Alon Rosen

Watermarking

However we happen to know he's in the water barrel.[*]

A *watermark* leaves hidden evidence of origin in content produced by a generative model. Someone holding the appropriate key is able to detect this evidence, while the output should remain useful and natural to everyone else.

We first study *autoregressive text generation*, where many small randomized choices accumulate into a response. We then return to *diffusion*, where the watermark can be placed directly in the Gaussian randomness from which generation begins.

In both settings, the basic resource is the model's freedom among plausible outputs. This leaves room for a hidden signal that a secret-key holder can detect.

Reliable detection alone is not enough, and other properties may be desirable. *Undetectability* asks that, without the key, the watermarked generator be indistinguishable from the original one. This both avoids observable changes to the output distribution and hides the use of the watermark from outsiders. *Robustness* asks that the hidden signal survive certain modifications.

The main points of this chapter are:

- In autoregressive models, small biases in token sampling can be reliably detected.
- Keyed sampling can hide the watermark from anyone without the secret key.
- Pseudorandom codes can make the watermark survive certain modifications.
- Even guarantees against limited yet well-specified adversaries can be useful.

[*]Monty Python, *Monty Python's Flying Circus*.

1 Watermarking

Mr. Ken Andrews, of Leighton Road, Slough, has concealed himself extremely well. He could be almost anywhere. To an observer who does not know which option was chosen, finding him may be difficult. But we have the advantage of knowing that he's in the water barrel.

A generative model has a slew of options at its disposal: the same input can yield many plausible outputs. A *watermark* uses this freedom to leave a hidden signal, detectable with a secret key, while the output still looks ordinary.

For text, this can help flag AI-generated homework, establish provenance, or identify synthetic text circulating online. The same idea applies to images, audio, video, and other generated content.

Let X and Y be the input and output spaces, and let G be a randomized generative model. On $x \in X$, it outputs $y \leftarrow G(x)$ from some distribution over Y.

Definition 1 (Watermarking scheme)**.** *A watermarking scheme for G consists of polynomial-time algorithms* $(\mathrm{KeyGen}, \mathrm{Wat}, \mathrm{Detect})$*: $k \leftarrow \mathrm{KeyGen}(1^\lambda)$ generates a secret key, $y \leftarrow \mathrm{Wat}_k(x)$ generates a watermarked output, and $\mathrm{Detect}_k(y) \in \{0,1\}$ tests whether y is watermarked.*

The syntax alone says nothing about whether the scheme works. The most basic requirement from a watermarking scheme is *reliable detection*.

Definition 2 (Reliable detection)**.** *Fix $\varepsilon > 0$. A reliable watermarking scheme satisfies:*

- ***Completeness:*** *for every $x \in X$, $\Pr_{k,y}[\mathrm{Detect}_k(y) = 1] \geq 1 - \varepsilon$.*
- ***Soundness:*** *for every fixed $y_0 \in Y$, $\Pr_k[\mathrm{Detect}_k(y_0) = 1] \leq \varepsilon$.*

where $k \leftarrow \mathrm{KeyGen}(1^\lambda)$ and, in completeness, also $y \leftarrow \mathrm{Wat}_k(x)$.

Soundness rules out false positives. Completeness requires that watermarked outputs be detected whenever the generator has enough freedom to leave a signal.

We will later see how to guarantee *undetectability* and *robustness*. The former asks that, without the key, the watermarked generator look like the original; the latter asks that the watermark survive certain modifications. Their absence does not make the basic notion useless. Even if removal requires only modest effort, most users will not bother, so the output remains detectable.

Our detector uses a secret key; public detection is a separate question.

1.1 Autoregressive models

We first study language models. For watermarking, we need very little of their internal architecture: only the way they generate text one token at a time.

Definition 3 (Autoregressive model)**.** *A language model over a finite vocabulary V is a deterministic algorithm* Model *that maps $(x, t_{<i})$ to a distribution p_i over V, where*

$$p_i = \mathrm{Model}(x, t_{<i}).$$

A response is generated by repeatedly sampling $t_i \sim p_i$ until a distinguished end token is produced.

Given an input x and a prefix $t_{<i} = (t_1, \ldots, t_{i-1})$, the model assigns each token $v \in V$ a score $\ell_i(v)$, called its *logit*. The next-token probabilities are obtained by exponentiating and normalizing, $p_i(v) = e^{\ell_i(v)} / \sum_{w \in V} e^{\ell_i(w)}$. This map is known as the *softmax* function.

For example, a tokenizer might represent the sentence *Nobody likes a clever dick.* as

$$[\text{Nobody}][\text{ likes}][\text{ a}][\text{ clever}][\text{ dick}][.].$$

After the prefix [Nobody][likes], the model might, for illustration, return

$$p_i(\text{a}) = 0.32, \qquad p_i(\text{him}) = 0.18, \qquad p_i(\text{this}) = 0.09, \qquad \dots$$

The model samples one token, appends it to the prefix, and repeats. The exact tokens and probabilities depend on the tokenizer, model, and input.

