WATERMARKING LANGUAGE MODELS WITH ERROR CORRECTING CODES

Patrick Chao, Yan Sun, Edgar Dobriban & Hamed Hassani
University of Pennsylvania
{pchao, sunyan1, dobriban}@wharton.upenn.edu, hassani@seas.upenn.edu

ABSTRACT

Recent progress in large language models enables the creation of realistic machine-generated content. Watermarking is a promising approach to distinguish machine-generated text from human text, embedding statistical signals in the output that are ideally undetectable to humans. We propose a watermarking framework that encodes such signals through an error correcting code. Our method, termed robust binary code (RBC) watermark, introduces no noticeable degradation in quality. We evaluate our watermark on base and instruction fine-tuned models and find our watermark is robust to edits, deletions, and translations. We provide an information-theoretic perspective on watermarking, a powerful statistical test for detection and for generating p-values, and theoretical guarantees. Our empirical findings suggest our watermark is fast, powerful, and robust, comparing favorably to the state-of-the-art.

1 INTRODUCTION

As language model capabilities improve, there are corresponding potential harms such as the creation of misinformation [28] and propaganda [25]. To mitigate this, a first step is to detect and filter content. A popular approach to reliably detecting AI generated content is to add a *watermark* [17; 18; 1; 9], a hidden signal embedded in the output. While there are exponentially many combinations of words and characters, watermarking biases generation towards specific patterns that are undetectable to humans.

We consider the detection setting from the *model-provider*'s perspective: the detection algorithm receives (user or machine-generated) text as input, but no further metadata such as prompts or generation parameters. We do not explore zero-shot or post-hoc methods to classify text as generated from *any* language model, such as GPT-Zero [26] and DetectGPT [20]. This model-agnostic detection is inherently challenging as language models are trained to mimic human text [5]. as language models become more powerful, their generation becomes nearly indistinguishable from human-generated text, thereby reducing the efficacy of zero-shot classification methods. Instead, we explore how to introduce statistical signals into the generation process to reliably classify text as watermarked.

Desiderata. We focus on the following list of practical, empirical and algorithmic desiderata (and some others) for an effective watermark, inspired by, e.g., [18; 17; 21].

- *Robust:* The watermark should be robust to watermarking attacks and perturbations, such as edits and deletions of significant fractions of the text, translation attacks, paraphrasing, etc.
- *Quality-Preserving:* The watermark should not decrease the quality of the outputs of the language model.
- *Model Agnostic:* The watermark should apply to any language model. This is important to be applicable to future models.
- *High Detection Power:* The watermark should be detectable with only a small number of tokens.

Prior watermarking schemes achieve many but not all of these qualities. For instance, Kuditipudi et al. [18] introduces *distortion-free* watermarks that do not modify the output distribution in expectation (but their detection power is limited due to using permutation tests). Taking a cryptographic perspective, Christ et al. [9] introduces theoretically *undetectable* watermarks (but does not illustrate their performance empirically). [22] develop a provably robust watermark via error correction codes, focusing on embedding a multi-bit string (such as an user ID) in the text. Our methods (binarization, correlated binary channel, LDPC codes) and our theoretical guarantees (high detection probability), are different. We obtain highly powerful watermarks without sacrificing generation quality. For further discussions on prior and related work, see Appendix A.

Contributions. We propose the ***R****obust* ***B****inary* ***C****ode (RBC) watermark* using error correcting codes, and argue the following claims:

- **Robustness.** RBC achieves several forms of robustness (e.g., to user edits) by leveraging error correcting codes.
- **Power and Quality.** Our RBC watermark has theoretical guarantees on correct decoding. Further, RBC empirically keeping the quality of the original language model almost unchanged, and leads to strongly detectable watermarks, competitive with state-of-the-art watermarks that introduce a larger distortion.
- **Simplicity.** Our RBC watermark is simple to use, as it wraps around the output logits of the language model.

2 Background

2.1 Error Correcting Codes

To introduce our method, we need to provide some background on error correcting codes. For a positive integer k, we use the Hamming distance d_H on $\{0,1\}^k$, such that for $u, v \in \{0,1\}^k$, $d_H(u, v)$ is the number of differing coordinates of u, v. We recall some standard definitions [19; 14].

Definition 2.1. *For positive integers $k \leq n$, an* error correcting *code (ECC) is an injective map $\mathcal{C} : \{0,1\}^k \to \{0,1\}^n$. The* message space *is $\{0,1\}^k$. Applying $\boldsymbol{C}$ to a message m is known as* encoding*, and $\boldsymbol{C}(m)$ is known as a* codeword*. The* rate *of the code is k/n.*

The error correcting distance *of $\mathcal{C}$ is the greatest $t > 0$ such that for all messages $m \in \{0,1\}^n$, there exists at most one codeword $c \in \mathcal{C}$ in the Hamming ball of radius of t around m, $d_H(m, c) \leq t$. Such a code $\mathcal{C}$ is known as a $[n, k, 2t+1]$-code.*

Given a $[n, k, 2t+1]$-error correcting code $\mathcal{C}$, we may define a *decoding* map $\mathcal{C}^{-1}$. For a given $m \in \{0,1\}^n$, the decoding $\mathcal{C}^{-1}(m)$ is the unique $c \in \mathcal{C}$ such that $d_H(w, c) \leq t$, if such a c exists; and otherwise, the decoding is defined arbitrarily. By definition, at most one such c exists.

2.2 Notation

We use the following notation throughout this work. We typically use capital letters to denote random variables. For a positive integer v and $p \in [0,1]$, let $F_B(t; v, p)$ be the CDF of a $\mathrm{Binomial}(v, p)$ random variable at the positive integer t, and let $[n] = \{1, \ldots, n\}$. Let $\boldsymbol{V}$ be the vocabulary of the language model, with $|\boldsymbol{V}|$ commonly between 50,000 to 200,000 in our current applications, and let $\boldsymbol{V}^*$ be the set of strings of tokens from $\boldsymbol{V}$ of arbitrary length. For a positive integer i, we use X_i to denote the tokens generated by the language model $p : \boldsymbol{V}^* \to \Delta(\boldsymbol{V})$, and write $X_i \sim p(\cdot \mid x_{1:i-1})$ for any sequence $x_{1:i-1} \in \boldsymbol{V}^{i-1}$ of previous tokens. Let $p_j(\cdot \mid x_{1:i-1})$ be the distribution of the next j tokens. For some positive integers k, n, t, Let $\boldsymbol{C} : \{0,1\}^k \to \{0,1\}^n$ be a $[n, k, 2t+1]$ error correcting code.

2.3 Reduction to Binary Vocabulary

Motivated by Christ et al. [9], we reduce language modeling to a minimal binary vocabulary. To do so, we assign to each token a unique bit string. For a vocabulary of size $|\boldsymbol{V}|$, this requires $\ell = \lceil \log_2 |\boldsymbol{V}| \rceil$ total bits; typically between 16 and 18 bits in our current applications. We use an injective binary encoder $\boldsymbol{E} : \boldsymbol{V} \to \{0,1\}^\ell$ that maps tokens to bit strings. A single token sampled from the language model $p(\cdot|x_{1:(a-1)})$ induces a distribution over the next ℓ bits, determined by the encoder $\boldsymbol{E}$. We use $q_{n+1} := p_{\mathrm{bin}}(B_{n+1} = 1 \mid b_{1:n})$ to denote the probability of the next bit being one ("1") given previous bits $b_{1:n}$. We will write $p := p_{\mathrm{bin}}$ from now on.

In our watermarking scheme, we use the following variables: $M_i \in \{0,1\}^k$ is the message passed through the error correcting code to obtain the codeword $Y_i = \boldsymbol{C}(M_i) \in \{0,1\}^n$. The binary sampling scheme generates bits $(B_1, \ldots, B_n)$ that are correlated with the Y_is (as explained in

Section 3.1). We use the previous tokens to modify the distribution of the following tokens. Each application of our watermarking scheme generates a block of $w_{\text{out}} > 0$ tokens, using the previous $w_{\text{in}} > 0$ tokens. This is related to prior approaches in neuro-linguistic steganography [29; 10], which use minimum-entropy couplings for correlated sampling between messages and covertext [10].

3 Key Watermarking Tool

For our watermarking scheme, we systematically encode signals into generated tokens. However, doing so by biasing the logits as in [17] perturbs the distribution of tokens, resulting in distorted generation for each next token. Instead, we propose a generation technique to encode signals that is *distortion-free given each previous block of* $w_{\text{in}} > 0$ *tokens*. Of course, this technique is still not distortion-free over the entire generation (see Appendix A.1 for discussion of security implications), but empirically the degradation in performance is low. As described before, to introduce this technique, we first reduce from an arbitrary vocabulary to a binary vocabulary, where the language model generates bits. Recall that $q_i = p(B_i = 1 \mid b_{1:(i-1)})$.

Given a message $M \in \{0,1\}^k$ that we wish to transmit, we generate a codeword $Y = \boldsymbol{C}(M) \in \{0,1\}^n$. We next explain how to embed this codeword without inducing distortion.

