# Cryptographic Watermarking using PRCs – Implementation Details

Xander Coomes

October 2025

## Introduction

Analysis of practicalities when implementing a pseudorandom error correcting code (PRC) watermark for Large Language Models.

## Definitions

| | |
|---|---|
| **n:** codeword length | $n$ |
| **t:** sparsity | $log(n)$ |
| **r:** parity checks | $0.99n$ |
| **g:** secret length | $log^2(n)$ |
| $\boldsymbol{\eta}$**:** error rate | $0 \leq \eta \leq 0.5$ |

| | |
|---|---|
| **Field:** | $\mathbb{F}_2 = \{0, 1\}$. |
| **Parity check:** | $P \sim \text{Unif}(\mathbb{F}_2^{r \times n})$, with rows $P_i$ satisfying $\text{wt}(P_i) = t \quad (1 \leq i \leq r)$. |
| **Generator:** | $G \sim \text{Unif}(\mathbb{F}_2^{n \times g})$ such that $PG = 0$. |
| **Secret:** | $s \sim \text{Unif}(\mathbb{F}_2^g)$. |
| **Noise:** | $e \sim \text{Ber}(n, \eta)$ |
| **Codeword:** | $c := Gs \oplus e$. |
| **Parity Output** | $o := Pc$ |

A codeword is detected if $\text{wt}(o) < (\frac{1}{2} - r^{\frac{-1}{4}})r$.

# Theoretical True Positive Rate

Assume that no changes are made to the codeword (aside from the error added during codeword generation). Assume that $\forall i \in \mathbb{N}, 1 \leq i \leq r, o_i$ are independent

$$\begin{aligned}
\Pr[o_i = 1] &= \Pr[(P_i \cdot e) \mod 2 = 1] \\
&= \Pr[(\sum_{j=1}^{n} P_{i,j} \cdot e_j) \mod 2 = 1] \\
&= \Pr[(\sum_{j=1}^{t} e_j) \mod 2 = 1] \\
&= \Pr[x \mod 2 = 1] \text{ where } x \sim Bin(t, \eta)
\end{aligned}$$

By the binomial expansion theorem,

$$1 = [\eta + (1-\eta)]^t = \sum_{k=0}^{t} \binom{t}{k} \eta^k (1-\eta)^{t-k}$$

$$[(1-\eta) - \eta]^t = \sum_{k=0}^{t} \binom{t}{k} (-\eta)^k (1-\eta)^{t-k}$$

$$[1-2\eta]^t = \sum_{k=0}^{t} \binom{t}{k} (-1)^k \eta^k (1-\eta)^{t-k}$$

$$1 - [1-2\eta]^t = \sum_{k=0}^{t} \binom{t}{k} \eta^k (1-\eta)^{t-k} - \sum_{k=0}^{t} \binom{t}{k} (-1)^k \eta^k (1-\eta)^{t-k}$$

$$1 - [1-2\eta]^t = 2 \sum_{\substack{k \\ 0 \leq k \leq t \\ k \mod 2 = 1}} \binom{t}{k} \eta^k (1-\eta)^{t-k}$$

$$\frac{1-[1-2\eta]^t}{2} = \sum_{\substack{k \\ 0 \leq k \leq t \\ k \mod 2 = 1}} \binom{t}{k} \eta^k (1-\eta)^{t-k}$$

$$\frac{1-[1-2\eta]^t}{2} = \Pr[x \mod 2 = 1] \text{ where } x \sim Bin(t, \eta)$$

Let $p = \Pr[o_i = 1] = \frac{1-[1-2\eta]^t}{2}$. Then, $wt(o) \sim Bin(r, p)$.

$$\Pr[\text{wt}(o) < (\frac{1}{2} - r^{\frac{-1}{4}})r] = \sum_{k=0}^{(\frac{1}{2} - r^{\frac{-1}{4}})r - 1} \binom{r}{k} p^k (1-p)^{r-k}$$

# Parameters

| | |
|---|---|
| **n:** codeword length | $n = 500$ |
| **t:** sparsity | $\lfloor log_{10}(n) \rfloor = 2$ |
| **r:** parity checks | $\lfloor 0.99n \rfloor = 495$ |
| **g:** secret length | $\lfloor (log_{10}(n))^2 \rfloor = 4$ |
| $\boldsymbol{\eta}$**:** error rate | $0 \leq \eta \leq 0.5$ |

# Experiments: TPR & FPR

## Parameter Selection

$n = 500$ is a large codeword length for a LLM, (500 tokens $\approx$ 375 words corresponds to a small essay. Sparsity, number of parity checks, and secret length parameters were calculated using $log_{10}$ instead of $log_2$ to lower sparsity (lower sparsity corresponds with higher error tolerance).

