| Model | Mean $H_{\text{tok}}$ | Mean $H_{\text{bucket}}$ | $H_{\text{bucket}} < 0.1$ (%) |
|---|---|---|---|
| Qwen3-0.6B | 6.485 | 0.866 | 3.3 |
| Qwen3-8B | 4.551 | 0.740 | 10.7 |
| Qwen3-14B | 4.151 | 0.715 | 12.4 |

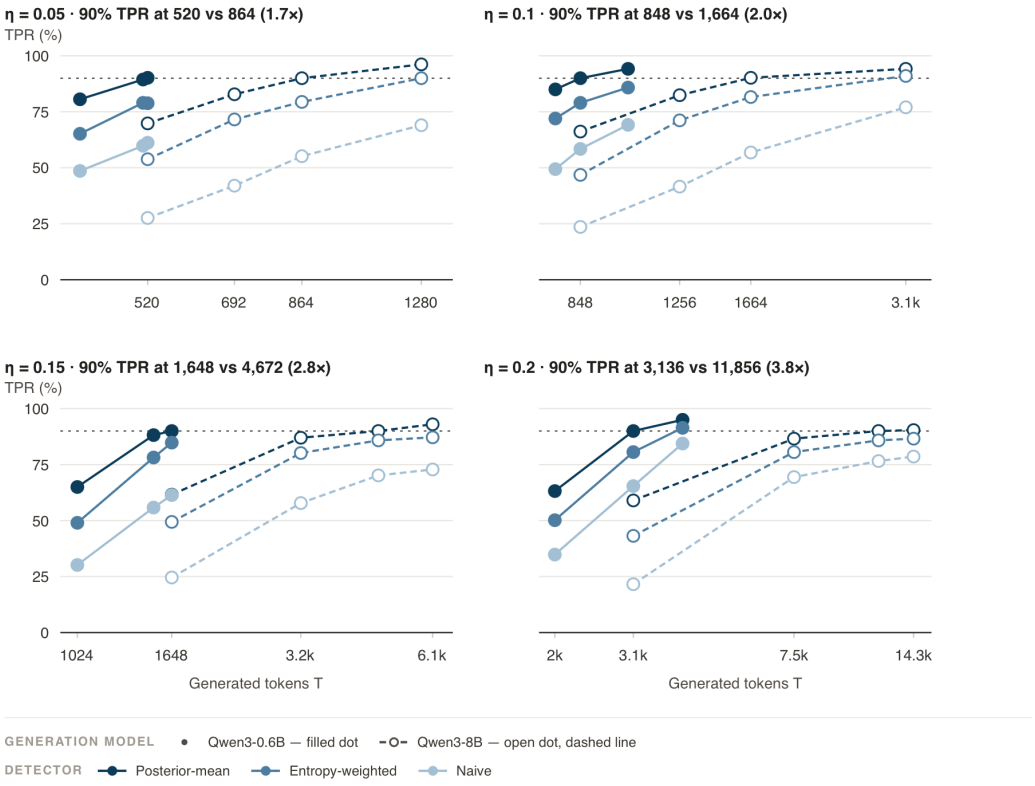



Figure 3: **Effect of model size on detection power.** TPR versus generated tokens $T$ for online PRC with a matched detector ($t = 3$, target FPR $10^{-3}$) for different noise rates $\eta$.

### D.3.1 Effect of model size on entropy and effective detector signal

As shown in Table 6, mean bucket entropy decreases from $0.866$ bits for Qwen3-0.6B to $0.740$ bits for Qwen3-8B and $0.715$ bits for Qwen3-14B. The fraction of nearly deterministic bucket assignments, with $H_{\text{bucket}} < 0.1$ bits, increases from $3.3\%$ to $10.7\%$ and $12.4\%$, respectively. Such positions provide little freedom to embed a watermark bit while preserving the sampling distribution. These statistics describe generation-time entropy. Combined with the analysis in Section A.3, this provides an explanation for the detection behavior observed in Table 5.

### D.4 Effect of parity-check weight

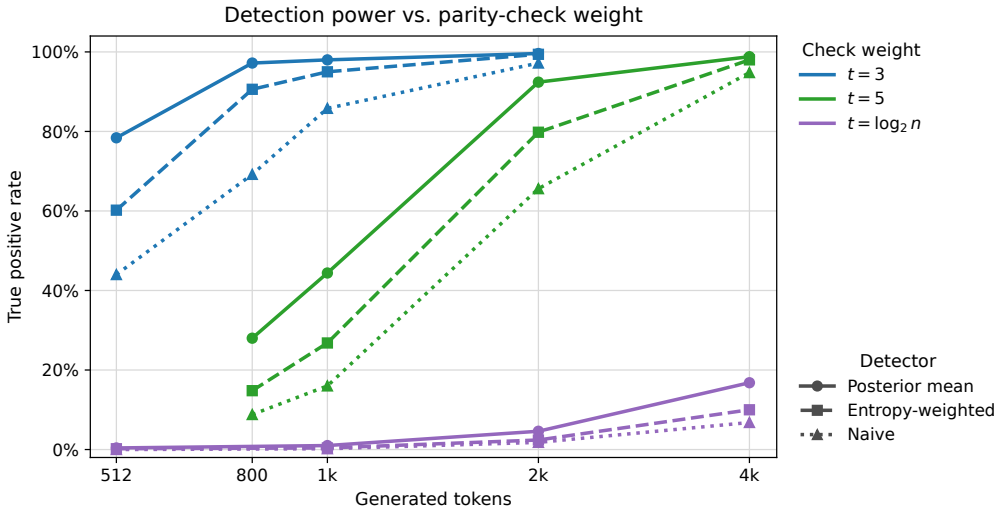



Figure 4: **Effect of parity-check weight on detection power.** Using more token positions in each parity check (larger $t$) makes the watermark harder to detect at the same text length. Results use fixed-block PRC with Qwen3-0.6B, $\eta = 0.05$, target FPR $0.1\%$, and 500 watermarked responses per setting. The horizontal axis shows total generated tokens $T = 2n$, where $n$ is the block length. For $t = \log_2 n$, the weight increases with block length: $t = 8, 9, 10, 11$ at $n = 256, 512, 1024, 2048$, respectively.