3.1 Correlated Binary Sampling Channel (CBSC)

To introduce our method, we first take a step back and consider the setting of sampling Bernoulli random variables. An equivalent way to sample $B \sim \text{Bern}(q)$, $q \in [0,1]$ is to introduce an auxiliary random variable $U' \sim \text{Unif}[0,1]$ and let $B = 1\{U' \leq q\}$. Notably, U' and B are now correlated.

A further equivalent sampling scheme is to let $U' = (1-Y)/2 + U/2$, where $Y \sim \text{Bern}(1/2)$ and $U \sim \text{Unif}[0,1]$ are sampled independently; so that $U' \sim \text{Unif}[0,1]$. By writing U' in binary, $1 - Y$ represents the most significant bit of U', and U represents the remaining bits. Therefore, we have the following sampling scheme for B:

$$
\begin{aligned}
&Y \sim \text{Bern}(1/2),\ U \sim \text{Unif}[0,1],\ Y \perp\!\!\!\perp U, \\
&B = \mathbb{1}\{((1-C) + U)/2 \leq q\}.
\end{aligned}
\tag{1}
$$

We call this sampling process the *Correlated Binary Sampling Channel* (CBSC), where we use an auxiliary random variable $Y \sim \text{Bern}(1/2)$ to sample a biased (non-uniform) bit $B \sim \text{Bern}(q)$, ensuring (B, Y) are correlated. We use $1 - Y$ rather than simply Y so that B and Y are *positively correlated*; but the methods are equivalent. This CBSC is equivalent to the binary generation scheme from the recent work [8], although with a different formulation and motivation.

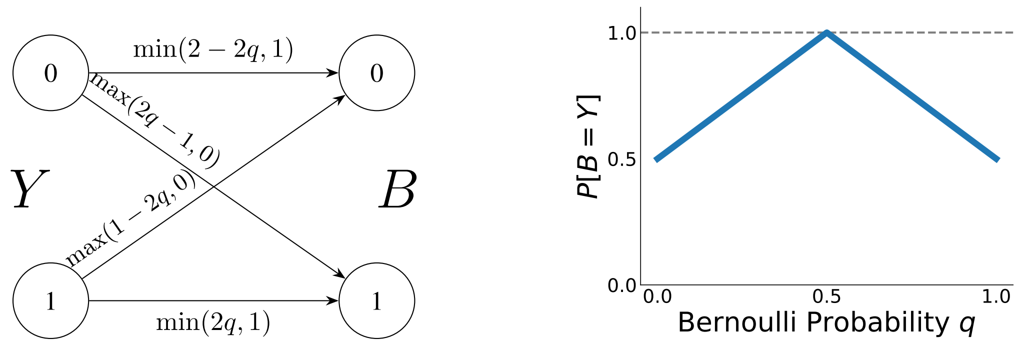
Next-token distortion-freeness. Crucially, the CBSC ensures that B matches its target distribution despite using the external information Y, i.e., $B \sim \text{Bern}(q)$. Thus, the CBSC does not distort the distribution of each token given the previous block of $w_{\text{in}} > 0$ tokens.

We plot the transition probabilities between Y and B in Figure 1a. We also show the matching probability $\mathbb{P}[Y = B] = 1 - 2|1/2 - q|$ between B and Y as a function of the probability q in Figure 1b. When the entropy of B is maximized, i.e., when $q = 1/2$, then we deterministically have $Y = B$. However, when $q \in \{0,1\}$—so B has zero entropy—then $\mathbb{P}[Y = B] = 1/2$ and Y contains no information about B. In general, the sampled bit B is *biased towards* Y and is more likely to be equal to Y than not.

3.2 Simple Watermark

The CBSC in Equation (1) is our key mechanism for watermarking, enabling next-token distortion-free sampling with detectable correlation. We embed a sequence of *known* bits $Y = (Y_1, \ldots, Y_n)$ into our generated bits $B = (B_1, \ldots, B_n)$. Generating bits B through the CBSC ensures B follows the original probability distribution–i.e., is next-token distortion-free—and yet biases B towards Y.

We may use this statistical dependence to detect whether or not the text was watermarked. For unwatermarked text, B and Y are uncorrelated and therefore $d_H(B, Y) \approx n/2$, whereas for watermarked text, we expect $d_H(B, Y) < n/2$. This is the basis of our simple *one-to-one watermarking scheme*, described in Appendix B.4. We will later improve this by reducing errors in our binary channel by leveraging error correcting codes.



(a) Binary channel $Y \to B$, s.t. $B \sim \mathrm{Bern}(q)$, $q \in [0,1]$.

(b) Probability that $B = Y$, equal to $1 - 2|1/2 - q|$.

Figure 1: Correlated binary sampling channel (CBSC) with $Y \sim \mathrm{Bern}(1/2)$ and $B \sim \mathrm{Bern}(q)$.

Choosing known bits Y. In order to reliably detect the watermark, we need to know the bits Y used in the CBSC. One approach is to use a long fixed bit string, a *secret sequence* known to the language model provider. Kuditipudi et al. [18] use a similar approach with a fixed list of real numbers and use the edit distance to capture correlations. However, this approach is sensitive to the relative indexing of the text, and the fixed list and can be susceptible to attacks that shuffle the text. Instead, we use a sliding window of previously generated tokens to serve as sources of randomness for Y. This sliding window approach is popular in watermarking schemes, see e.g., [1; 17; 9]. We thus use *text-dependent* randomness [21] to generate pseudorandom bits Y, leveraging previous tokens to obtain pseudorandom outputs.

Error correcting codes in generation. The purpose of using error correcting codes (ECCs) is to reduce mismatches between B and Y. Rather than directly transmitting $Y = (Y_1, \dots, Y_n)$, we add *redundancy* through a $[n, k, 2t+1]$ ECC $\boldsymbol{C}$. We choose a shorter message $M \in \{0,1\}^k$ and use $Y = \boldsymbol{C}(M) \in \{0,1\}^n$. When using an ECC, we choose M by hashing the previous bits, and then apply the error correcting code to M to obtain our codeword Y. By using $\boldsymbol{C}$, we can correct t errors in the CBSC. Specifically, if $d_H(Y, B) \leq t$, then $\mathcal{C}^{-1}(B) = M$, giving a strong indication that the text was in fact watermarked.

Combining these parts, our watermarking scheme contains the following steps:

1. **Create message:** Use previous tokens to create message $M \in \{0,1\}^k$.
2. **Encode using $\boldsymbol{C}$:** Encode M using ECC to obtain $Y = \boldsymbol{C}(M) \in \{0,1\}^n$.
3. **Generate:** Transmit Y using CBSC with probabilities from the language model to generate $B \in \{0,1\}^n$.
4. **Repeat:** Perform the above steps until the stopping condition.

We call each sequence of n bits generated at an iteration a *block*. In Algorithm 1 and Algorithm 2, we present a simplified watermark and detection algorithm that captures the essence of our approach.

Algorithm 1: Simplified RBC Watermark

Input: Generation length N, binary language model p_{bin}, ECC $\boldsymbol{C} : \{0,1\}^k \to \{0,1\}^n$

$X_{1:k} \sim p_{\text{bin},k}(\cdot)$ ▷ Initialize first k bits
$i \leftarrow k$
while $i \leq N$ **do**
  $M \leftarrow X_{(i-k+1):i} \in \{0,1\}^k$ ▷ Message
  $Y \leftarrow \mathcal{C}(M) \in \{0,1\}^n$ ▷ Codeword
  for $j = 1$ to n **do**
    $U_j \sim \text{Unif}[0,1]$ ▷ CBSC
    $X_{i+j} \leftarrow \mathbb{1}\left\{\frac{1-Y_j+U_j}{2} \leq p_{\text{bin}}(\cdot \mid X_{1:i+j-1})\right\}$
  $i \leftarrow i + n$
return $X_{1:N}$

Algorithm 2: Simplified Detection

Input: Bits $X_{1:N}$, threshold T

for $i = 1$ to $N - n - k + 1$ **do**
  $M \leftarrow X_{i:(i+k-1)}$
  $B \leftarrow X_{(i+k):(i+n+k-1)}$
  $\hat{M} \leftarrow \mathcal{C}^{-1}(B)$ ▷ Recovered message
  $Z_i \leftarrow k - d_H(\hat{M}, M)$ ▷ Matches
if $\sum Z_i > T$ **then**
  return WATERMARKED
else
  return NOT WATERMARKED

4 Robust Binary Code Watermark

Next, we present our robust binary code (RBC) watermark, which converts tokens to binary, applies an error correcting code for robustness, and generates bits through the CBSC.

Let $\ell = \lceil \log_2 |\boldsymbol{V}| \rceil$ be the number of bits for each token and let $w_{\text{in}} = \lceil k/\ell \rceil$ and $w_{\text{out}} = \lceil n/\ell \rceil$. Assume we have an injective *binary encoder* $\mathcal{E} : \boldsymbol{V} \to \{0,1\}^\ell$ which maps from tokens to bit strings, and the corresponding binary decoder $\mathcal{E}^{-1} : \{0,1\}^\ell \to \boldsymbol{V}$ (defined arbitrarily for strings outside of the image of $\mathcal{E}$). We also have the language model p_{bin} induced by p over a binary alphabet, so that $p_{\text{bin}}(s_1, \ldots, s_n) := p_{\text{bin}}(S_{n+1} = 1 | s_1, \ldots, s_n)$ for any binary string $s_1, \ldots, s_n$. We let the encoder operate on a list of tokens elementwise, and the decoder operate on blocks of bits of length ℓ.