1.2 The red–green trick

The *red–green trick* plants a small statistical bias in the token choices. At position i, the key and prefix $t_{<i}$ pseudorandomly mark a γ fraction of the vocabulary green and the rest red.

The watermark samples green tokens slightly more often than the original model. Ordinary text therefore hits the secret green sets at about rate γ, while watermarked text does so more often. Knowing the key, the detector reconstructs the green sets and counts the hits.

Theorem 1 (Red–green detection)**.** *If watermarking gives green tokens a positive probability advantage, then sufficiently long outputs can be detected with arbitrarily small error.*

Proof. The scheme can be implemented by using a PRF to rank the vocabulary at each position: token $v \in V$ receives the value $F_k(t_{<i}, v)$, and the first $\lfloor \gamma |V| \rfloor$ tokens in this ranking are declared green. The logits of the green tokens are then increased by $\delta > 0$, giving

$$\widehat{p}_i(v) = \frac{p_i(v) e^{\delta \mathbf{1}[v \text{ is green}]}}{\sum_{w \in V} p_i(w) e^{\delta \mathbf{1}[w \text{ is green}]}}.$$

If the green set has probability q under p_i, the bias changes this to $\widehat{q}(\delta) = e^\delta q/(1 - q + e^\delta q) > q$.

Suppose this increase is at least $\rho > 0$ at each step. Ordinary generation then has green rate about γ, while watermarked generation has rate at least $\gamma + \rho$. The detector can accept whenever the observed green fraction exceeds $\gamma + \rho/2$. Standard concentration bounds make either error probability $\exp(-\Omega(\rho^2 L))$, so $L = O(\rho^{-2} \log(1/\varepsilon))$ makes the error at most ε. □

2 Undetectable watermarking

Since $\widehat{p}_i \neq p_i$, the red–green trick changes the token distribution. Can the signal be hidden without changing that distribution?

We want two things from the sampling rule: the chosen token should still have distribution p_i, while the hidden random value attached to it should be unusually large. The first preserves the output distribution; the second gives the detector a signal. This is achieved by *Gumbel sampling*. For a distribution p over V and $U_v \leftarrow \text{Unif}(0, 1)$, output

$$X = \arg\max_{v \in V} U_v^{1/p_v}.$$

This has the property that $X \sim p$ and, moreover, $\Pr[U_v \leq u \mid X = v] = u^{1/p_v}$.

A key holder generates the sampling values pseudorandomly. For an observed token t_i, let $u_i = F_k(t_{<i}, t_i) \in (0, 1)$ be its keyed value. The second property says that, for watermarked text, these winning values are biased toward 1. The detector therefore accumulates

$$S = -\sum_{i=1}^{L} \log(1 - u_i).$$

For unwatermarked text, $\mathbb{E}[S] = L$ with standard deviation $\sqrt{L}$. Watermarking raises $\mathbb{E}[S]$; with enough randomness, the increase exceeds the $\sqrt{L}$ noise level and is detectable given k.

We also want a watermarked generator to look unwatermarked to anyone without the key.

Definition 4 (Undetectability)**.** *A watermark is* undetectable *if no efficient algorithm can distinguish access to G from access to* Wat_k*, even after polynomially many adaptively chosen queries.*

Undetectability serves two purposes. It gives a model-independent *quality* guarantee: any efficiently observable change caused by watermarking would distinguish Wat_k from G. It also *hides* whether watermarking is being used from anyone without the key.

Because keyed values are fixed by the prefix, repeated prefixes reuse randomness. The relevant quantity is the *empirical entropy* of the realized text,

$$H_e(t) = -\log \Pr[\mathrm{Model}(x) = t] = \sum_i -\log p_i(t_i).$$

It can be tracked during generation from the probabilities assigned to sampled tokens. Once a prefix has accumulated enough entropy, use it to seed the pseudorandom values; such a prefix is unlikely to recur, keeping the values fresh across queries. If enough entropy never accumulates, generation stays unwatermarked: completeness may fail, but undetectability is preserved.

3 Robust watermarking

A watermark is *robust* to a class E of efficient transformations if it remains detectable after any of them is applied to the watermarked text. Users may substitute words, delete passages, or keep only a substring, so the signal needs error correction. Ordinary error-correcting codes are not enough: their codewords can have visible structure. We want codewords that also look random.

