WaterMod: Modular Token-Rank Partitioning for Probability-Balanced LLM Watermarking

Shinwoo Park[1], **Hyejin Park**[2], **Hyeseon Ahn**[1], **Yo-Sub Han**[1*]

[1]Yonsei University, Seoul, Republic of Korea
[2]Rensselaer Polytechnic Institute, Troy, NY, USA
{pshkhh, hsan, emmous}@yonsei.ac.kr, parkh12@rpi.edu

Abstract

Large language models now draft news, legal analyses, and software code with human-level fluency. At the same time, regulations such as the EU AI Act mandate that each synthetic passage carry an imperceptible, machine-verifiable mark for provenance. Conventional logit-based watermarks satisfy this requirement by selecting a pseudorandom green vocabulary at every decoding step and boosting its logits, yet the random split can exclude the highest-probability token and thus erode fluency. WaterMod mitigates this limitation through a probability-aware modular rule. The vocabulary is first sorted in descending model probability; the resulting ranks are then partitioned by the residue rank $\bmod k$, which distributes adjacent—and therefore semantically similar—tokens across different classes. A fixed bias of small magnitude is applied to one selected class. In the zero-bit setting ($k = 2$), an entropy-adaptive gate selects either the even or the odd parity as the green list. Because the top two ranks fall into different parities, this choice embeds a detectable signal while guaranteeing that at least one high-probability token remains available for sampling. In the multi-bit regime ($k > 2$), the current payload digit d selects the color class whose ranks satisfy rank $\bmod k = d$. Biasing the logits of that class embeds exactly one base-k digit—equivalently $\log_2 k$ bits—per decoding step, thereby enabling fine-grained provenance tracing. The same modular arithmetic therefore supports both binary attribution and rich payloads. Experimental results demonstrate that WaterMod consistently attains strong watermark detection performance while maintaining generation quality in both zero-bit and multi-bit settings. This robustness holds across a range of tasks, including natural language generation, mathematical reasoning, and code synthesis.

1 Introduction

Large language models (LLMs) now draft news copy (Goyal, Li, and Durrett 2022), answer legal queries (Hu et al. 2025), and refactor production code (Cordeiro, Noei, and Zou 2024) with near-human fluency (Achiam et al. 2023). The same realism, however, obscures provenance and amplifies downstream risks—misinformation (Pan et al. 2023), plagiarism (Lee et al. 2023), and data-poisoned training corpora (Sander et al. 2024, 2025). Regulators have begun to respond. The European Union's Artificial Intelligence Act (EU AI Act) requires that outputs generated by general-purpose AI models be clearly identified as such, with disclosure obligations expected to take effect by 2026. Watermarking is among the recommended mechanisms for satisfying this requirement, aligning with principles already established for the disclosure of deepfake content (WilmerHale 2024). Regulatory guidance emphasizes that the disclosure mark should remain resilient to common post-processing operations and be verifiable through algorithmic means (EUAIAct 2024). Industry stakeholders are increasingly converging on watermarking as a practical solution. OpenAI publicly acknowledges an internal text-watermark detector under evaluation for ChatGPT (OpenAI 2024), while Google DeepMind has released SynthID-Text (Dathathri et al. 2024), a watermarking algorithm for LLM-generated text. These efforts reflect a growing consensus that imperceptible identifiers, such as watermarks, offer the most practical pathway toward regulatory compliance (Golowich and Moitra 2024; Hu and Huang 2024; Giboulot and Furon 2024; Li, Li, and Zhang 2024; Pang et al. 2024; Zhou et al. 2024; Fu, Xiong, and Dong 2024; Panaitescu-Liess et al. 2025).