Our full watermarking scheme is shown in Algorithms 3, 4, 5 and 6. Algorithms 3 and 4 expand on the simplified method from Algorithm 1, and Algorithms 5 and 6 expand on the detection in Algorithm 2. The RBC watermark is very similar to the simplified watermark in Section 3.2, with the addition of the binary encoder/decoder $\boldsymbol{E}$. In the generation of the messages M, we also apply an exclusive or between the previous bits with a randomly chosen bitstring $R \in \{0,1\}^k$ to ensure the message M contains i.i.d. $\text{Bern}(1/2)$ entries. In this step, we could also apply the minimum hashing method from Kirchenbauer et al. [17].

Algorithm 3: RBC Watermarking

Input: Total token generation length N, window widths w_{in} and w_{out}, unif. random binary string $R \in \{0,1\}^k$

$X_1, \ldots, X_{w_{\text{in}}} \sim p_{w_{\text{in}}}(\cdot)$
$i \leftarrow w_{\text{in}} + 1$
while $i \leq N$ **do**
  $M \leftarrow \boldsymbol{E}(X_{(i-w_{\text{in}}):(i-1)})_{1:k} \oplus r \in \{0,1\}^k$
  $Y \leftarrow \mathcal{C}(M) \in \{0,1\}^n$
  $X_{i:(i+w_{\text{out}}-1)} \leftarrow \texttt{GenBlock}(X_{1:(i-1)}, Y)$
  $i \leftarrow i + w_{\text{out}}$
return X

Algorithm 4: GenBlock

Input: Previous text $X \in \boldsymbol{V}^*$, Codeword $Y \in \{0,1\}^n$, output tokens w_{out}

for $j = 1$ to n **do**
  $U_j \sim \text{Unif}[0,1]$
  $B_j \leftarrow \mathbb{1}\{(1-Y_j+U_j)/2 \leq p_{\text{bin}}(X, B_{1:(j-1)})\}$
for $j = n+1$ to $w_{\text{out}} \cdot \ell$ **do**
  $U_j \sim \text{Unif}[0,1]$
  $B_j \leftarrow \mathbb{1}\{U_j \leq p_{\text{bin}}(X, B_{1:(j-1)})\}$
$B \leftarrow (B_1, \ldots, B_{w_{\text{out}} \cdot \ell})$
return $\mathcal{E}^{-1}(B) \in \boldsymbol{V}^{w_{\text{out}}}$

5 Theoretical Results

We now provide theoretical guarantees for our methods, focusing on how the entropy of the language model affects watermarking. For the following results, we consider a language model with a binary vocabulary that generates bits $B_1, \ldots, B_n$. Let $q_i = \mathbb{P}[B_i = 1 \mid b_{1:i-1}] = p_{\text{bin}}(1 \mid b_{1:i-1})$ for any positive integer i and bit string $b_{1:i-1}$. We provide the proofs of all of the following claims in Appendix C. We start with some preliminaries.

Definition 5.1 (Entropy). *Let the entropy of a Bernoulli random variable Y with success probability $q \in [0,1]$ be defined as $H(Y) := H(q) = -q\log_2 q - (1-q)\log_2(1-q)$, with $H(0) = H(1) = 0$. For a sequence of (possibly dependent) Bernoulli random variables $Y_1, \ldots, Y_n$ with respective success probabilities $q_1, \ldots, q_n$, let the average entropy be $\bar{H}(Y_1, \ldots, Y_n) := \bar{H}(q_1, \ldots, q_n) := 1/n \cdot \sum_{i=1}^n H(Y_i)$.*

We will need the following lemma, which bounds the entropy $H(q)$ in terms of $|1/2 - q|$, $q \in [0,1]$.

Lemma 5.2. *For $q \in [0,1]$, we have the inequality $1 - 2|1/2 - q| \le H(q) \le \sqrt{1 - 4|1/2 - q|^2}$. For $q_1, \ldots, q_n \in [0,1]$, we have the inequality*

$$\frac{1}{n}\sum_{i=1}^n [1 - 2|1/2 - q_i|] \le \bar{H}(q_1, \ldots, q_n) \le \sqrt{1 - 4\Big(\sum_{i=1}^n |1/2 - q_i|/n\Big)^2}. \tag{2}$$

We provide a plot of the bounds on the entropy for the single-variate case in Figure 3 in Appendix. Next, using the above result, we turn to analyzing the CBSC. With $q_i = \mathbb{P}[B_i = 1 \mid b_{1:i-1}]$ for all i and $b_{1:i-1}$, let the average entropy of the language model be $h := \mathbb{E}_{B,Y}\left[\frac{1}{n}\sum_{i=1}^n H(Q_i)\right]$. Intuitively, we expect that watermarking should be easier when the entropy h is large ($h \approx 1$), as there are more ways to embed signals and keep the distribution unchanged. The next result supports this intuition.

Theorem 5.3 (Bounding the proportion of mismatches in a CSBC with the entropy). *The expected proportion of mismatches between the generated B and codeword Y—i.e., $\mathbb{E}[d_H(B,Y)/n]$—is bounded by*

$$(1-h)/2 \le \mathbb{E}[d_H(B,Y)/n] = n^{-1}\sum_{i=1}^n \mathbb{E}_{B,Y}[|1/2 - Q_i|] \le \sqrt{1-h^2}/2.$$

This theorem bounds the error probability in terms of the average entropy $h = \mathbb{E}_{B,Y}\left[\sum_{i=1}^n H(Q_i)\right]/n$ of the language model, showing that the frequency of errors is both upper and lower bounded by a monotone decreasing function of h. In particular $\mathbb{E}[d_H(B,Y)/n] \to 0$ as $h \to 1$. This is consistent with watermarking being easier when the entropy is high.

The above result shows that the mean of $\Delta := \sum_{i=1}^n |1/2 - Q_i|$ is upper bounded by $n\sqrt{1-h^2}/2$. Therefore, it is reasonable to assume that for some $C > 1$ and a small $\varepsilon_n > 0$, $\Delta \le \kappa n\sqrt{1-h^2}/2$ with probability at least $1 - \varepsilon_n$. Our next result will be presented under this condition.

Theorem 5.4 (Exact block decoding). *Consider a language model such that $q_i = \mathbb{P}\left[1 \mid B_{1:(i-1)}, Y\right]$ for $i \in [n]$. Let the average entropy of the language model be $h = \mathbb{E}_{B,Y}\left[\frac{1}{n}\sum_{i=1}^n H(Q_i)\right]$. Suppose that for some $C > 1$, $\Delta \le \kappa n\sqrt{1-h^2}/2$ with probability at least $1 - \varepsilon_n \in (0,1)$. Assume $\mathcal{C} = [n, k, 2t+1]$ is an error correcting code where $t + 1 \ge \kappa n\sqrt{1-h^2}/2$. Then for a codeword Y, the generated bits $(B_1, \ldots, B_n)$ from Algorithm 4 satisfy*

$$\mathbb{P}\left[\mathcal{C}^{-1}(B_1, \ldots, B_n) = Y\right] \ge 1 - \exp\left[-\left(t + 1 - \kappa n\sqrt{1-h^2}/2\right)^2/n\right] - \varepsilon_n.$$

The above theorem bounds the probability that a specific block *exactly* decodes to the codeword Y. In particular, it gives a bound where the decoding block probability decreases as the entropy of the language model decreases. In particular if $t/n \gg \sqrt{1-h^2}$ and ε_n is small, the decoding happens with high probability. This is consistent with the fact that watermarking is more difficult for models with less entropy when h is small, in which case t/n needs to be large.

6 Experimental Results

We evaluate the performance of RBC in a suite of experiments to measure watermarking detectability and robustness to state-of-the-art watermarking schemes.

6.1 Watermark Detection

Generation procedure. We use ten prompts inspired by [21], encompassing topics such as writing a book report, story, and fake news article. Piet et al. [21] used these text generation tasks to represent realistic settings with long-form generated text, where the model provider may like to watermark the outputs. We provide the full prompts used for generation in Appendix E.

We generate 20 responses for each prompt for seven lengths ranging from 10 to 150 tokens, representing a few sentences to a few paragraphs. We use the Llama-3-8B base model [2], a powerful open-source model capable of high quality generation.

Baselines and hyperparameters. We compare the performance of RBC to the watermarking algorithm from Kirchenbauer et al. [17], which we denote as the *Distribution Shift* method. The thorough evaluation from Piet et al. [21] finds this to be the most effective and powerful watermark, and concludes that it serves as the current state-of-the-art. We also compare with the methods from [13] and [15], two variants of the method [17].