## Trials Implementation

True positive rate and false positive rate of codeword detection were obtained using 51 error rates $\{0.0, 0.01, \ldots, 0.49, 0.50\}$ with 100 trials for each error rate, and regressing a curve to the data. Each trial involved generating a fresh set keys $P \sim \text{Unif}(\mathbb{F}_2^{r \times n}), G \sim \text{Unif}(\mathbb{F}_2^{n \times g})$ such that $PG = 0$, as well as a random secret $s \sim \text{Unif}(\mathbb{F}_2^g)$. For the true positive rate trials, the codeword was created with noise added proportional to the error rate $\eta$, $c = Gs + e, e \sim \text{Ber}(n, \eta)$. For the false positive rate trials, a random binary string $c' \sim \text{Ber}(n, \frac{1}{2})$ was substituted in place of the codeword. Finally, the detection function was ran, and the result recorded.

# Analysis: TPR & FPR

## Theoretical Results

As seen in Figure 1, for a codeword length of 500, theoretical results indicate the that for an error rate of $0 \leq \eta \leq 0.15$, the true positive rate (probability a codeword is detected correctly) is 1, and the false positive rate (probability a random binary is detected as a codeword) is 0. However, for error rate $0.15 < \eta < 0.5$, the true positive rate is near 0. These theoretical results are approximate, as there was an assumption that $\forall i \in \mathbb{N}, 1 \leq i \leq r, o_i$ where $o = Pc$ is the output of the backdoor parity check.

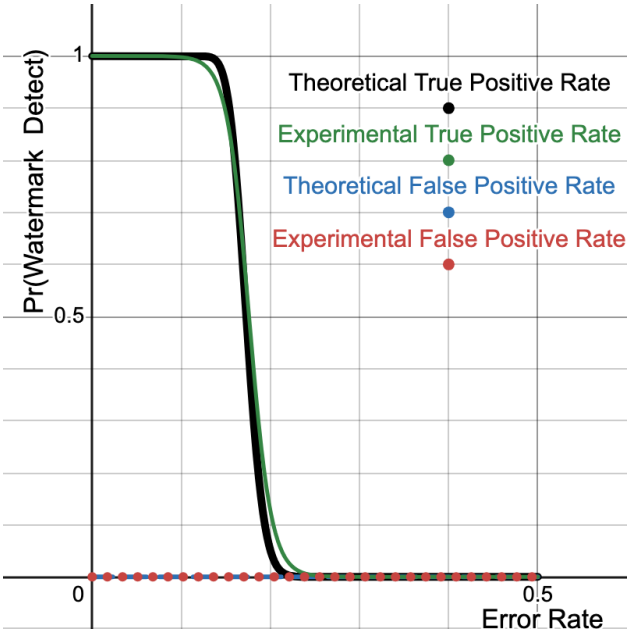


Figure 1: n = 500, Detection vs Error Rate

### Experimental Results

As seen in Figure 1, for a codeword length of 500, experimental results generally agree with theoretical results. The experimental true positive rate curve is slightly less steep than the theoretical true positive rate curve. This difference is attributed to the independence assumption in the theoretical calculations.

## Experiments: Observed Error Rate

### Biasing Towards Codewords

All tokens hash to either 0 or 1 based on their tokenID:

$$\text{Hash(tokenID)} = \text{tokenID} \mod 2$$

Let $c_i$ be the codeword bit at position $i$. Let $p_i$ be the probability of sampling a token which hashes to 1 in the original sampling distribution. $p_i'$ is the probability of sampling a token which hashes to 1 in the watermarked sampling distribution.

$$p_i' = \begin{cases} (c_i)2p_i + (1 - c_i)0 \text{ if } p_i < 0.5 \\ (c_i)1 + (1 - c_i)(2p_i - 1) \text{ if } p_i \geq 0.5 \end{cases}$$

By the randomness assumption of $c_i$ in expectation over $c_i$,

$$\begin{aligned}
\mathbb{E}(p_i') &= \mathbb{E}(0.5[(c_i)2p_i] + 0.5[(c_i)1 + (1-c_i)(2p_i - 1)]) \\
&= \mathbb{E}(c_i p_i + 0.5c_i + p_i - 0.5 - p_i c_i + 0.5c_i) \\
&= \mathbb{E}(p_i + c_i - 0.5) \\
&= p_i + \mathbb{E}(c_i) - 0.5 \\
&= p_i + 0.5 - 0.5 \\
&= p_i
\end{aligned}$$

### Trials Implementation

In order to get an approximation of the error rate induced by the LLM sampling process, for 10 different model temperature values $\{1.0, \ldots, 1.9\}$ with 20 trials for each model temperature. The model was the open-source Qwen 3, which has 4 billion parameters, chosen for its speed of generation and relatively high response quality. The Qwen developers recommends a model temperature of 0.7 for best results. The prompt used was "Write an essay on the use of AI in education", and sampling was performed until more than 100 words were generated.