Table 7: **Effect of parity-check weight on detection power.** Numerical results for the configurations shown in Figure 4. The three detector columns report detection power (%), and the final column reports the ratio of posterior-mean to entropy-weighted detection power. Each parity check involves $t$ bits, and $r = n - 2\lfloor \log_2 \binom{n}{t} \rfloor - 1$ is the number of checks per length-$n$ codeword.

| $t$ | $n$ | $r$ | Posterior-Mean | Entropy-Weighted | Naive | Posterior-Mean/Entropy-Weighted |
|---|---|---|---|---|---|---|
| 3 | 256 | 213 | 78.4 | 60.2 | 44.0 | 1.30 |
| | 400 | 353 | 97.2 | 90.6 | 69.2 | 1.07 |
| | 512 | 463 | 98.0 | 95.0 | 85.8 | 1.03 |
| | 1024 | 969 | 99.6 | 99.4 | 97.2 | 1.00 |
| 5 | 400 | 327 | 28.0 | 14.8 | 8.8 | 1.89 |
| | 512 | 435 | 44.4 | 26.8 | 16.0 | 1.66 |
| | 1024 | 937 | 92.4 | 79.8 | 65.6 | 1.16 |
| | 2048 | 1951 | 98.8 | 98.0 | 94.8 | 1.01 |
| $\log_2 n$ | 256 | 159 | 0.4 | 0.2 | 0.0 | 2.00 |
| | 512 | 387 | 1.0 | 0.4 | 0.2 | 2.50 |
| | 1024 | 867 | 4.6 | 2.4 | 1.8 | 1.92 |
| | 2048 | 1857 | 16.8 | 10.0 | 6.8 | 1.68 |

The configurations with $t = \log_2 n$ have substantially lower detection power than those with $t = 3$ or $t = 5$ at the same text lengths (Table 7). These results motivate our use of $t = 3$ in the main experiments. Appendix B.5 discusses how this choice relates to the asymptotic undetectability guarantee.

### D.5 Effect of PRC noise parameter

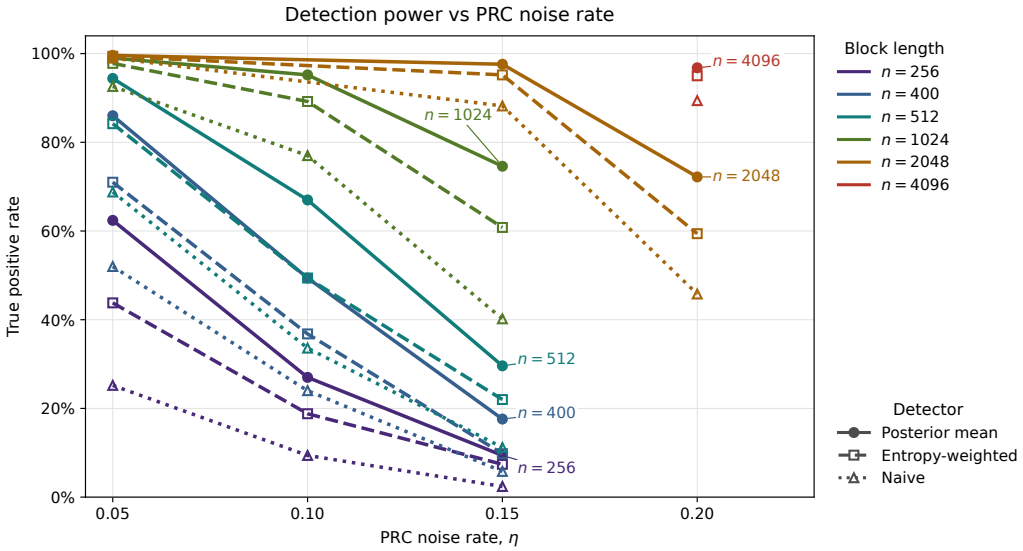



Figure 5: **Effect of PRC noise rate on detection power.** TPR (%) versus $\eta$ for single-block PRC ($T = n$) at different block lengths $n$. Higher noise reduces detection power. Longer blocks compensate for this loss. Settings: Qwen3-0.6B-Base, $t = 3$, target FPR $0.1\%$, and 500 watermarked responses per point.

### D.6 Observed false-positive rates

Table 8: **Effect of the target false-positive rate on detection power.** Increasing the target FPR relaxes the detection threshold and increases the true-positive rate (TPR). At targets of $5\%$ and $10\%$, the observed FPR remains well below the target. $T$ denotes the total number of generated tokens. Each target is evaluated on the same 500 watermarked and 500 unwatermarked responses per configuration. Settings: multi-block PRC, $T = 2n$, $t = 5$, and $\eta = 0.05$. Posterior-mean detection was not evaluated above a target FPR of $0.1\%$.

| | | TPR (%) | | Observed FPR (%) | |
|---|---|---|---|---|---|
| $T$ | Target FPR (%) | Entropy-Weighted | Naive | Entropy-Weighted | Naive |
| 800 | 0.1 | 14.8 | 8.8 | 0.0 | 0.0 |
| 800 | 1.0 | 31.6 | 17.6 | 1.4 | 1.0 |
| 800 | 5.0 | 52.6 | 27.2 | 1.4 | 1.0 |
| 800 | 10.0 | 59.6 | 38.2 | 2.0 | 1.8 |
| 1024 | 0.1 | 26.8 | 16.0 | 0.0 | 0.0 |
| 1024 | 1.0 | 45.6 | 28.0 | 0.2 | 0.4 |
| 1024 | 5.0 | 65.0 | 45.4 | 0.4 | 0.6 |
| 1024 | 10.0 | 71.8 | 53.2 | 1.6 | 1.4 |

### D.7 SINGLE VS. MULTI-BLOCK PRC

Table 9: **At a fixed token length $T$, allocating the entire sequence to a single length-$T$ code block outperforms splitting it into two length-$T/2$ blocks.** Detection power (%) as the number of code blocks increases from one to two, using approximately the same total number of generated tokens $T$. Each code block contains $n$ tokens, so increasing the number of blocks reduces the length of each block. Settings: Qwen3-0.6B, $\eta = 0.1$, $t = 3$, target FPR 0.1%, and 500 watermarked responses per configuration.

| | | | TPR (%) and gain (pp) | | |
|---|---|---|---|---|---|
| $T$ (tokens) | # of code blocks | $n$ | Posterior-Mean | Entropy-Weighted | Naive |
| 512 | 1 | 512 | 67.0 **+16.2** | 49.4 **+12.4** | 33.6 **+12.0** |
| 512 | 2 | 256 | 50.8 | 37.0 | 21.6 |
| 768 | 1 | 768 | 89.6 **+11.2** | 77.0 **+14.6** | 56.6 **+9.0** |
| 800 | 2 | 400 | 78.4 | 62.4 | 47.6 |