The dominant research thread biases sampling toward a pseudorandom green list of tokens while a detector counts their statistical over-representation. Kirchenbauer et al. (2023) randomly partition the vocabulary into a green list and a red list, and force the decoder to prefer tokens from the green list at each decoding step. While conceptually simple, such random partitioning frequently assigns contextually appropriate tokens to the forbidden set (*i.e.*, the red list), thereby reducing lexical diversity and harming fluency. Chen et al. (2024) mitigate semantic degradation by introducing lexical-redundancy clusters, which help ensure that at least one suitable synonym remains in the green list. They achieve this by clustering synonyms using WordNet (Fellbaum 1998) look-ups or LLM prompting. However, the method relies on external synonym resources and prompt engineering, which introduces issues such as limited dictionary coverage, polysemy-related errors, and prompt sensitivity, ultimately hindering consistency across domains. A parallel line of research (Guo et al. 2024) applies locality-sensitive hashing (LSH) over token embeddings to induce

*Corresponding author.

semantically coherent partitions. While this approach enables clustering based on semantic similarity, it remains susceptible to collision errors and instability arising from hyperplane sensitivity, often leading to brittle behavior and semantic drift. Moreover, most existing approaches rely on zero-bit watermarking, which merely indicates that a text has been generated by an AI model, without embedding any richer information to support provenance tracing. This limitation becomes critical in high-stakes applications—such as tracking leaked fine-tuning data or identifying the specific model instance responsible for generating disinformation—where regulators and service providers require more expressive payloads to ensure traceability.

We propose WaterMod, **Wa**termarking via **T**ok**e**n-**r**ank **Mod**ular Arithmetic, a probability-balanced watermarking framework that replaces heuristic green/red vocabularies with a modular partitioning of the vocabulary.

- **Probability-ranked palette**: At each decoding step, the vocabulary is sorted in descending order based on the conditional probabilities assigned by the model. Since tokens with contiguous ranks are deemed contextually similar and interchangeable by the model, rank-based partitioning naturally preserves high-probability candidates.
- **Modular coloring**: Token ranks are partitioned by rank $\bmod k$: $k = 2$ yields a zero-bit split, while $k > 2$ supports multi-bit embedding.

WaterMod builds entirely on the probability scores the model already produces, so it needs no synonym dictionaries, hashing tricks, or prompt engineering. By partitioning the ranked vocabulary into residue classes, WaterMod deliberately distributes near-synonyms across distinct color groups, ensuring that each decoding step retains a high-probability token within the designated class. The result is a fluent yet verifiable watermark that can be flexibly scaled—from a binary attribution tag to a full multi-bit provenance string—positioning the method for future transparency and disclosure mandates. Our experimental results show that WaterMod reliably embeds detectable watermarks without compromising content quality across a range of tasks—including natural language generation, mathematical reasoning, and code generation—in both zero-bit and multi-bit watermarking settings.

2 Preliminaries

2.1 Text Generation in LLMs

Let $\mathcal{V}$ be the vocabulary ($|\mathcal{V}| = V$) and $x_{<t} = x_{1:t-1}$ the prefix available at time step t. An LLM with parameters θ produces a *logit vector*

$$\boldsymbol{\ell}_t = f_\theta(x_{<t}) \in \mathbb{R}^V,$$

where each entry $\ell_{t,i}$ denotes the unnormalized compatibility score for token $v_i \in \mathcal{V}$. Sampling probabilities are defined by the softmax function:

$$p_\theta\big(x_t = v_i \mid x_{<t}\big) \;=\; \frac{\exp(\ell_{t,i})}{\sum_{j=1}^{V} \exp(\ell_{t,j})}. \tag{1}$$

The decoder draws x_t from equation 1, appends it to the context, and repeats until an end-of-sequence token is produced.

2.2 Logit-based Text Watermarking

Embedding. Logit-based watermarking perturbs $\boldsymbol{\ell}_t$ before sampling. At each step a *green list* $\mathcal{G}_t \subset \mathcal{V}$ and *red list* $\mathcal{R}_t = \mathcal{V} \setminus \mathcal{G}_t$ are determined (typically via a hash seeded by the previous token). Given a bias magnitude $\delta > 0$, the encoder raises the logits of green tokens:

$$\tilde{\ell}_{t,i} = \begin{cases} \ell_{t,i} + \delta, & v_i \in \mathcal{G}_t, \\ \ell_{t,i}, & v_i \in \mathcal{R}_t. \end{cases}$$

Sampling from $p_\theta\big(x_t = v_i \mid x_{<t}\big)$ is thus biased toward $\mathcal{G}_t$; a position is considered watermarked if the generated token at that step falls within $\mathcal{G}_t$.