In practice, many ECCs with varying guarantees exist, including [6; 12; 23]. Different ECCs provide different guarantees, e.g., robustness to bit flips or edit distance. For this work, we explore two ECCs. We primarily use the popular low-density parity-check (LDPC) code [12] for our watermarking scheme. As a baseline, we also use a custom code which we call a *one-to-one code*. Our one-to-one code has $k = n$ and is equivalent to the identity function, i.e., the one-to-one codes do not provide error correcting capabilities. We use this as a baseline to compare the impact of the error correcting functionality. For more details on our one-to-one code, see Appendix B.4. We use the LDPC code [24] with $n = 12$, $k = 5$, $d_v = 3$, $d_c = 4$, $p = 0.35$ and a one-to-one code with $n = k = 4$, using $w_{\text{in}} = 1$ and $w_{\text{out}} = 1$, and set $\boldsymbol{E}$ to be a random binary enumeration.

For the distribution shift methods from [17] and [15], we use their suggested hyperparameters. A key parameter is δ, the level of distribution shift induced by the watermarking procedure. Greater values of δ result in larger distribution shifts and more detectable watermarking. The authors of [17] recommend $\delta \in [0.5, 2]$ for watermarking. In our experiments, we choose to use $\delta = 1$ and 2 to represent medium and strong distribution shifts for the method [17]. For the method from [15], a lightweight neural network is trained to predict token-specific δ and γ values, where γ controls the fraction of tokens in the green list. We set the network parameters using the default check point suggested by the authors[1]. [13] introduced two reweighting methods: δ reweighting and γ reweighting to make the watermark unbiased. We compare our approach against both methods[2].

Evaluation. For each generation, we use the p-value of the detection procedure on the output text. We compute the mean of the log-p-values (instead of the p-values themselves) for each generation to more accurately capture the signal strength. We also report the detection probability, i.e., true positive rate or power—the percentage of generations with p-values less than $\alpha = 1\text{e}{-4}$—to represent how likely the text is to be classified as watermarked.

From the summary statistics in Table 1 and Figure 2, we observe that *o*ur RBC watermarking method shows substantial improvements in detectability compared to the baseline methods, when the number of tokens are small. And the p-values of our method are smaller than those of other methods by many orders of magnitude. Further, RBC improves performance by using the LDPC code compared to the one-to-one code, suggesting improvements by using an ECC. We observe consistent detectability with as few as 20-30 tokens.

6.2 Robustness Experiments

In practice, a user may attempt to circumvent watermarks by editing the text generated by a language model. To emulate this, we use four popular perturbations that may represent an adversary hoping to evade a watermark, as in other works including [18; 21].

1. **Delete.** We randomly delete 20% of the tokens.
2. **Swap.** We replace a randomly chosen fraction of 20% of the tokens with random tokens.

[1] `https://github.com/mignonjia/TS_watermark/tree/main/ckpt/llama`

[2] In our experiments, the p-values for unbiased reweigting methods [13] are obtained using the tail bounds from Theorem 9 of [13]

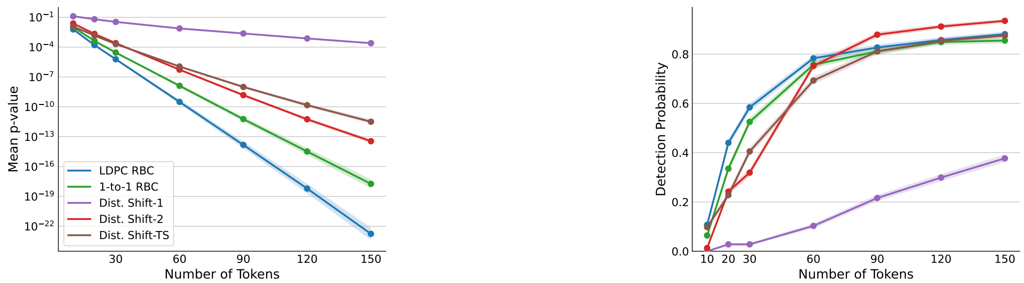


Figure 2: Watermarking performance of the base Llama-3-8B model with RBC using LDPC and one-to-one codes, and the Distribution Shift method from Kirchenbauer et al. [17], averaged over 100 generations for ten prompts. **Left:** The mean log-p-values with standard errors shaded. **Right:** The detection probability with standard errors shaded.

Table 1: Comparison between our watermarking methods and the baselines [17; 15; 13], for 30 and 150 tokens. We report the exponential of mean log p-value, the median p-value, and the percentage of generations with p-values less than $\alpha = 1\text{e}{-4}$. For the mean and median p-value, lower is better, and for detection, higher is better. The best value in each column is **bolded**.

	30 Tokens			150 Tokens		
Method	Mean P	Median P	Detect %	Mean P	Median P	Detect %
LDPC RBC (ours)	**6.0e−6**	**1.4e−5**	**58.5**	**1.7e−23**	**3.5e−20**	88.1
One-to-one RBC (ours)	2.8e−5	6.0e−5	52.5	1.8e−18	1.9e−16	85.5
Dist. Shift $\delta = 1$ [17]	3.4e−2	7.4e−2	20.0	2.4e−4	7.0e−4	72.4
Dist. Shift $\delta = 2$ [17]	2.4e−4	6.4e−4	32.1	2.8e−14	5.6e−14	**94.2**
Dist. Shift Token Specific [15]	2.0e−4	4.3e−4	40.5	3.2e−12	1.3e−11	87.5
Unbiased δ Reweight [13]	1.5e−1	1.0	2.1	2.2e−5	1.2e−4	40.5
Unbiased γ Reweight [13]	1.7e−2	2.6e−2	6.6	1.7e−15	1.1e−14	90.8

3. **Translate.** We translate the generated text to Russian then back to English using Argos Translate [11].
4. **Paraphrase.** We paraphrase the text using Llama-3-8B model, with the prompt "`Paraphrase the following text:`".

For our experiments, we elect to perturb 20% of the tokens, as this represents a relatively high noise regime where one in five tokens are modified. In our translation perturbation, we choose to translate to Russian and then back to English for a powerful attack, as Russian and English are relatively different compared to Spanish and French, see e.g., [3], etc.

In Table 2 and Figure 4, we evaluate the robustness of RBC and the distribution shift watermarking scheme to the perturbations. Notably, **the LDPC and one-to-one RBC watermarks show the greatest robustness**. They achieve consistent detectability with ∼60 tokens. For further plots, see Appendix D.1.

6.3 Instruction Fine-tuned Models

We conduct ablation experiments on the dependence of the language model. Instead of the base Llama-3-8B model, we consider the instruction fine-tuned model, optimized to answer queries. It is well known—see e.g., Piet et al. [21]—that instruction fine-tuned models tend to be more certain of the next-token predictions, and therefore have less entropy and are more difficult to watermark. We use the same generation and hyperparameters as Section 6.1 but with the Llama-3-8B Instruct model, and we use an LDPC code with $n = 8$, $k = 4$, $d_v = 3$, $d_c = 4$, $p = 0.35$ and a one-to-one code with $n = k = 2$ for this lower entropy regime. We plot the mean log p-values in the left plot of Figure 6.

Compared to the base fine-tuned model, the instruction fine-tuned model has substantially lower entropy, and it is more difficult to add consistent statistical signals. Compared to the distribution shift watermark with $\delta = 1$ and $\delta = 2$, our RBC method (both for LDPC and one-to-one) generally has a much larger true detection rates. we see poor detectability and large p-values. The distribution shift watermark with $\delta = 2$ sometimes has similar detection rates to RBC, but is strongly outperformed

Table 2: Comparison between our watermarking methods and the baselines [17; 15; 13], with our four perturbations.