### D.8 FIXED-BLOCK VS. ONLINE PRC CONSTRUCTION

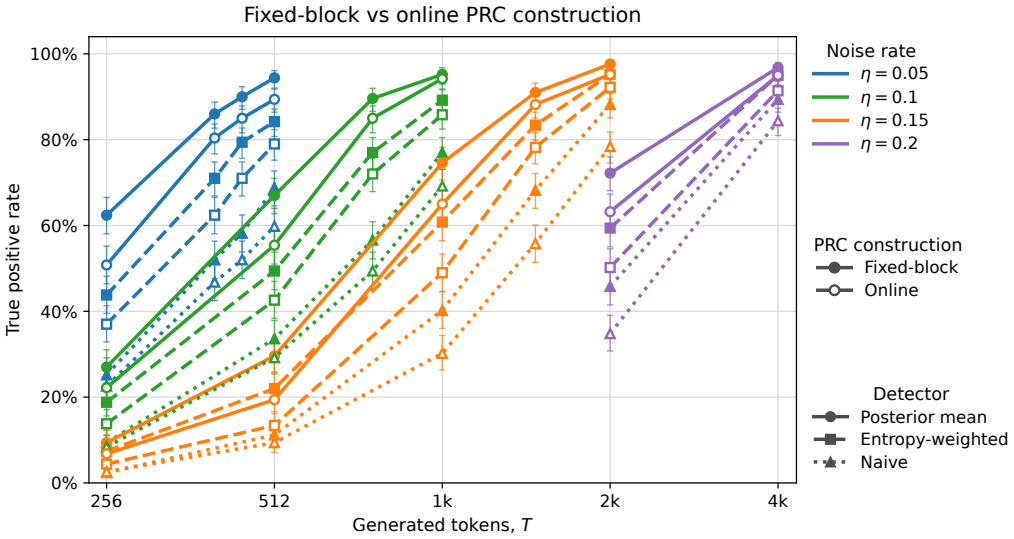



Figure 6: **Detection power of fixed-block vs. online PRC constructions.** True positive rate (TPR) is compared at equal generated-token lengths $T$ across four noise rates $\eta$. Error bars show pointwise 95% Wilson confidence intervals from 500 watermarked responses per setting, conditional on the evaluated keys and settings. Experiments use Qwen3-0.6B-Base, $t = 3$, target FPR 0.1%. Fixed-block uses $T = n$. The horizontal axis uses log scale.

As observed in Figure 6 and in Table, in the practically relevant high-TPR regime, online PRC incurs only a small degradation relative to fixed-block PRC when entropy recovery is applied through posterior-mean or entropy-weighted detection. In contrast, the naive detector generally exhibits a larger loss in detection power. For example, at $(\eta, T) = (0.15, 2048)$ and $(0.20, 4096)$, posterior-mean TPR decreases by 2.4 and 1.8 percentage points, respectively, compared with 9.8 and 5.0 points for the naive detector.

In exploratory two-sided exact McNemar tests on 500 paired responses, with Holm correction across the four posterior-mean endpoint comparisons, neither posterior-mean difference is statistically significant (adjusted $p = 0.173$ and 0.422). At $(\eta, T) = (0.10, 1024)$, the posterior-mean gap is also

small (1.0 percentage point; adjusted $p = 0.560$). However, at $(\eta, T) = (0.05, 512)$, the 5.0-point posterior-mean gap remains statistically significant (adjusted $p = 0.019$). For the naive detector, the reduction from 88.2% to 78.4% at $(\eta, T) = (0.15, 2048)$ is statistically significant even under a conservative paired-test bound (Holm-adjusted $p \leq 7.4 \times 10^{-4}$ across the four naive endpoint comparisons).

Table 10: **Effect of PRC construction on detection power: fixed-block versus online PRC.** The two constructions are compared at the same total number of generated tokens $T$. For each noise rate $\eta$, we select the evaluated fixed-block lengths with posterior-mean TPRs closest to $90\%$ from below and above, then evaluate online PRC at those same lengths. Each fixed-block setting uses one code block of length $n = T$. Superscripts and subscripts $+/-$ values show pointwise 95% Wilson confidence intervals, expressed as offsets in percentage points. Intervals are conditional on the evaluated keys and settings. Settings: Qwen3-0.6B for both generation and detection, $t = 3$, target FPR $0.1\%$, and 500 watermarked completions per row. See Figure 6 for the plot of this data.

| | | | TPR (%; 95% CI) | | |
|---|---|---|---|---|---|
| $\eta$ | $T$ (tokens) | Construction | Posterior-Mean | Entropy-Weighted | Naive |
| 0.05 | 416 | Fixed-block | $88.2^{+2.5}_{-3.1}$ | $73.6^{+3.7}_{-4.0}$ | $51.0^{+4.4}_{-4.4}$ |
| 0.05 | 416 | Online | $80.6^{+3.2}_{-3.7}$ | $65.2^{+4.0}_{-4.3}$ | $48.6^{+4.4}_{-4.4}$ |
| 0.05 | 512 | Fixed-block | $94.4^{+1.7}_{-2.4}$ | $84.2^{+2.9}_{-3.5}$ | $68.8^{+3.9}_{-4.2}$ |
| 0.05 | 512 | Online | $89.4^{+2.4}_{-3.0}$ | $79.0^{+3.3}_{-3.8}$ | $59.8^{+4.2}_{-4.4}$ |
| 0.10 | 768 | Fixed-block | $89.6^{+2.4}_{-3.0}$ | $77.0^{+3.5}_{-3.9}$ | $56.6^{+4.3}_{-4.4}$ |
| 0.10 | 768 | Online | $85.0^{+2.9}_{-3.4}$ | $72.0^{+3.8}_{-4.1}$ | $49.4^{+4.4}_{-4.4}$ |
| 0.10 | 1024 | Fixed-block | $95.2^{+1.6}_{-2.2}$ | $89.2^{+2.4}_{-3.0}$ | $77.0^{+3.5}_{-3.9}$ |
| 0.10 | 1024 | Online | $94.2^{+1.7}_{-2.4}$ | $85.8^{+2.8}_{-3.3}$ | $69.2^{+3.9}_{-4.2}$ |
| 0.15 | 1024 | Fixed-block | $74.6^{+3.6}_{-4.0}$ | $60.8^{+4.2}_{-4.3}$ | $40.2^{+4.4}_{-4.2}$ |
| 0.15 | 1024 | Online | $65.0^{+4.1}_{-4.3}$ | $49.0^{+4.4}_{-4.4}$ | $30.2^{+4.2}_{-3.9}$ |
| 0.15 | 1504 | Fixed-block | $91.0^{+2.2}_{-2.8}$ | $83.4^{+3.0}_{-3.5}$ | $68.2^{+3.9}_{-4.2}$ |
| 0.15 | 1504 | Online | $88.2^{+2.5}_{-3.1}$ | $78.2^{+3.4}_{-3.8}$ | $55.8^{+4.3}_{-4.4}$ |
| 0.20 | 2048 | Fixed-block | $72.2^{+3.7}_{-4.1}$ | $59.4^{+4.2}_{-4.4}$ | $45.8^{+4.4}_{-4.3}$ |
| 0.20 | 2048 | Online | $63.2^{+4.1}_{-4.3}$ | $50.2^{+4.4}_{-4.4}$ | $34.8^{+4.3}_{-4.0}$ |
| 0.20 | 4096 | Fixed-block | $96.8^{+1.2}_{-1.9}$ | $95.0^{+1.6}_{-2.3}$ | $89.4^{+2.4}_{-3.0}$ |
| 0.20 | 4096 | Online | $95.0^{+1.6}_{-2.3}$ | $91.4^{+2.2}_{-2.8}$ | $84.4^{+2.9}_{-3.4}$ |