Detection. Let a generated sequence have length T and let $G = \#\{\text{green tokens}\}$. Under the null hypothesis $\mathcal{H}_0$ (no watermark, i.e. $\delta = 0$), each position is green with a fixed probability

$$\varepsilon \;=\; \frac{|\mathcal{G}_t|}{V} \quad \text{(assumed identical for all } t\text{)},$$

so the total green count G follows a binomial distribution: $G \sim \mathrm{Binom}(T, \varepsilon)$. The detector evaluates the z-score

$$z \;=\; \frac{G - T\varepsilon}{\sqrt{T\varepsilon(1-\varepsilon)}}, \tag{2}$$

which converges to the standard normal distribution under $\mathcal{H}_0$ and shifts to larger values when $\delta > 0$ (*i.e.*, when a watermark is embedded). A one-sided hypothesis test is performed: if z exceeds a pre-specified threshold τ (chosen to attain the desired false-positive rate), $\mathcal{H}_0$ is rejected and the sequence is declared watermarked.

2.3 Modular Arithmetic

For an integer modulus $k \geq 2$, the set of integers decomposes into residue classes

$$[r]_k \;=\; \big\{\, n \in \mathbb{Z} \mid n \equiv r \pmod{k} \big\}, \quad r \in \{0, \ldots, k-1\}.$$

A key advantage of using modular arithmetic lies in its collision-free and deterministic partitioning property. Each token rank (logit-sorted rank) is deterministically mapped to a unique residue class, yielding nearly uniform class sizes that differ by at most one. This partitioning scheme requires no external resources such as lexicons or hash functions, making it efficient to implement.

3 Methods

At each decoding step, tokens are first sorted in descending order of model probability. In the *zero-bit* setting, the vocabulary is partitioned into even- and odd-ranked tokens, which are alternately assigned to the green and red groups. When the distribution is sharp—*i.e.*, most probability mass lies on the top one or two tokens—the entropy gate assigns a low probability to the odd-rank choice, so the even-ranked group is more likely to become green. As the distribution flattens and entropy rises, that probability increases, making the odd-ranked group increasingly likely to be selected instead. This dynamic assignment ensures that at least one

high-probability token remains in the green set, preserving fluency while enabling watermark insertion. Next, a small bias is added to the logits of green tokens to subtly guide sampling. During detection, the presence of a watermark is inferred by testing whether green tokens are statistically overrepresented.

In the *multi-bit* setting, the parity rule is extended to rank $\mathrm{mod}\ k$, which partitions the probability-sorted vocabulary into k color classes. A pseudorandom function (PRF) permutes the payload digits, and at each decoding step the hash of the previous token selects a target position p. The current base-k digit $m = \mathbf{m}[p]$ then determines the class with indices satisfying rank $\mathrm{mod}\ k = m$, and only the logits of those tokens receive the bias δ. Each generated token therefore carries one base-k digit, *i.e.* $\log_2 k$ payload bits in expectation. At detection time, majority voting over the observed color counts recovers every digit; the repeated observations act as a natural form of error correction and maintain robustness even for short passages.

3.1 Zero-bit Watermarking

Probability-sorted parity partition. Algorithm 1 describes the watermark embedding procedure of WaterMod in zero-bit case. Given logits $\boldsymbol{\ell}_t$ the permutation $\pi = \mathrm{argsort}(\boldsymbol{\ell}_t; \downarrow)$ orders the vocabulary by the model probability. Mapping $r \mapsto r \bmod 2$ creates disjoint even and odd classes. Adjacent ranks, which the model views as interchangeable, are distributed across the two classes.