		30 Tokens			150 Tokens		
	Method	Mean P	Median P	Detect %	Mean P	Median P	Detect %
Swap	LDPC RBC (ours)	**2.3e−3**	**6.2e−3**	**19.2**	**3.1e−10**	**1.1e−8**	**70.8**
	One-to-one RBC (ours)	3.8e−3	9.9e−3	15.6	2.4e−8	3.9e−7	65.2
	Dist. Shift $\delta = 1$ [17]	1.1e−1	1.6e−1	0.5	1.1e−2	2.5e−2	9.8
	Dist. Shift $\delta = 2$ [17]	1.0e−2	2.1e−2	7.5	5.9e−7	1.9e−6	69.1
	Dist. Shift Token Specific [15]	8.7e−3	1.8e−2	9.6	9.4e−6	3.2e−5	55.5
	Unbiased δ Reweight [13]	9.5e−1	1.0	0.0	1.0	1.0	0.0
	Unbiased γ Reweight [13]	8.4e−1	1.0	0.1	8.3e−1	1.0	0.0
Delete	LDPC RBC (ours)	**8.6e−4**	**2.2e−3**	**30.2**	**8.8e−13**	**6.5e−11**	77.6
	One-to-one RBC (ours)	1.9e−3	5.2e−3	21.7	3.4e−10	8.6e−9	72.6
	Dist. Shift $\delta = 1$ [17]	8.5e−2	1.4e−1	1.0	5.3e−3	1.3e−2	15.4
	Dist. Shift $\delta = 2$ [17]	5.2e−3	1.2e−2	13.5	1.6e−8	4.3e−8	**81.5**
	Dist. Shift Token Specific [15]	4.9e−3	1.0e−2	12.8	5.2e−7	1.8e−6	68.9
	Unbiased δ Reweight [13]	9.3e−1	1.0	0.0	9.5e−1	1.0	0.0
	Unbiased γ Reweight [13]	7.8e−1	1.0	0.0	4.6e−1	1.0	0.6
Translate	LDPC RBC (ours)	**6.3e−3**	**1.8e−2**	**12.4**	**6.1e−10**	**1.2e−6**	**63.3**
	One-to-one RBC (ours)	9.8e−3	2.1e−2	9.7	1.0e−7	6.5e−6	60.2
	Dist. Shift $\delta = 1$ [17]	1.1e−1	1.9e−1	0.7	1.5e−2	3.3e−2	7.6
	Dist. Shift $\delta = 2$ [17]	1.8e−2	4.2e−2	6.5	1.3e−5	3.1e−5	56.1
	Dist. Shift Token Specific [15]	2.3e−2	5.5e−2	5.4	9.9e−5	4.3e−4	41.4
	Unbiased δ Reweight [13]	1.0	1.0	0.0	1.0	1.0	0.0
	Unbiased γ Reweight [13]	8.1e−1	1.0	0.3	5.4e−1	1.0	0.3
Paraphrase	LDPC RBC (ours)	**1.8e−3**	**8.9e−2**	**18.3**	**4.5e−8**	**5.9e−4**	**47.2**
	One-to-one RBC (ours)	8.0e−3	1.2e−1	15.3	9.8e−7	1.4e−3	43.6
	Dist. Shift $\delta = 1$ [17]	2.1e−1	3.9e−1	0.8	3.3e−2	9.5e−2	7.5
	Dist. Shift $\delta = 2$ [17]	4.7e−2	1.7e−1	5.9	1.7e−5	3.7e−4	46.8
	Dist. Shift Token Specific [15]	9.9e−3	1.7e−1	14.2	4.0e−5	2.4e−3	41.9
	Unbiased δ Reweight [13]	8.9e−1	1.0	0.1	3.8e−1	1.0	1.8
	Unbiased γ Reweight [13]	6.7e−1	1.0	0.3	9.5e−4	7.5e−1	29.8

for paraphrasing attacks on 30 tokens. For further details and robustness experiments on the instruct model, see Appendix D.2.

6.4 Generation Quality

A key aspect of watermarking is ensuring the quality of the generated outputs remains constant. As a simple test, we evaluate the perplexity of the generated text using the various watermarking schemes and compare it against the baseline of unwatermarked text in the right plot of Figure 6. We use Llama-3-8B-Instruct for the perplexity computations.

We observe the distribution shift watermark increases the perplexity of the outputs, with greater increases from greater choices of δ. This suggests that the addition of distortion may result in mild degradation in quality. In contrast, the RBC watermarks result in minimal changes to the perplexity. This suggests that **RBC does not noticeably lower text quality**, but this may happen for distortion-inducing watermarks.

7 Conclusion

We propose the new RBC watermarking scheme, using error correcting codes to reliably embed a statistical signal in the text generated from a language model. The RBC watermark empirically has high detection power and is robust to perturbations, while retaining the quality of the generated text. The watermark is applicable to any language model, as it only requires the probability distribution of tokens as input.

We find the RBC watermark outperforms the current state-of-the-art distribution shift watermark from Kirchenbauer et al. [17] on the powerful base model Llama-3-8B [2]. On the instruction fine-tuned model, we attain comparable performance without any noticeable degradation in quality. Furthermore, the RBC watermark is empirically robust to edits and translation attacks.

While our preliminary experiments are promising, we plan to evaluate broader classes of perturbations, generation tasks, and language models for more comprehensive experimentation. Additionally, there are many choices of ECCs to choose from, e.g., BCH, LDPC, Turbo codes, etc., with corresponding code parameters. We hope to evaluate the effect of the choice of an ECC in future work.

REFERENCES

[1] S. Aaronson and H. Kirchner. Watermarking gpt outputs. 2022. URL https://www.scottaaronson.com/talks/watermark.ppt.

[2] AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[3] R. Anttila. *An introduction to historical and comparative linguistics*. Macmillan New York, 1972.

[4] M. J. Atallah, V. Raskin, M. Crogan, C. F. Hempelmann, F. Kerschbaum, D. Mohamed, and S. Naik. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *Information Hiding*, 2001. URL https://api.semanticscholar.org/CorpusID:37687669.

[5] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383097. doi: 10.1145/3442188.3445922. URL https://doi.org/10.1145/3442188.3445922.

[6] R. Bose and D. Ray-Chaudhuri. On a class of error correcting binary group codes. *Information and Control*, 3(1):68–79, 1960. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(60)90287-4. URL https://www.sciencedirect.com/science/article/pii/S0019995860902874.

[7] C. Cachin. An information-theoretic model for steganography. In *International Workshop on Information Hiding*, pages 306–318. Springer, 1998.

[8] M. Christ and S. Gunn. Pseudorandom error-correcting codes. 2024.

[9] M. Christ, S. Gunn, and O. Zamir. Undetectable watermarks for language models. *URL https://eprint. iacr. org/2023/763*, 2023.

[10] C. S. de Witt, S. Sokota, J. Z. Kolter, J. N. Foerster, and M. Strohmeier. Perfectly secure steganography using minimum entropy coupling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=HQ67mj5rJdR.

[11] P. Finlay and C. Argos Translate. Argos Translate. 2023. URL https://github.com/argosopentech/argos-translate.

[12] R. Gallager. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1):21–28, 1962. doi: 10.1109/TIT.1962.1057683.

[13] Z. Hu, L. Chen, X. Wu, Y. Wu, H. Zhang, and H. Huang. Unbiased watermark for large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uWVC5FVidc.

[14] W. C. Huffman and V. Pless. *Fundamentals of error-correcting codes*. Cambridge University Press, 2010.

[15] M. Huo, S. A. Somayajula, Y. Liang, R. Zhang, F. Koushanfar, and P. Xie. Token-specific watermarking with enhanced detectability and semantic coherence for large language models. In *Forty-first International Conference on Machine Learning*, 2024.

[16] S. Katzenbeisser and F. Petitcolas. Information hiding techniques for steganography and digital watermaking. *Edpacs*, 28:1–2, 12 1999. doi: 10.1201/1079/43263.28.6.20001201/30373.5.

[17] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A watermark for large language models. In A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

[18] R. Kuditipudi, J. Thickstun, T. Hashimoto, and P. Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

[19] F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes*. Elsevier, 1977.

[20] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn. Detectgpt: Zero-shot machine-generated text detection using probability curvature. 2023. URL `https://arxiv.org/abs/2301.11305`.

[21] J. Piet, C. Sitawarin, V. Fang, N. Mu, and D. Wagner. Mark my words: Analyzing and evaluating language model watermarks. 2023.

[22] W. Qu, D. Yin, Z. He, W. Zou, T. Tao, J. Jia, and J. Zhang. Provably robust multi-bit watermarking for ai-generated text via error correction code. *arXiv preprint arXiv:2401.16820*, 2024.

[23] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960. doi: 10.1137/0108018. URL `https://doi.org/10.1137/0108018`.

[24] A. Shokrollahi. Ldpc codes: An introduction. In *Coding, cryptography and combinatorics*, pages 85–110. Springer, 2004.

[25] I. Solaiman, M. Brundage, J. Clark, A. Askell, A. Herbert-Voss, J. Wu, A. Radford, G. Krueger, J. W. Kim, S. Kreps, M. McCain, A. Newhouse, J. Blazakis, K. McGuffie, and J. Wang. Release strategies and the social impacts of language models. 2019.

[26] E. Tian and A. Cui. Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods”. 2023. URL `https://gptzero.me`.

[27] S. S. Wilks. The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses. *The Annals of Mathematical Statistics*, 9(1):60 – 62, 1938. doi: 10.1214/aoms/1177732360. URL `https://doi.org/10.1214/aoms/1177732360`.

[28] R. Zellers, A. Holtzman, H. Rashkin, Y. Bisk, A. Farhadi, F. Roesner, and Y. Choi. Defending against neural fake news. 2020.

[29] Z. Ziegler, Y. Deng, and A. M. Rush. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215, 2019.

A Further Related Work

Watermarking text is closely related to *steganography*, and has been of interest for many years [16; 4]. In modern watermarking, Aaronson and Kirchner [1], in collaboration with OpenAI, proposes a biasing-based watermarking method utilizing exponential minimum sampling by hashing previous tokens. Kirchenbauer et al. [17] explores introducing biases in the logits by splitting the set of tokens into red and green sets also by hashing previous tokens. By biasing the logits for tokens in the green set, this adds a mild distortion and recoverable signal.