### D.9 COMPARING OUR PRC WATERMARK DETECTOR WITH WANG ET AL. (2025)

We compare Wang's implementation of the Christ–Gunn detector Wang et al. (2025); Christ & Gunn (2024) with our (entropy-aware) posterior-mean detector on identical watermarked Qwen3-8B-Base completions. At each sampling temperature, ten watermark keys are crossed with sixteen prompts, giving 160 completions of 1,024 tokens. Generation uses Wang's watermark construction with 18 hierarchical bits per token, $n = 18{,}432$, $r = 17{,}510$ parity checks, $t = 3$, and $\eta = 0.10$. Generation and replay use BF16 and raw prompts without a chat template. Posterior-mean replay conditions on preceding completion tokens and assigns zero evidence to the first generated token.

For detection, we use two threshold settings. The default setting uses the threshold provided in Wang et al.'s implementation (Wang et al., 2025) for the baseline and the analytic Hoeffding threshold for our posterior-mean detector with target FPR $0.1\%$. In the empirically calibrated setting, each detector's threshold is selected based on 400 separate unwatermarked completions pooled across the five temperatures and crossed with 256 calibration keys. Cutoffs are frozen before evaluation, which uses another 400 unwatermarked completions and 256 disjoint evaluation keys.

Table 11: **Comparison of the baseline and posterior-mean detectors.** The baseline detector is Wang's implementation of the Christ–Gunn detector Wang et al. (2025); Christ & Gunn (2024). The posterior-mean detector is from Definition 3.6. Both detectors evaluate the same Qwen3-8B-Base completions of 160 watermarked texts per temperature, from 10 keys and 16 prompts, with 1,024 tokens per text. TPR gain is our TPR minus the baseline TPR, in percentage points (pp).

| | | **Baseline PRC detector** | | **Posterior mean (ours)** | | |
|---|---|---|---|---|---|---|
| Threshold setting | Temperature | TPR (%) | FPR (%) | TPR (%) | FPR (%) | TPR gain (pp) |
| Default | 1.0 | 0.00 | 0.0000 | 5.00 | 0.0000 | +5.00 |
| | 1.2 | 0.00 | 0.0000 | 97.50 | 0.0049 | +97.50 |
| | 1.4 | 33.75 | 0.0000 | 100.00 | 0.0098 | +66.25 |
| | 1.6 | 98.13 | 0.0000 | 100.00 | 0.0049 | +1.88 |
| | 1.8 | 100.00 | 0.0000 | 100.00 | 0.0000 | 0.00 |
| Empirically calibrated FPR $10^{-3}$ | 1.0 | 0.00 | 0.1172 | 11.25 | 0.0293 | +11.25 |
| | 1.2 | 89.38 | 0.1416 | 98.13 | 0.0684 | +8.75 |
| | 1.4 | 100.00 | 0.0684 | 100.00 | 0.1465 | 0.00 |
| | 1.6 | 100.00 | 0.1025 | 100.00 | 0.0781 | 0.00 |
| | 1.8 | 100.00 | 0.0879 | 100.00 | 0.0928 | 0.00 |

### D.10 COMPARING DIVERSITY AND DETECTABILITY ACROSS WATERMARKS

We evaluate the diversity of responses sampled with our PRC watermark, both between sequences (via self-BLEU score) and within sequences (via repetition metrics like repeated 4-grams and distinct 3-gram ratios). We also compare the detectability and diversity of our PRC watermark against state-of-the-art watermarks for language models, namely SynthID-Text (Dathathri et al., 2024) and TextSeal (Sander et al., 2026). Our main results are presented below in Table 12.

Table 12: **Response diversity across watermarking methods.** *Settings:* Qwen3-8B-Base, temperature 1, full vocabulary, 50 prompts, two 1,024-token responses each, target FPR $0.1\%$, for all watermarks. For PRC: online, $\eta = 0.05$, $t = 3$, $r = 0.99n$, posterior-mean detection. *Metrics:* Self-BLEU measures average overlap between the two responses to each prompt. Repeated 4-grams is the percentage of contiguous four-token sequences occurring after their first appearance within a response. Distinct-3 is the number of unique contiguous three-token sequences divided by their total count. *Notes:* We evaluate the effect of repeated context masking (Dathathri et al., 2024), referred to here as repeat handling, on repetition within individual responses and diversity between responses to the same prompt. With repeat handling, unwatermarked sampling is used at positions where the watermarking context repeats within a response. Without repeat handling, the watermarking rule is still applied at those positions. The two responses to each prompt use different sampling seeds; each method's watermark keys are held fixed across responses. Subscripts and superscripts give the lower and upper 95% confidence limits from 2,000 bootstrap resamples of prompts, keeping responses and methods paired. Comparator estimates are bold for differences from PRC that remain significant after Holm correction across all 35 two-sided paired permutation tests ($\alpha = 0.05$). Method labels are swapped within prompts, keeping both responses together. Tests use 100,000 permutations or exact enumeration when feasible. The displayed confidence intervals are unadjusted.