Algorithm 1: ZERO-BIT : Embedding at step t

Input: logits $\boldsymbol{\ell}_t \in \mathbb{R}^V$, previous token x_{t-1}, secret key K, entropy scaling factor H_{scale}, bias δ
Output: next token $\hat{x}_t$

$p_i \leftarrow \mathrm{softmax}(\boldsymbol{\ell}_t)_i$
$H_t \leftarrow -\sum_{i=1}^{V} p_i \log p_i$ ▷ Shannon entropy
$H_{\max} \leftarrow \log_2 V$ ▷ uniform distribution $p_i = \frac{1}{V}$
$p_{\text{odd}} \leftarrow \left(H_t / H_{\max}\right)^{H_{\text{scale}}}$
seed $\leftarrow \mathrm{PRF}(x_{t-1})$ ▷ a pseudorandom seed derived from the previous token
$u \leftarrow \mathrm{Hash2Uniform}(\textit{seed} \oplus K) \in (0,1)$ ▷ uniform random variable u derived from a secret key K
$g \leftarrow \mathbf{1}[\, u < p_{\text{odd}} \,]$ ▷ g=1: tokens with odd ranks are green.
$\pi \leftarrow \mathrm{argsort}(\boldsymbol{\ell}_t; \downarrow)$
for $r = 0$ **to** $V - 1$ **do**
  if $r \bmod 2 = g$ **then**
    $\boldsymbol{\ell}_{t,\pi[r]} \leftarrow \boldsymbol{\ell}_{t,\pi[r]} + \delta$
  end if
end for
$\hat{x}_t \leftarrow \arg\max_j \mathrm{softmax}(\boldsymbol{\ell}_t)_j$
return $\hat{x}_t$

Entropy-driven green-list selection. Given p_i, the Shannon entropy at time step t is defined as:

$$H_t = -\sum_{i=1}^{V} p_i \log p_i, \quad H_{\max} = \log_2 V. \tag{3}$$

The entropy is transformed into a Bernoulli parameter

$$p_{\text{odd}} = \left(\frac{H_t}{H_{\max}}\right)^{H_{\text{scale}}}, \tag{4}$$

where the exponent $H_{\text{scale}} > 0$ controls the steepness of the mapping. Choosing $H_{\text{scale}} > 1$ makes the rise in p_{odd} steeper—almost zero for low-entropy (sharp) distributions and close to one only when the distribution becomes flat—thereby protecting fluency in deterministic contexts. Conversely, $0 < H_{\text{scale}} < 1$ yields a gentler slope, raising p_{odd} even at moderate entropy and embedding the watermark more densely when probability mass is already spread across many tokens. A uniformly distributed value u, deterministically derived from the key, is used to select the green parity via $g = \mathbf{1}[\, u < p_{\text{odd}} \,]$. Under high-entropy conditions, the output distribution of the model tends to be relatively uniform, indicating that probability mass is distributed across multiple candidate tokens and that the second-ranked token is nearly as likely as the top-ranked one. In such cases, assigning the odd-ranked group to the green list enhances watermarking capacity while maintaining textual fluency. In contrast, low-entropy distributions are sharply concentrated on a few top tokens, making the even-ranked group a more stable and semantically reliable choice. Notably, the odd-ranked group still includes several high-probability candidates beyond the top-1 token, allowing watermark insertion without substantially degrading generation quality. The trade-off between watermark strength and fluency can be further adjusted through the entropy scaling parameter H_{scale}.

Logit biasing for watermark insertion. WaterMod raises every logit whose rank satisfies $r \bmod 2 = g$ by the constant δ and samples the next token. The green list is guaranteed to include at least one high-ranked token, thereby preserving fluency during generation. Figure 1 illustrates the watermark embedding process of WaterMod.

Algorithm 2: ZERO-BIT : Sequence-level Detection

Input: token sequence $(x_0, \ldots, x_{T-1})$, generator f_θ, threshold τ, secret key K
Output: TRUE if watermarked, else FALSE

$G \leftarrow 0$ ▷ green-token counter
for $t = 1$ **to** $T - 1$ **do**
  $\boldsymbol{\ell}_t \leftarrow f_\theta(x_{<t})$
  recompute p_{odd}, u, g with $\boldsymbol{\ell}_t, x_{t-1}, K$ ▷ reconstruction of the green list
  $\pi \leftarrow \mathrm{argsort}(\boldsymbol{\ell}_t; \downarrow)$
  $\mathcal{G} \leftarrow \{\pi[r] \mid r \bmod 2 = g\}$
  if $x_t \in \mathcal{G}$ **then**
    $G \leftarrow G + 1$
  end if
end for
$N \leftarrow T - 1$
$z \leftarrow \dfrac{G - \frac{N}{2}}{\sqrt{N/4}}$ ▷ null proportion $\varepsilon = 0.5$
return $\mathbf{1}[\, z > \tau \,]$