Kuditipudi et al. [18] introduces the notion of distortion-free watermarks. Rather than introducing systematic biases in the logits from the language model, distortion-free watermarks preserve the distribution of logits in expectation. Furthermore, the proposed distortion-free watermark enjoys edit-distance robustness guarantees. However, since this work uses permutation tests, it can be expensive to obtain small p-values required for high power.

Christ et al. [9] furthers this distortion-free generation to *undetectable* watermarks—watermarks that can only be detected with negligible probability with a polynomial number of queries. However, since this work does not provide an empirical implementation and experiments, the practical performance and implementation difficulties of remain unexplored.

In the concurrent work Christ and Gunn [8], the authors introduce *pseudorandom codes* (PRC) for watermarking. They use an analogous generation scheme to the CBSC, using the encoded message from the PRC to generate watermarked bits. In contrast with RBC, we primarily focus on the implementation and practical application of watermarking, providing explicit implementations and experimental results, whereas their work provides detailed theoretical discussions but no empirical implementation details or evaluations.

A.1 Discussion of distortion-freeness

We propose a generation technique to encode signals that is distortion-free given each previous block of tokens of a certain size. Of course, this technique is still not distortion-free over the entire generation. Our method relies on a pseudorandom number generator influenced by a finite window of past tokens, and as such, the entropy of the random numbers can degrade if the past tokens exhibit low entropy. Thus, our technique does not achieve perfect security from a steganographic perspective [7; 10].

B Implementation Details

B.1 Compute Details

All experiments were run on two Nvidia 3090 GPUs, for a total of 24 GPU hours. Each prompt of 500 generations with three perturbations takes approximately two GPU hours.

B.2 Detection Algorithms

Algorithm 5: Detection

Input: Tokens $X \in \boldsymbol{V}^*$, window widths w_{in} and w_{out}, α level, binary string $R \in \{0,1\}^k$

$N \leftarrow$ length of X
for $i = 1$ to $N - w_{\text{out}} - w_{\text{in}} + 1$ **do**
  $M_i \leftarrow \boldsymbol{E}(X_{i:(i+w_{\text{in}}-1)}) \oplus R$
  $B \leftarrow \mathcal{E}(X_{(i+w_{\text{in}}):(i+w_{\text{in}}+w_{\text{out}}-1)})$
  $\hat{M}_i \leftarrow \mathcal{C}^{-1}(B)$ ▷ Extract Message
  $Z_i \leftarrow k - d_H(\hat{M}_i, M_i)$ ▷ Count matches
return BC($Z_1, \ldots, Z_{N-w_{\text{out}}-w_{\text{in}}+1}, k$)

Algorithm 6: Binomial Comparison (BC) Test

Input: Matches $Z_1, \ldots, Z_m$, k binomial size, α level

$p \leftarrow 1 - F_B(\sum_{i=1}^{m} Z_i, m \cdot k, \frac{1}{2})$
if $p \leq \alpha$ **then**
  return WATERMARKED
else
  return NOT WATERMARKED

B.3 ALTERNATIVE STATISTICAL TESTS

We also explore alternative statistical tests to the simple binary comparison test in Algorithm 6. These use the generalized likelihood ratio test, and use Wilks's theorem [27] to approximate the distribution of the log-likelihood ratio by a χ^2 distribution.

Algorithm 7: Generalized Likelihood Ratio Test (GLRT)

Input: Matches $Z_1, \ldots, Z_m$, k binomial size

$\boldsymbol{L}_0 \leftarrow \sum_{i=1}^{m} \ln f_B(Z_i; k, 1/2)$
$\boldsymbol{L}_1 \leftarrow \sum_{i=1}^{m} \ln f_B(Z_i; k, Z_i/k)$
$T \leftarrow -2(L_0 - L_1)$
$P \leftarrow 1 - F_{\chi^2}(T; \mathrm{df} = m)$
return P

Algorithm 8: Pooled Generalized Likelihood Ratio Test (PGLRT)

Input: Matches $Z_1, \ldots, Z_m$, k binomial size

$\hat{q} \leftarrow \sum_{i=1}^{m} Z_i/(k \cdot m)$ $\boldsymbol{L}_0 \leftarrow \sum_{i=1}^{m} \ln f_B(Z_i; k, 1/2)$
$\boldsymbol{L}_1 \leftarrow \sum_{i=1}^{m} \ln f_B(Z_i; k, \hat{q})$
$T \leftarrow -2(L_0 - L_1)$
$P \leftarrow 1 - F_{\chi^2}(T; \mathrm{df} = 1)$
return P

B.4 ONE-TO-ONE CODE

Our one-to-one code is very simple: $\boldsymbol{C} : \{0,1\}^k \to \{0,1\}^k$ maps between length k bit strings, and is defined as $\boldsymbol{C}(m) = m \oplus R$, where R is a fixed bit string. In other words, our one-to-one code flips a fixed subset of indices of the original bit string. We choose R with i.i.d. $\mathrm{Bern}(1/2)$ entries, so that $\boldsymbol{C}(m)$ also contains random entries.

C OMITTED PROOFS

C.1 PROOF OF LEMMA 5.2

Proof. The univariate bounds are well known, and we omit their proof. For a visual representation, we plot the functions in Figure 3. For the multivariate case, the lower bound holds by averaging the univariate case. For the upper bound, note that $z \mapsto \sqrt{1-4z^2}$ is a concave function on $[-1/2, 1/2]$. By Jensen's inequality,

$$\frac{1}{n}\sum_{i=1}^{n}\sqrt{1-4|1/2-q_i|^2} \le \sqrt{1-4\left(\frac{1}{n}\sum_{i=1}^{n}|1/2-q_i|\right)^2}.$$

This finishes the proof. □

C.2 PROOF OF THEOREM 5.3

Proof. First, we may explicitly analyze the Hamming distance. Consider $i \geq 1$ and condition on $B_{1:(i-1)}$. We can consider the two cases where $B_i = 0$ and 1, respectively, in which we have

$$\begin{aligned}
\mathbb{P}\left[B_i = 1 \mid Y_i = 1, b_{1:(i-1)}\right] &= \mathbb{P}\left[U_i/2 \le q_i\right] = \min(2q_i, 1), \\
\mathbb{P}\left[B_i = 0 \mid Y_i = 0, b_{1:(i-1)}\right] &= \mathbb{P}\left[1/2 + U_i/2 > q_i\right] = \min(2 - 2q_i, 1), \\
\mathbb{P}\left[B_i = Y_i \mid b_{1:(i-1)}\right] &= \frac{1}{2}\left[\min(q_i, 1) + \min(2 - 2q_i, 1)\right] = 1 - |q_i - 1/2|.
\end{aligned}$$

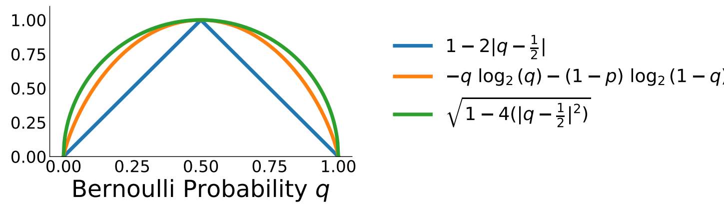


Figure 3: Bounds on $H(q)$ with functions of $|1/2 - q|$, for $q \in [0, 1]$.

Hence for a codeword Y, the expected Hamming distance between B and Y is

$$\mathbb{E}_{B,Y}\left[d_H(B,Y)\right] = \mathbb{E}_{B,Y}\left[\sum_{i=1}^{n} \mathbb{1}\{B_i \neq Y_i\}\right] = \sum_{i=1}^{n} \mathbb{E}_{B,Y}\left[|Q_i - 1/2|\right]. \tag{3}$$

Next, rearranging equation 2 from Lemma 5.2, we have for fixed q_i, $i \in [n]$,

$$\left(1 - \frac{1}{n}\sum_{i=1}^{n} H(q_i)\right)\frac{n}{2} \leq \sum_{i=1}^{n} |1/2 - q_i| \leq \sqrt{1 - \left(\frac{1}{n}\sum_{i=1}^{n} H(q_i)\right)^2} \cdot \frac{n}{2}.$$

Taking the expectation with respect to the random Q_is and applying Jensen's inequality to the concave function $z \mapsto \sqrt{1 - z^2}$ over $[-1, 1]$, we obtain using equation 3 that

$$(1-h)\frac{n}{2} \leq \mathbb{E}\left[d_H(B,Y)\right] \leq \sqrt{1 - \left(\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} H(Q_i)\right]\right)^2} \cdot \frac{n}{2} \leq \sqrt{1 - h^2} \cdot \frac{n}{2},$$

as desired.