Definition 5 (PRC)**.** *A pseudorandom code for an error model E consists of* $(\mathrm{KeyGen}, \mathrm{Enc}, \mathrm{Dec})$*. For $k \leftarrow \mathrm{KeyGen}$ and $c \leftarrow \mathrm{Enc}_k$, c is computationally indistinguishable from uniform;* Dec_k *accepts any corruption of c allowed by E and rejects any fixed string independent of k.*

These are exactly the three properties we need: pseudorandomness gives undetectability, error correction preserves detection under E, and soundness rejects key-independent strings.

For a generative model $G(x; r)$ with $r \leftarrow \{0, 1\}^n$, assume there is an efficient Recover such that, with overwhelming probability, the recovered string differs from r by no more errors than the PRC can correct. The key idea is to put redundancy in the randomness, not in the text.

Theorem 2 (Watermarking recoverable randomness)**.** *Replacing uniform r by a PRC codeword gives a reliably detectable watermark indistinguishable from the unwatermarked model. It remains robust whenever recovery introduces no more errors than the PRC can correct.*

Proof. Generate $c \leftarrow \mathrm{Encode}_k$ and output $y = G(x; c)$. Pseudorandomness $c \approx r$ gives

$$G(x; c) \approx G(x; r).$$

since applying an efficient map preserves computational indistinguishability. For completeness, $\mathrm{Recover}(y)$ has no more errors than the PRC can correct, so Decode_k accepts. For soundness, fixed y_0 gives fixed $\mathrm{Recover}(y_0)$ independent of k, which Decode_k rejects. The same argument gives robustness whenever an allowed transformation preserves the recovery guarantee. □

The requirements are easy separately but hard to combine. One-way functions give pseudorandom generators, while random linear codes have constant relative distance at constant rate. The challenge is keyed error correction: the key holder must efficiently recognize corrupted codewords while they still look uniform without the key. A generic random linear code has the desired distance but no known polynomial-time decoder for a constant fraction of worst-case errors.

Theorem 3. *Assume LPN is subexponentially hard. Then there exist PRCs robust to a constant fraction of random substitutions.*

Proof. Choose a parity-check matrix $P \in \mathbb{F}_2^{r \times n}$ with row weight w and $G \in \mathbb{F}_2^{n \times g}$ with $PG = 0$; the key is $(P, \mathbf{z})$ for uniform $\mathbf{z} \in \mathbb{F}_2^n$. Encode by sampling $\mathbf{s} \leftarrow \mathbb{F}_2^g$ and $e_i \leftarrow \mathrm{Ber}(\eta)$ independently, and output

$$\mathbf{c} = G\mathbf{s} + \mathbf{z} + \mathbf{e} \pmod 2.$$

For a codeword, $P(\mathbf{c} + \mathbf{z}) = P\mathbf{e}$. Each check is zero with probability $\frac{1}{2} + \frac{1}{2}(1 - 2\eta)^w$, so small w preserves a detectable bias. Random substitutions at rate p replace η by $\eta' = \eta + p - 2\eta p$ and weaken, but do not erase, this bias; a fixed string independent of the key gives about half ones.

If G were uniform, $(G, G\mathbf{s} + \mathbf{e})$ would be LPN samples and hence pseudorandom. The construction chooses G statistically close to uniform while hiding P, so the codewords remain pseudorandom and P gives efficient keyed detection after errors. □

This robustness guarantee is limited: one can often preserve the meaning of a text while moving the recovered PRC word far in Hamming distance, causing detection to fail. For example, suppose schematically that the detector recovers one PRC coordinate from each token:

$$\underset{0}{[\text{This}]}\underset{1}{[\text{ parrot}]}\underset{1}{[\text{ is}]}\underset{1}{[\text{ no}]}\underset{0}{[\text{ more!}]} \qquad \underset{1}{[\text{He}]}\underset{0}{[\text{ has}]}\underset{0}{[\text{ ceased}]}\underset{0}{[\text{ to}]}\underset{1}{[\text{ be!}]}$$

Now swap the two clauses. The meaning is essentially unchanged, while the corresponding recovered coordinates are transposed with the tokens:

$$\underset{1}{[\text{He}]}\underset{0}{[\text{ has}]}\underset{0}{[\text{ ceased}]}\underset{0}{[\text{ to}]}\underset{1}{[\text{ be!}]} \qquad \underset{0}{[\text{This}]}\underset{1}{[\text{ parrot}]}\underset{1}{[\text{ is}]}\underset{1}{[\text{ no}]}\underset{0}{[\text{ more!}]}$$

The Hamming distance is $d_H(01110\,10001, 10001\,01110) = 10$, so a meaning-preserving reordering can look like many substitutions to a decoder designed only for Hamming errors.