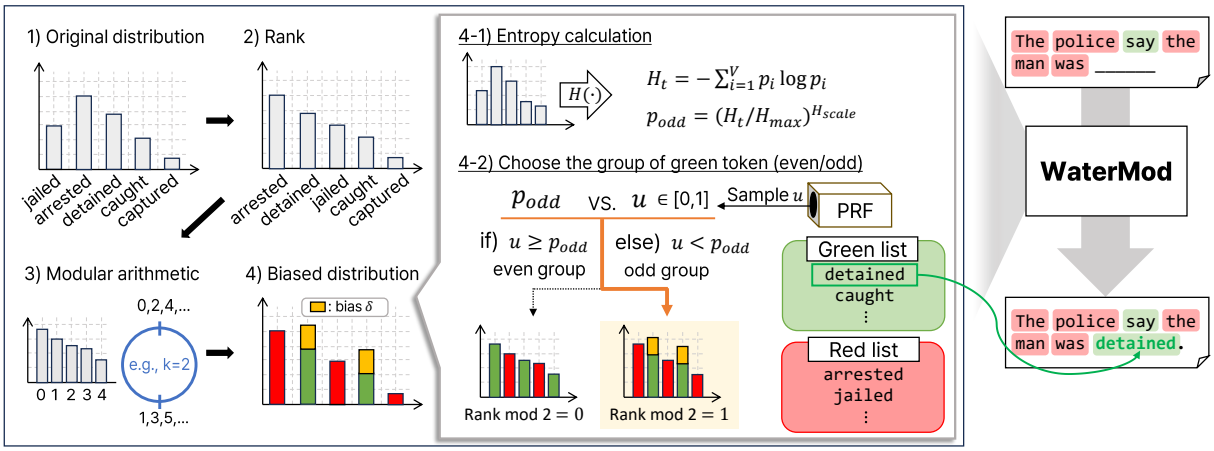

Figure 1: Watermark embedding procedure of WaterMod in the zero-bit setting.

z-score calculation for watermark detection. Algorithm 2 outlines the watermark detection procedure under the zero-bit watermarking scenario. The detector reconstructs the green-list parity for every position using the identical secret key, counts green hits G over $N = T - 1$ positions, and evaluates z-score:

$$z = \frac{G - \frac{N}{2}}{\sqrt{N/4}} \,. \tag{5}$$

A sequence is classified as watermarked when $z > \tau$.

3.2 Extension to Multi-bit Watermarking

k-residue color partition of ranks. Algorithm 3 presents the message embedding procedure of WaterMod under the multi-bit watermarking scenario. Applying the modular rule $r \bmod k$ generalizes the parity split to k color classes,

$$\mathcal{C}_d \;=\; \big\{\, \pi[r] \mid r \bmod k = d \big\}, \quad d \in \{0, \ldots, k-1\}, \tag{6}$$

so every token rank belongs to exactly one residue class $\mathcal{C}_d$.

Algorithm 3: MULTI-BIT: Embedding a b-bit payload at step t

Input: logits $\boldsymbol{\ell}_t$, previous token x_{t-1}, secret key K, bit length b, base k, payload digits $\mathbf{m}$, bias δ
Output: next token $\hat{x}_t$

$\tilde{b} \leftarrow \lceil b / \log_2 k \rceil$ ▷ length of $\mathbf{m}$ in base-k
$seed \leftarrow \mathrm{PRF}(x_{t-1})$
$u \leftarrow \mathrm{Hash2Uniform}(seed \oplus K) \in (0, 1)$
$p \leftarrow \min(\lfloor u\tilde{b} \rfloor, \tilde{b} - 1)$ ▷ pseudorandom position in the digit string
$d \leftarrow \mathbf{m}[p]$ ▷ digit to embed at this step
$\pi \leftarrow \mathrm{argsort}(\boldsymbol{\ell}_t; \downarrow)$ ▷ rank permutation of the vocabulary
for $r = 0$ **to** $V - 1$ **do** ▷ bias *only* the target color
  if $r \bmod k = d$ **then**
    $\boldsymbol{\ell}_{t,\pi[r]} \leftarrow \boldsymbol{\ell}_{t,\pi[r]} + \delta$
  end if
end for
$\hat{x}_t \leftarrow \arg\max_j \mathrm{softmax}(\boldsymbol{\ell}_t)_j$
return $\hat{x}_t$