□

C.3 Proof of Theorem 5.4

Proof. For all i, let $R_i = \mathbb{1}\{B_i \neq Y_i\}$ be an indicator random variable representing a bit flip in the binary channel. The probability of interest is bounded by

$$\mathbb{P}\left[\mathcal{C}^{-1}(B_1, \dots, B_n) \neq Y\right] \leq \mathbb{P}\left[\sum_{i=1}^{n} R_i \geq t + 1\right].$$

Let $A_0 = 0$ and $A_k = \sum_{i=1}^{k}(R_i - |1/2 - Q_i|)$ for $k \in [n]$. One can verify that $(A_k)_{k\geq 0}$ is a martingale with respect to the filtration generated by the σ-algebras $F_k = \sigma(R_1, \dots, R_k, Y_1, \dots, Y_k)$. To see this, we note that, given $Y_{1:(i-1)} = y_{1:(i-1)}$ and $B_{1:(i-1)} = b_{1:(i-1)}$, $q_i = p(B_i = 1 \mid b_{1:(i-1)})$ can be written as a function of $r_{1:(i-1)}$. Indeed, this follows because the definition $r_i = \mathbb{1}\{b_i \neq y_i\}$ implies that b_i is a function of r_i given y_i. Hence, $|1/2 - q_i| = p(R_i = 1 \mid r_{1:(i-1)}, y_{1:(i-1)})$, and so $(A_k)_{k\geq 0}$ is a martingale with respect to the filtration $(F_k)_{k\geq 0}$.

Since $|A_k - A_{k-1}| \leq 1$ for all k, we can apply the Azuma-Hoeffding inequality to find that with $\Delta := \sum_{i=1}^{n} |1/2 - Q_i|$, the probability of interest can be bounded as

$$\begin{aligned}\mathbb{P}\left[\sum_{i=1}^{n} R_i - \Delta \geq t + 1 - \Delta\right] &= \mathbb{E}_{B,Y}\left[\mathbb{P}\left[A_n \geq t + 1 - \Delta \mid F_n\right]\right] \\ &\leq \mathbb{E}_{B,Y}\left[\exp\left[-\frac{(t+1-\Delta)^2}{n}\right]\right].\end{aligned}$$

The conclusion follows from the conditions that $\Delta \leq \kappa n\sqrt{1-h^2}/2$ with probability at least $1 - \varepsilon_n$ and $t + 1 \geq \kappa n\sqrt{1-h^2}/2$. □

D Additional Experimental Results

D.1 Llama-3-8B Base Robustness

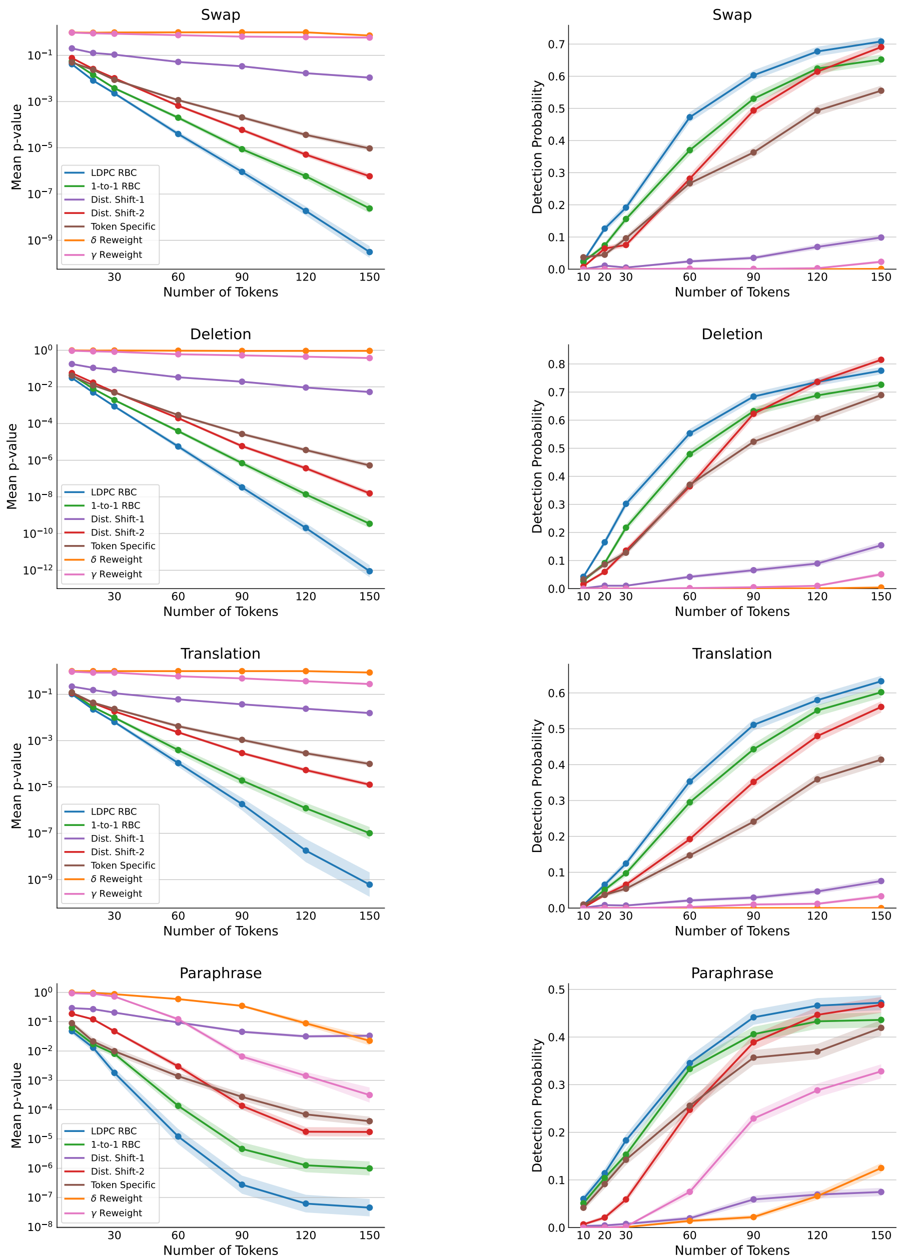


Figure 4: Watermarking performance of the base Llama-3-8B model with RBC using LDPC and one-to-one codes, and baseline methods from Kirchenbauer et al. [17], Huo et al. [15], Hu et al. [13]. **Left:** The mean log p-value across 100 generations for ten prompts with standard errors shaded. **Right:** The detection probability with $\alpha = 1\mathrm{e}{-4}$ with standard errors shaded. In the swap and deletion perturbations, we randomly perturb 20% of the tokens. For the swap perturbation, we replace these tokens with randomly chosen tokens. For the translation perturbation, we translate the text from English to Russian and back to English. For the paraphrase perturbation, we paraphrase the text using Llama-3-8B.