There are impossibility results showing that, under strong assumptions on the high-quality output space, no watermark can survive every efficient quality-preserving transformation. Such results rule out robustness to all quality-preserving changes, but leave open the possibility of robustness to changes that are close in *meaning*. Formalizing semantic distance may be elusive, yet strong models appear to capture aspects of semantic similarity implicitly, suggesting they might provide a useful proxy.

4 Watermarking diffusion models

Chapter 5 described diffusion by a sequence of distributions $D_0, D_1, \ldots, D_T$, where $D_0 = D$ is the data distribution and D_T is close to a standard Gaussian. Recall the forward step

$$X_{k+1} = \sqrt{1-\eta}\, X_k + \sqrt{\eta}\, Z_k, \qquad Z_k \sim \mathcal{N}(0, I_n),$$

and the corresponding reverse path

$$\mathcal{N}(0, I_n) \approx D_T \longrightarrow D_{T-1} \longrightarrow \cdots \longrightarrow D_0 = D.$$

A diffusion sampler therefore begins with $X_T \sim \mathcal{N}(0, I_n)$ and uses it to generate an output. For a conditional model with input x, write this simply as

$$y = G(x; X_T).$$

This isolates exactly the randomness that we would like to watermark.

The signs of a standard Gaussian are independent uniform bits and are independent of the magnitudes. Let $c \in \{-1, 1\}^n$ be a pseudorandom codeword and sample $Z \sim \mathcal{N}(0, I_n)$. Instead of starting reverse diffusion from Z, start from

$$\widetilde{X}_{T,i} = c_i |Z_i|.$$

If c were uniform, then $\widetilde{X}_T$ would be distributed exactly as $\mathcal{N}(0, I_n)$. Replacing uniform signs by a pseudorandom codeword is therefore computationally invisible, and so is the resulting output $G(x; \widetilde{X}_T)$. This hiding step is not specific to images; it applies to any diffusion sampler whose initial randomness is Gaussian.

Detection requires recovering the signs of the initial X_T from y. Generic diffusion may be noninvertible or use additional randomness, so this works only when recovery is sufficiently accurate.

In *latent diffusion*, images are first encoded into a lower-dimensional continuous latent space; diffusion runs there, and a decoder maps the final latent back to pixels. Given an image y, one can encode it and approximately invert the latent diffusion trajectory to recover $\widehat{X}_T$, then read $\mathrm{sign}(\widehat{X}_T)$ and decode. This works when the sampler is sufficiently invertible: inversion error simply becomes codeword bit error, which the PRC is designed to tolerate.

Thus we again put redundancy in the generator's randomness and recover enough of it for detection. For images, recovery quality depends empirically on the encoder and inversion procedure, not on the diffusion equations alone.

5 Historical notes

The red–green watermark is due to Kirchenbauer et al. [1]. Aaronson proposed keyed sampling randomness for distortion-free watermarking [2]; Christ, Gunn, and Zamir formalized adaptive undetectability and constructions from one-way functions [3].

Pseudorandom codes and the LDPC/LPN construction above are due to Christ and Gunn [4]. Later work obtains binary PRCs with subexponential pseudorandomness and worst-case edit robustness under the permuted-codes conjecture [5], and robust LLM watermarking at constant per-token entropy under related assumptions [6].

Gunn, Zhao, and Song applied PRCs to latent diffusion images [7]; detection uses diffusion inversion to recover the initial latent, with DDIM inversion [8] as a useful baseline.

The strong-watermarking impossibility result is due to Zhang et al. [9].

References

[1] J. Kirchenbauer, J. Geiping, Y. Wen, J. Katz, I. Miers, and T. Goldstein. A Watermark for Large Language Models. In *ICML*, 2023.

[2] S. Aaronson. My AI Safety Lecture for UT Effective Altruism. *Shtetl-Optimized*, November 2022.

[3] M. Christ, S. Gunn, and O. Zamir. Undetectable Watermarks for Language Models. In *COLT*, 2024.

[4] M. Christ and S. Gunn. Pseudorandom Error-Correcting Codes. In *CRYPTO*, 2024.

[5] M. Christ, N. Golowich, S. Gunn, A. Moitra, and D. Wichs. Improved Pseudorandom Codes from Permuted Puzzles. In *STOC*, 2026.

[6] N. Mazor, A. Morgan, and R. Pass. Can We Watermark Low-Entropy LLM Outputs? In *FORC*, 2026.

[7] S. Gunn, X. Zhao, and D. Song. An Undetectable Watermark for Generative Image Models. In *ICLR*, 2025.

[8] J. Song, C. Meng, and S. Ermon. Denoising Diffusion Implicit Models. In *ICLR*, 2021.

[9] H. Zhang, B. L. Edelman, D. Francati, D. Venturi, G. Ateniese, and B. Barak. Watermarks in the Sand: Impossibility of Strong Watermarking for Generative Models. *arXiv:2311.04378*, 2023.