Payload-conditioned color choice. A b-bit message is represented as the base-k vector:

$$\mathbf{m} \in \{0, \ldots, k-1\}^{\tilde{b}}, \quad \tilde{b} = \left\lceil \tfrac{b}{\log_2 k} \right\rceil. \tag{7}$$

At step t the key hash u selects the pseudorandom position $p = \min(\lfloor u\tilde{b} \rfloor, \tilde{b} - 1)$, and the encoder picks the color $d = \mathbf{m}[p]$.

Digit-wise biasing for message embedding. WaterMod adds a bias δ to the logits of all tokens satisfying $r \bmod k = d$. Since the probability mass is uniformly distributed across color groups, the overall generation quality is preserved.

Algorithm 4: MULTI-BIT: Payload Recovery

Input: sequence $(x_0, \ldots, x_{T-1})$, secret key K, generator f_θ, base k
Output: recovered digits $\hat{\mathbf{m}}$, z-score

$\tilde{b} \leftarrow |\mathbf{m}|$; initialize tallies $C[p][d] \leftarrow 0$ ▷ $C[p][d]$ counts how often color d appears at position p
$G \leftarrow 0$; $T \leftarrow 0$ ▷ G: hits, T: inspected steps
for $t = 1$ **to** $T - 1$ **do**
  $\boldsymbol{\ell}_t \leftarrow f_\theta(x_{<t})$ ▷ recompute logits for step t
  $seed \leftarrow \mathrm{PRF}(x_{t-1})$
  $u \leftarrow \mathrm{Hash2Uniform}(seed \oplus K) \in (0, 1)$
  $p \leftarrow \min(\lfloor u\tilde{b} \rfloor, \tilde{b} - 1)$ ▷ digit position used at step t
  $\pi \leftarrow \mathrm{argsort}(\boldsymbol{\ell}; \downarrow)$
  $r \leftarrow$ index of x_t in π; $d \leftarrow r \bmod k$ ▷ observed color
  $C[p][d] \leftarrow C[p][d] + 1$ ▷ update tallies
  if $d = \mathbf{m}[p]$ **then**
    $G \leftarrow G + 1$
  end if
  $T \leftarrow T + 1$
end for
for $p = 0$ **to** $\tilde{b} - 1$ **do**
  $\hat{m}[p] \leftarrow \arg\max_d C[p][d]$ ▷ majority vote per position
end for
$p_0 \leftarrow 1/k$ ▷ null success probability
$z \leftarrow \dfrac{G - Tp_0}{\sqrt{T\,p_0(1 - p_0)}}$ ▷ z-score
return $\hat{\mathbf{m}}$, z