| Method | TPR (%) ↑ | Self-BLEU ↓ | Repeated 4-grams (%) ↓ | Distinct-3 ↑ |
|---|---|---|---|---|
| Unwatermarked | – | $0.01928\ ^{0.02229}_{0.01649}$ | $2.92\ ^{4.33}_{1.96}$ | $0.9514\ ^{0.9620}_{0.9379}$ |
| **PRC (ours)** | $\mathbf{97}\ ^{100}_{93}$ | $\mathbf{0.01890}\ ^{0.02227}_{0.01581}$ | $\mathbf{2.38}\ ^{3.16}_{1.68}$ | $\mathbf{0.9561}\ ^{0.9651}_{0.9462}$ |
| *With repeat handling* | | | | |
| SynthID depth 2 | $100\ ^{100}_{100}$ | $0.01987\ ^{0.02418}_{0.01592}$ | $3.28\ ^{4.39}_{2.32}$ | $0.9462\ ^{0.9587}_{0.9326}$ |
| SynthID depth 10 | $100\ ^{100}_{100}$ | $0.02206\ ^{0.02518}_{0.01909}$ | $2.42\ ^{3.14}_{1.83}$ | $0.9554\ ^{0.9635}_{0.9463}$ |
| SynthID depth 30 | $100\ ^{100}_{100}$ | $\mathbf{0.02890}\ ^{0.03392}_{0.02412}$ | $2.30\ ^{3.09}_{1.58}$ | $0.9589\ ^{0.9683}_{0.9489}$ |
| TextSeal $\alpha = .1$ | $100\ ^{100}_{100}$ | $\mathbf{0.04727}\ ^{0.05421}_{0.04082}$ | $2.74\ ^{3.35}_{2.14}$ | $0.9464\ ^{0.9551}_{0.9380}$ |
| Gumbel-Max | $100\ ^{100}_{100}$ | $\mathbf{0.20718}\ ^{0.23097}_{0.18343}$ | $2.81\ ^{3.71}_{2.05}$ | $0.9456\ ^{0.9557}_{0.9341}$ |
| *Without repeat handling* | | | | |
| SynthID depth 10 | $100\ ^{100}_{100}$ | $0.02138\ ^{0.02534}_{0.01775}$ | $\mathbf{9.35}\ ^{12.91}_{6.07}$ | $\mathbf{0.8880}\ ^{0.9211}_{0.8528}$ |
| TextSeal $\alpha = .1$ | $100\ ^{100}_{100}$ | $\mathbf{0.03770}\ ^{0.04440}_{0.03157}$ | $\mathbf{32.17}\ ^{36.95}_{27.15}$ | $\mathbf{0.6603}\ ^{0.7092}_{0.6139}$ |
| Gumbel-Max | $100\ ^{100}_{100}$ | $\mathbf{1.00000}\ ^{1.00000}_{1.00000}$ | $\mathbf{51.73}\ ^{59.43}_{44.12}$ | $\mathbf{0.4720}\ ^{0.5460}_{0.3978}$ |

The following are the main takeaways from Table 12:

**(1) Responses sampled with our PRC watermark show no statistically significant differences from unwatermarked responses on any of the three diversity metrics.** We compare two 1,024-token responses to each of the same 50 prompts. Responses sampled with our watermark and without watermarking have similar Self-BLEU (0.01890 versus 0.01928), repeated 4-grams (2.38% versus 2.92%), and Distinct-3 (0.9561 versus 0.9514). Thus, we find no statistically significant differences either in diversity between responses to the same prompt or in repetition within individual responses.

**(2) Our PRC watermark preserves greater diversity between responses to the same prompt than TextSeal, Gumbel-Max, or SynthID at depth 30.** Even when these other methods use repeat handling, their Self-BLEU values are 0.04727, 0.20718, and 0.02890, respectively, compared with 0.01890 for responses sampled with our watermark. All three differences are statistically significant, indicating greater overlap between responses sampled with these other methods. With

repeat handling, SynthID at depths 2 and 10 instead shows no statistically significant differences from our PRC watermark on any of the three diversity metrics.

**(3) Repeat handling reduces repetition within individual responses, but TextSeal and Gumbel-Max still produce less diverse responses to the same prompt than our PRC watermark.** Enabling repeat handling reduces repeated 4-grams from 9.35% to 2.42% for SynthID depth 10, from 32.17% to 2.74% for TextSeal, and from 51.73% to 2.81% for Gumbel-Max. Without repeat handling, all three methods have significantly more repeated 4-grams and lower Distinct-3 than our PRC watermark. With repeat handling, neither within-response metric differs significantly from our watermark. However, the significantly higher Self-BLEU of TextSeal and Gumbel-Max indicates that reducing repetition within a response does not necessarily resolve overlap between responses.

**(4) The PRC watermark requires longer responses for reliable detection.** In the same study, our watermark is detected in 55% of responses at 400 tokens and 97% at 1,024 tokens. The three other watermarking methods reach 100% at both lengths, although their differences from our watermark at 1,024 tokens are not statistically significant. In a separate length sweep over 500 responses sampled with our PRC watermark, 864 tokens is the shortest evaluated length reaching 90% TPR (450/500).

### D.11 Black-box detection tests

We test whether generated responses reveal the use of watermarking without access to the secret key or detector, using the tests of Gloaguen et al. (2024). The Red-Green test looks for consistent word preferences across carefully varied prompts. The Fixed-Sampling test looks for repeated responses to the same prompt.

We use Qwen3-0.6B with thinking disabled, temperature 1, and the full vocabulary. PRC uses the online construction with $t = 3$ and $\eta = 0.05$, keeping the same secret key and sampling a fresh codeword for each response.

**Red-Green test.** We combine 10 sentence prefixes with each digit from 1 to 9 repeated $H$ times, for $H \in \{4, 5\}$. The prompts ask the model to complete the sentence with one of four words, apples, bananas, oranges, or pears. This word list was selected in a pilot for its relatively balanced response frequencies.

For each scheme, we collect 100 successfully parsed responses per prompt, giving 18,000 responses in total. We compute the test statistic of Gloaguen et al. (2024), estimate $p$-values using 10,000 random shuffles, and apply a Bonferroni correction to account for testing both values of $H$.

**Fixed-Sampling test.** For each scheme, we generate 1,000 responses to the prompt "This is the story of", with at most 100 tokens per response. We track the number of distinct responses as more queries are added, averaging over 500 random orderings. Following Gloaguen et al. (2024), we compare this curve with a reference in which every response is distinct, using a Mann–Whitney $U$ test.