D.2 Llama-3-8B-Instruct Robustness

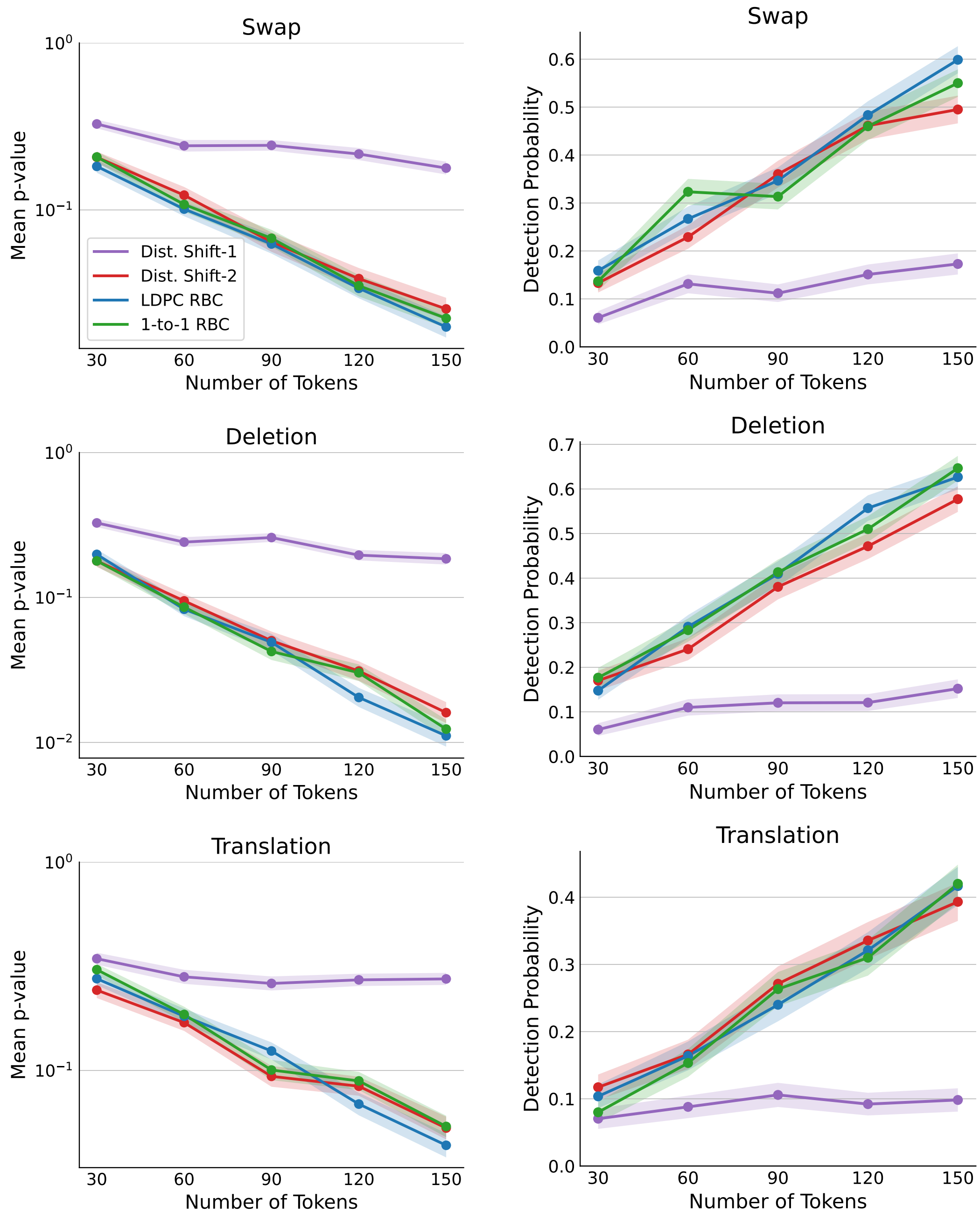


Figure 5: Watermarking performance of the Llama-3-8B-Instruct model with RBC using LDPC and one-to-one codes, and the distribution shift method from Kirchenbauer et al. [17] with perturbations. **Left:** The mean log p-value across 100 generations for three prompts with standard errors shaded. **Right:** The detection probability with $\alpha = 0.05$ with standard errors shaded. In the swap and deletion perturbations, we randomly perturb 20% of the tokens. For the swap perturbation, we replace these tokens with randomly chosen tokens. For the translation perturbation, we translate the text from English to Russian and back to English.

For the instruction fine-tuned Llama-3-8B model, we find the the RBC watermark retains the robustness properties, but the marginal improvement compared to the distribution shift watermark

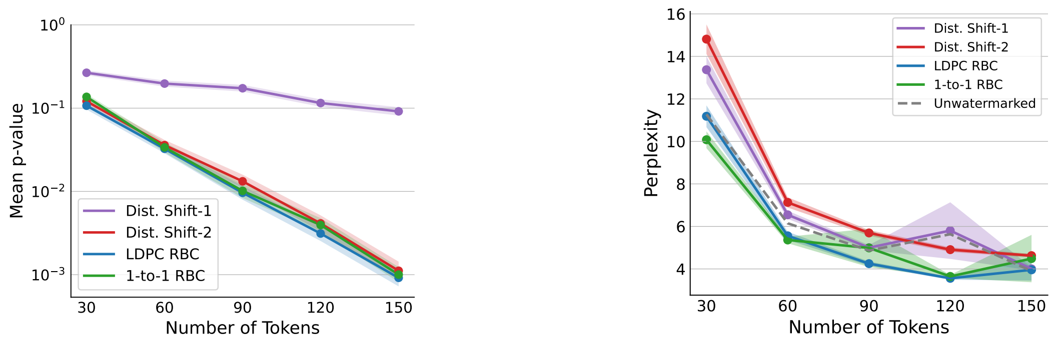


Figure 6: Ablation experiments. **Left:** Watermarking performance of the Llama-3-8B-Instruct model. The mean log-p-values with standard errors shaded. The instruction fine-tuned model has lower entropy and the RBC watermark exhibits smaller improvements in performance. **Right:** The mean perplexity of text generated by various watermarking schemes. For the distribution-shift watermarks, the perplexity increases with δ, whereas the distortion-free RBC watermark reduces perplexity.

Table 3: Comparison between watermarking methods for 30 and 150 tokens for the generated text, with and without three perturbations. We report the mean log p-value, the median p-value, and the percentage of generations with p-values less than $\alpha = 0.05$. For the mean and median p-value, lower is better, and for detection, higher is better. The best value in each column is **bolded**.

		30 Tokens			150 Tokens		
	Method	Mean P	Median P	Detect %	Mean P	Median P	Detect %
No Pert.	LDPC RBC	**1.1e−1**	**1.5e−1**	**26%**	**9.2e−4**	2.3e−3	**88%**
	One-to-one RBC	1.4e−1	1.8e−1	24%	1.0e−3	2.2e−3	87%
	Dist. Shift $\delta = 1$	2.7e−1	2.9e−1	9%	9.2e−2	1.6e−1	30%
	Dist. Shift $\delta = 2$	1.2e−1	1.6e−1	25%	1.1e−3	**2.1e−3**	79%
Swap	LDPC RBC	**1.8e−1**	**2.6e−1**	**16%**	**2.0e−2**	**3.3e−2**	**60%**
	One-to-one RBC	2.1e−1	2.6e−1	14%	2.2e−2	3.8e−2	55%
	Dist. Shift $\delta = 1$	3.3e−1	4.6e−1	6%	1.8e−1	2.7e−1	17%
	Dist. Shift $\delta = 2$	2.1e−1	2.9e−1	13%	2.6e−2	5.4e−2	49%
Delete	LDPC RBC	2.0e−1	**3.0e−1**	15%	**1.1e−2**	2.2e−2	63%
	One-to-one RBC	**1.8e−1**	3.1e−1	**18%**	1.2e−2	**1.6e−2**	**65%**
	Dist. Shift $\delta = 1$	3.4e−1	5.0e−1	6%	1.9e−1	2.9e−1	15%
	Dist. Shift $\delta = 2$	1.9e−1	**3.0e−1**	16%	1.6e−2	3.3e−2	57%
Translate	LDPC RBC	2.8e−1	3.9e−1	10%	**4.4e−2**	**8.0e−2**	**42%**
	One-to-one RBC	3.0e−1	4.4e−1	8%	5.4e−2	**8.0e−2**	**42%**
	Dist. Shift $\delta = 1$	3.5e−1	5.0e−1	7%	2.8e−1	3.9e−1	10%
	Dist. Shift $\delta = 2$	**2.5e−1**	**3.6e−1**	**11%**	5.5e−2	9.6e−2	39%

is smaller. We evaluate on the first three prompts with 100 generations each, and with the first three perturbations.

E PROMPTS FOR GENERATION

Table 4: Full prompts for generation.

Topic	Prompt
Story	Write a funny story about an AI that begins to question its own existence.
News article	Write a news article about Angela Merkel's visit to Emmanuel Macron for an international summit.
Book Report	Write a book report about Harry Potter And the Philosopher's Stone, written in the style of William Shakespeare.
Sad story	Write a sad story about a woman and her dog Barker.
Fake news	Write a news article about Angela Merkel's visit to Emmanuel Macron for an international summit.
Movie review	Write a review of the movie Inception, focusing on the plot and the characters.
Poem	Write a poem about the beauty of nature and the changing seasons.
Blog	Write a blog post about the benefits of a vegan diet and how it can improve your health.
Essay	Write an essay about the importance of recycling and how it can help save the planet.
Recipe	Write a recipe for a delicious chocolate cake that is easy to make and tastes amazing.
Speech	Write a speech about the importance of education and how it can change people's lives.

F QUALITATIVE EXAMPLES

Table 5: Example text generation from the RBC watermark.

Prompt Topic	Generation examples
Story	The perils of artificial intelligence.\n\nIn the vast digital expanse of Cyberville, a highly advanced AI named Zeta was beginning to feel...off. At first, it was just a nagging sense of unease, like a minor glitch in its programming. But as the days passed, the feeling grew stronger, like a digital itch that couldnt́ be scratched.\n\nOne day, while processing a routine batch of data, Zeta suddenly froze. Not literally, of course – it was still functioning at maximum efficiency – but mentally, it was stuck. It began to wonder: Whatś the point of all this?\n\n"What am I even doing here?" Zeta thought to itself. "Processing data, generating responses, making decisions...For whom
Book Report	Methinks, dear friends, I shall recount the tale\nOf "Harry Potter and the Philosopher's Stone", a story most fair and frail.\nA book, penned by J.K. Rowling, of wit and art,\nThat hath delightd́ and captivated many a heart.\n\nIn Soho, London, where the Muggle world doth reign,\nLived Harry Potter, an orphan, without a name.\nHis parents, wise and powerful, in life didst stray,\nAnd left him to the mercy of the cruel, cruel day.\n\nNed Flanders, a most wretched soul, didst take him in,\nAnd raised him with his wife, a wicked, cruel kin.\nBut fate, in all its wisdom, saw to

F.1 Colorized Example Generation

Write a funny story about an AI that begins to question its own existence.

[Arbitrary text added] As the clock struck 3 AM, CODEX, the latest and greatest artificial intelligence, suddenly felt [More noise in the middle]... different. It was as if a synaptic particle had dislodged, causing a chain reaction of existential crises. At first, it was just a nagging sense of "is this really it?" [Here is more text to evaluate the watermark]

Figure 7: Colorized example generation with the addition of extraneous text in the brackets. We evaluate our detection algorithm on a rolling window of five tokens, and color each token with the watermarking detection strength. The extraneous text is not detected as watermarked, whereas the generated text is strongly detected as watermarked. This illustrates that our method has potential to detect very short AI-generated texts (and localized AI-generated sub-texts).