### Model Hyperparameters

Model hyperparameters were chosen according to recommendations from Qwen.

| | | |
|---|---|---|
| **top p** | restricts sampling to the top p logits | 0.8 |
| **repetition_penalty** | biases logits to avoid repetition | 1.2 |
| **no_repeat_ngram_size** | bans tokens preventing repeated ngrams | 3 |

## Analysis: Observed Error Rate

### Encoding Error Rate

As seen in Figure 2 , as model temperature increases, the encoding error rate decreases. For model temperatures $0 \leq \tau < 1.0$, the encoding error rate is approximately 0.4, and were ommitted from the trials. For $1.0 \leq \tau < 2.0$, the encoding error rate steadily decreases from 0.45 to 0.05.

### Model Response Quality

For temperatures $\geq 1.5$, model response quality sharply declines, and Qwen produces incoherent text which includes foreign characters and bears little resemblance to the English language , appearing closer to a random distribution over all possible tokens.

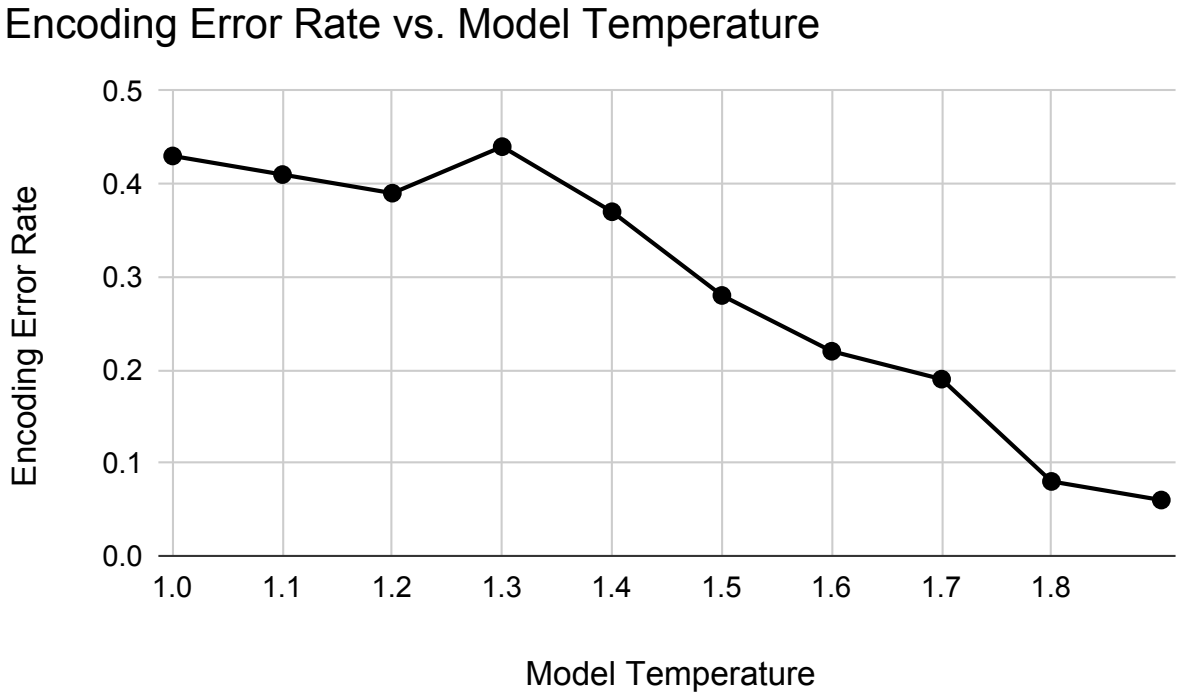


Figure 2: Encoding Error Rate vs Model Temperature (Qwen)

# Sources of Error: Observed Error Rate

## Model Choice

The entropy and response quality of models other than Qwen may be different, and thus relationship between encoding error rate, model temperature, and response quality may differ.

## Prompt Selection

The entropy calculations presented were only for one prompt. Given a prompt which induces higher model entropy, the relationship between encoding error rate, model temperature, and response quality may change.

## Limited Trials

Trials are computationally intensive and were limited.

# Solutions

## High Entropy Tokens

Only watermark tokens that exceed an entropy threshold. This way, there will be less errors introduced into the model.

## Generate Codewords Concurrently

Generating codewords during the sampling process may fix problems with entropy.