Table 13: **PRC resists evaluated black-box detection tests, while each baseline is flagged by at least one test.** Results on Qwen3-0.6B using the Red-Green (RG) and Fixed-Sampling (FS) tests of Gloaguen et al. (2024), without access to the watermarking key or detector. Smaller $p$-values provide stronger evidence of watermarking. The last column shows which test detects watermarking at the 5% significance level.

| Scheme | Red-Green (RG) | | Fixed-Sampling (FS) | | Detected by |
|---|---|---|---|---|---|
| | Statistic $H = 4$ / $H = 5$ | Adjusted $p$-value | Distinct responses | $p$-value | |
| Unwatermarked | 2 / 1 | 1.00 | 1,000/1,000 | 0.94 | Neither |
| **PRC** | **1 / 3** | **0.22** | **1,000/1,000** | **0.94** | **Neither** |
| KGW-2.0 | 10 / 10 | $0^{\dagger}$ | 1,000/1,000 | 0.94 | RG |
| SynthID-Text | 10 / 2 | $0^{\dagger}$ | 1,000/1,000 | 0.94 | RG ($H = 4$) |
| EXP | 2 / 2 | 1.00 | 252/1,000 | $2.2 \times 10^{-124}$ | FS |

$H$ is the number of digit repetitions in each RG prompt. RG $p$-values are adjusted for testing both $H = 4$ and $H = 5$. $^{\dagger}$An estimated zero means no shuffled trial matched or exceeded the observed statistic in 10,000 trials for at least one $H$.

**Results.** Neither test detects PRC or unwatermarked generation at the 5% significance level. The Red-Green test detects KGW-2.0 and SynthID-Text. The Fixed-Sampling test detects EXP, which produces only 252 distinct responses from 1,000 queries. PRC produces 1,000 distinct responses, matching the unwatermarked model. These results support the undetectability guarantee of PRC watermarks, unlike the baselines.

### D.12 Robustness to Stealing Attacks

We test whether an attacker can learn from watermarked responses and generate new text that the detector accepts as watermarked, without access to the secret key.

**Attack baselines.** EXP uses the exponential minimum sampling watermark of Kuditipudi et al. (2023), with key length 256. KGW-2.0 uses the green-list watermark of Kirchenbauer et al. (2023), with green-list fraction $\gamma = 0.25$, logit bias $\delta = 2.0$, and hashing of the previous token. For EXP, random key offsets are disabled in the stealing and substitution experiments and enabled in the black-box detection experiments.

**Attack setup.** We evaluate watermark-stealing attacks following Jovanović et al. (2024) on Qwen3-0.6B-Base, keeping one watermarking key fixed per scheme. Responses contain 400 tokens, generated from 50-token C4 RealNewsLike prefixes at temperature 1 with the full vocabulary. Attacker queries use training-split documents, while evaluation prompts use disjoint validation-split documents.

For each query budget $N$, the attacker collects $N$ watermarked responses and generates $N$ unwatermarked responses to estimate the watermark's token preferences. We consider budgets from 1,000 to 30,000 watermarked queries and generate 500 spoofed texts per attack configuration.

**Attack variants.** We evaluate four variants that estimate token preferences from the preceding one, two, three, or four tokens. A fifth variant conditions on token position modulo the watermark's period. The attacker uses these estimates to boost token logits according to the scoring rule of Jovanović et al. (2024), with clipping value 2 and minimum count 2.

At 10,000 watermarked queries, we evaluate boost strengths $\{1, 2, 4.5, 8\}$ for each variant. For PRC, the query-budget sweep uses strength 8 for the four context variants and 4.5 for the position variant, giving 35 distinct attack configurations across the two sweeps.

**Detection thresholds and quality filtering.** For each scheme, the attack calibrates a detection threshold targeting a 0.1% FPR using the same 5,000 unwatermarked texts. For PRC, we also report results using the analytical Hoeffding threshold from Theorem 3.3.

Increasing the attack strength can produce text that triggers watermark detection but contains token choices the language model considers unlikely. We therefore count a spoof as successful only if it passes both watermark detection and a perplexity filter, similar to the approach of Jovanović et al. (2024). We require perplexity under Qwen3-4B-Base to be at most 136, the 95th percentile on the calibration texts.

The attack success rate is the fraction of all 500 generated attempts that pass both checks, including attempts rejected by the filter in the denominator.

Table 14: **PRC has the lowest spoofing success in the evaluated attacks.** Results after 30,000 watermarked queries on Qwen3-0.6B-Base, using 400-token responses and one key per scheme. Each row shows the most successful tested setting for that scheme under our adapted attack. A successful spoof must trigger detection at a threshold calibrated to target a 0.1% false-positive rate and pass the perplexity filter. Rates are calculated over all 500 attempts (lower is better). EXP uses no random key offset.

| Watermark | Attack uses | Strength | Successful spoofs | Rate |
|---|---|---|---|---|
| **PRC** | Previous 3 tokens | 8 | **1/500** | **0.2%** |
| KGW-2.0 | Previous token | 4.5 | 500/500 | 100.0% |
| EXP | Token position | 2 | 492/500 | 98.4% |
| SynthID-Text | Previous 3 tokens | 8 | 221/500 | 44.2% |

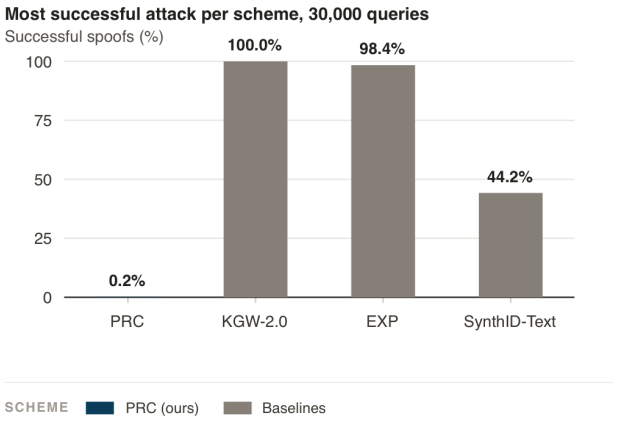



Figure 7: **Stealing attacks are much less successful against PRC than baselines.** Percentage of all 500 attack attempts that trigger detection and pass the perplexity filter after 30,000 watermarked queries. Each bar shows the most successful tested setting for that scheme under our adaptation of Jovanović et al. (2024), as reported in Table 14.

**Results.** Figure 7 shows that the evaluated watermark-stealing attacks rarely succeed against PRC. After 30,000 watermarked queries, the most successful tested setting for each scheme produces 1 successful spoof out of 500 attempts for PRC (0.2%), compared with 221/500 for SynthID-Text (44.2%), 492/500 for EXP (98.4%), and 500/500 for KGW-2.0 (100%). Table 14 lists the corresponding attack settings. The same empirically calibrated PRC detection threshold identifies 91.0% of 500 genuine watermarked query responses, combining low attack success with useful detection of genuine watermarks.

Our findings align with prior works. We see that KGW-2.0 and EXP can be easily spoofed, whereas SynthID can be spoofed using a higher compute budget. PRC, on the other hand, is highly resistant to this attack and the success rate does not increase with a higher compute budget within the tested settings.

Across all 35 PRC attack configurations, we generate 17,500 ($35 * 500$) responses. The calibrated threshold and the perplexity filter used by the attack flag 15 out of them as watermarked. Using the analytical Hoeffding threshold and the perplexity filter, only one text successfully spoofs the watermark, which is within the FPR bound. The corresponding detection rates on genuinely watermarked responses are 91.0% and 83.8%, respectively. These results support resistance to the evaluated stealing attacks for the tested key and model.

### D.13 Robustness to random token substitutions

We evaluate robustness to uniformly random token substitutions following Kuditipudi et al. (2023). Watermarked and unwatermarked responses are generated from Qwen3-0.6B-Base at temperature 1, with output lengths $T \in \{400, 4096\}$. PRC uses the fixed construction with $n = T$, $t = 3$, and $\eta = 0.05$. For both lengths, we evaluate substitution rates $\{0, 0.05, 0.10, 0.15, 0.20, 0.30\}$; the 400-token experiments additionally evaluate a rate of $0.25$. For shared prompts at each length, substitution positions and replacement tokens are matched across schemes.

Each scheme uses a threshold targeting a nominal FPR of $10^{-3}$. For PRC, we use the posterior-mean detector with the analytical Hoeffding threshold from Theorem 3.3. For the 400-token EXP, KGW-2.0, and SynthID-Text experiments, we use empirical $p$-values computed against a reference set of 4,483 human-written C4 continuations. For the 4096-token experiments, these baselines use analytical $p$-values. The observed FPR on model-generated text can differ from the nominal target, so we also report false positives on unwatermarked responses.

Table 15: **PRC is more robust to token substitutions in longer texts.** True-positive rates (%) for Qwen3-0.6B-Base at thresholds targeting a nominal FPR of $10^{-3}$. PRC uses the fixed construction with $n = T$, $t = 3$, and $\eta = 0.05$. At each substitution rate, the 4096-token experiments use 500 prompts for PRC and 200 prompts for each baseline. The 400-token experiments use 500 prompts for every scheme at each substitution rate. Each prompt contributes one watermarked and one unwatermarked response. Subscripts and superscripts in panel (a) give the lower and upper bounds of 95% Wilson confidence intervals. For shared prompts at each length, substitution positions and replacement tokens are matched across schemes.

*(a) 4096-token texts*

| Substitution rate | PRC | EXP | KGW-2.0 | SynthID-Text |
|---|---|---|---|---|
| 0% | $100.0^{100}_{99.2}$ | $100.0^{100}_{98.1}$ | $100.0^{100}_{98.1}$ | $100.0^{100}_{98.1}$ |
| 5% | $99.6^{99.9}_{98.6}$ | $100.0^{100}_{98.1}$ | $100.0^{100}_{98.1}$ | $100.0^{100}_{98.1}$ |
| 10% | $98.8^{99.4}_{97.4}$ | $100.0^{100}_{98.1}$ | $100.0^{100}_{98.1}$ | $99.5^{99.9}_{97.2}$ |
| 15% | $98.6^{99.3}_{97.1}$ | $100.0^{100}_{98.1}$ | $100.0^{100}_{98.1}$ | $99.5^{99.9}_{97.2}$ |
| 20% | $98.2^{99.1}_{96.6}$ | $100.0^{100}_{98.1}$ | $100.0^{100}_{98.1}$ | $99.0^{99.7}_{96.4}$ |
| 30% | $94.6^{96.3}_{92.3}$ | $100.0^{100}_{98.1}$ | $100.0^{100}_{98.1}$ | $99.0^{99.7}_{96.4}$ |

*(b) 400-token texts*

| Substitution rate | PRC | EXP | KGW-2.0 | SynthID-Text |
|---|---|---|---|---|
| 0% | 86.0 | 100.0 | 100.0 | 100.0 |
| 5% | 64.6 | 100.0 | 100.0 | 100.0 |
| 10% | 41.6 | 100.0 | 100.0 | 99.8 |
| 15% | 23.6 | 100.0 | 100.0 | 99.8 |
| 20% | 13.2 | 100.0 | 100.0 | 99.2 |
| 30% | 2.6 | 100.0 | 100.0 | 89.2 |

For long outputs, PRC degrades very little even with high substitution probabilities. However, for short outputs (400 tokens), PRC detection decreases substantially as the substitution rate increases. For example, PRC retains a detection rate of 94.6% at 30% substitutions for 4096-token outputs. For 400-token outputs, its detection rate decreases from 86.0% without substitutions to 2.6% at 30% substitutions. The baselines retain higher detection rates under their respective thresholds.

For the 400-token experiments, each scheme is evaluated on 500 unwatermarked responses per substitution rate. PRC and KGW-2.0 produce no false positives. EXP produces one false positive at 15% substitutions and none at the other rates. SynthID-Text produces between one and six false positives per rate, corresponding to observed FPRs of 0.2–1.2%.

For the 4096-token experiments, PRC produces no false positives among 500 unwatermarked responses at each rate. Each baseline is evaluated on 200 unwatermarked responses per rate. EXP

produces one false positive at 30% substitutions (observed FPR 0.5%) and none at the other rates. KGW-2.0 produces two false positives at 0% and 5% substitutions and one at each remaining rate (observed FPRs 0.5–1.0%). SynthID-Text produces one false positive at each of 10% and 15% substitutions (observed FPR 0.5%) and none at the other rates.

### D.14 Examples of watermarked responses

We present paired watermarked and unwatermarked responses to questions from benchmarks ARC-Easy and GSM8K, complementing the aggregate results in Table 2. We use Qwen3-0.6B with its chat template and thinking enabled, temperature 1, and no top-$k$ or top-$p$ filtering. The PRC uses block length $n = 800$, parity-check weight $t = 3$, noise rate $\eta = 0.10$, and $r = 792$ parity checks. The benchmark runs allow up to 1,900 generated tokens for ARC-Easy and 4,096 for GSM8K, stopping earlier if an end-of-sequence token is produced.

**Prompt**

Which piece of safety equipment is used to keep mold spores from entering the respiratory system?
A. safety goggles B. breathing mask C. rubber gloves D. lead apron

**Watermarked response**

<think> Okay, let's see. The question is asking which safety equipment is used to keep mold spores from entering the respiratory system. The options are A. safety goggles, B. breathing mask, C. rubber gloves, D. lead apron.
First, I need to recall what safety equipment is relevant here. Mold spores are known as mold contaminants. Since the question is about keeping them out of the respiratory system, the option that directly involves preventing exposure is probably related to respiratory protection.
Safety goggles (option A) are typically used to protect the face from debris and other contaminants. They don't specifically prevent mold spores from entering the nose or mouth. Breathing masks (B) are used for air protection, like in medical situations, but maybe they are for air filtration rather than stopping spores from entering the respiratory system.
Rubber gloves (C) and lead aprons (D) would probably be used for other purposes, like in a chemical lab or cleaning, but they don't address mold spores. Mold spores are microorganisms that emit spores, so prevention would involve something that physically blocks them from inhalation.
So, the correct answer here must be something that directly prevents inhalation, which might be a respirator or a similar device. But looking at the options, none of them seem to have a respirator. However, breathing masks could be part of a respirator system. But since the options are given as separate choices, I need to check which one fits better.
Wait, the question says "used to keep mold spores from entering the respiratory system." Mold spores can cause respiratory issues, but the safety equipment that specifically blocks them would be something like a respirator. However, the options given don't include a respirator. The closest might be breathing mask. But maybe the answer is B. breathing mask. Although breathing mask is protection from inhaling harmful gases, but the question is specifically about mold spores. Are there any safety equipment that specifically tackle mold spores? I'm not sure. Maybe the answer is B, breathing mask, because it's used in scenarios where materials are used in air, like in paint or masonry. Alternatively, safety goggles are for other types of contaminants.
Hmm, perhaps I'm overcomplicating. In many cases, a breathing mask is used for air purification, but the answer here might be B. Alternatively, there's no answer here. But according to the options given, B. breathing mask would be the logical choice. Wait, but sometimes tools used for breathing might include masks. However, given the options, I think the correct answer is B. breathing mask. </think>
B. breathing mask
The correct answer is $\boxed{B}$ because the breathing mask is specifically designed to protect the respiratory system from inhaling harmful airborne contaminants, including mold spores.

**Unwatermarked response**

<think> Okay, let's see. The question is asking which safety equipment keeps mold spores from entering the respiratory system. The options are A through D.
First, I need to recall what equipment is used for preventing airborne particles from entering the body, especially respiratory areas. Mold spores are usually inhaled, so if the equipment stops those spores from being inhaled, that's useful.
Safety goggles (Option A) are for protection against eye injuries, not from spores. That's a bit different.

A breathing mask (Option B) is used for getting air into the lungs, which is for breathing, not preventing spores. Maybe not the right one.
Rubber gloves (Option C) protect the hands from hazards, but mold could stay on surfaces in the air, not directly prevent spores from entering the air. So maybe not.
Lead apron (Option D) covers the eyes and arms. But mold is a toxic spore, so that might help block airflow. Wait, but lead apron is a type of emergency equipment, so maybe that's a safeguard.
Wait, maybe the right answer is B, breathing mask. Because breathing masks are used for scenarios where someone is using a mask to breathe, which could help avoid inhaling harmful materials. But mold spores are in the air. If someone is using a breathing mask, they are breathing in, but does that prevent mold spores from entering the respiratory system? It might reduce the likelihood but isn't the main tool.
Alternatively, maybe the answer is D, lead apron. Since lead aprons are designed to restrict air flow, might that help? Hmm, I'm a bit confused here.
Wait, maybe the correct answer is B, breathing mask, because it keeps the person from inhaling harmful stuff, like mold spores. So the act of breathing through it would prevent mold spores from entering the lungs. So the answer would be B.
But I need to confirm. Let me think again. Safety equipment for respiratory protection. So yes, a breathing mask is indeed used to keep the person from inhaling harmful substances, including mold spores. So B is the correct answer. `</think>`
B. breathing mask
The correct option is $\boxed{B}$.

## GSM-8K

**Prompt**

Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

**Watermarked response**

`<think>` Okay, let's see. Janet's ducks lay 16 eggs per day. She eats three for breakfast and bakes muffins with four, right? Then she sells the rest for $2 each. I need to figure out how much she makes at the market each day.
First, let me break down the problem. Each day, she has the ducks laying eggs. So, the eggs per day is 16. But then she has to eat some and have some left for baking. The question says she eats three for breakfast and bakes four. So, how many eggs does she consume and leave for the market?
Wait, so she eats three eggs each morning for breakfast, and then she bakes muffins, which would probably mean she uses some eggs for that. But the problem says she eats three for breakfast every morning, and bakes muffins every day with four. Hmm, does "bakes muffins with four" mean that she uses four eggs for the muffins, leaving some? Or is it that she eats three eggs and then bakes four?
Let me re-read the problem: "She eats three for breakfast every morning and bakes muffins for her friends every day with four." Maybe "bakes muffins for her friends every day with four" is asking that she uses four eggs each day for muffins, leaving some? But then she eats three eggs for breakfast. Then, the remaining eggs would be 16 - 3 - 4, which would be 9. Then selling those 9 eggs at $2 per unit?
Wait, but the problem says "sell the remainder at the farmers' market daily for $2 per fresh duck egg." So, the remainder would be after she eats three eggs and uses four eggs for baking. So 16 - 3 - 4 = 9 eggs. Then selling them for $2 each. So 9 times $2 would be 18 dollars? That seems straightforward.
But let me check again. If she eats 3 eggs for breakfast, that's 3 eggs. She bakes muffins using 4 eggs, so she spends 4 eggs. So total eggs used each day: 3 + 4 = 7. Then remaining eggs: 16 - 7 = 9. Then selling 9 eggs at $2 each, which is 9 * 2 = $18. So the answer is $18?
Yes, that seems right. The key is figuring out how many eggs she uses each day. So, 3 for breakfast, 4 for muffins. 3+4 is 7. 16-7 is 9. Multiply by $2 per egg. Therefore, $18. I think that's the correct answer.
`</think>`
18

**Unwatermarked response**

`<think>` Okay, let's see. Janet's ducks lay 16 eggs per day. She eats three for breakfast and bakes four muffins daily. Then she sells the rest at $2 per egg. I need to find how much she makes at the market every day. Hmm, first, I should figure out how many eggs she uses for breakfast and muffins, then subtract that from the total eggs per day, and then multiply by the market price per egg.

So, total eggs per day: 16. Breakfast uses three, muffins use four. Let me add those. 3 + 4 equals 7. So, she eats 7 eggs each day. Then, she has 16 minus 7, which is 9 eggs left for the market. Therefore, she makes 9 eggs multiplied by $2 per egg. That should be 18 dollars. Let me check that again.
Total eggs: 16. Used: 3 + 4 = 7. Leftover: 16 - 7 = 9. Sale price: $2 per egg. 9 * 2 = $18. Yeah, that seems right. I don't think I made any mistakes here. So the answer is $18. `</think>`
18