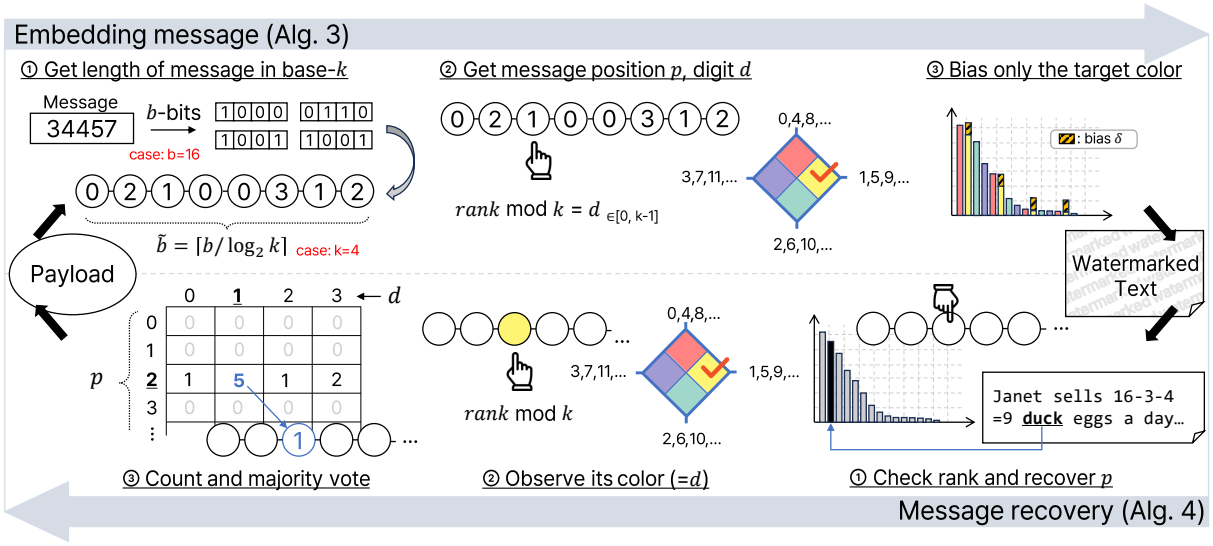

Figure 2: Overview of the message encoding and recovery process in WaterMod under the multi-bit watermarking regime.

z-score calculation for watermark detection. Algorithm 4 describes the message recovery procedure of WaterMod. For each token generated after a fixed-length prefix, the detector performs the following steps: 1) it reconstructs the target color that should have been favored by the WaterMod encoding scheme; 2) it registers a hit if the observed token belongs to the reconstructed color; and 3) it accumulates the total number of hits G over T inspected positions. Under the null hypothesis—*i.e.*, when no watermark is embedded—each color is equally likely to be selected with probability $p_0 = 1/k$. Consequently, the hit count G follows a binomial distribution:

$$G \sim \mathrm{Binom}(T, p_0), \quad z = \frac{G - Tp_0}{\sqrt{T\,p_0(1-p_0)}} \tag{8}$$

The resulting standardized statistic z approximately follows the standard normal distribution $\mathcal{N}(0,1)$ under the null hypothesis. In the presence of a watermark, however, the token distribution becomes biased, increasing the effective success probability beyond p_0 and shifting z toward positive values. A one-sided hypothesis test flags a sequence as watermarked if the computed z-score exceeds a predefined threshold τ. Varying τ yields a receiver operating characteristic (ROC) curve; the corresponding area under the curve (AUROC) quantifies the detection power. Notably, the same color-position tally table $C[p][d]$ used for detection also supports payload recovery via majority vote. Therefore, WaterMod enables both source attribution via the z-score and message retrieval of the embedded digits $\hat{\mathbf{m}}$ in a single decoding pass. Figure 2 provides an overview of the message embedding and recovery processes in WaterMod.

Discussion. A single modular arithmetic on the probability ranking guarantees quality preservation in both watermarking regimes. For $k = 2$ the even–odd split sends near-synonymous tokens to different sides of the green–red boundary, ensuring that every decoding step retains at least one high-probability candidate. For $k > 2$ the same mapping distributes the vocabulary almost uniformly across the k color lists, so multi-bit embedding enjoys the same fluency safeguard. Because all algorithms depend on k only through the residue condition $r \bmod k$, adjusting that single hyper-parameter moves WaterMod seamlessly from binary attribution to a payload capacity of $\log_2 k$ bits per position. This unified framework for zero- and multi-bit watermarking represents the key advance over prior work.

4 Results and Analysis

4.1 Experimental Setup

Datasets. We evaluate WaterMod across three domains:

- **Natural Language Continuation** We use the news-like subset of the Colossal Common Crawl Cleaned corpus (C4) (Raffel et al. 2020). Given the opening fragment of an article, the model completes the remainder, simulating fake news generation. We randomly sample 500 instances for our experiments.
- **Mathematical Reasoning** We adopt the GSM8K (Cobbe et al. 2021) dataset, which consists of 8,000 arithmetic and grade-school-level math problems designed to assess the reasoning capabilities of LLMs. The task requires solving each problem through chain-of-thought reasoning and presenting the final answer. We use the 1,319 instances in the test split.
- **Code Generation** We use the MBPP+ (Liu et al. 2023) dataset, where the goal is to generate Python code that satisfies a given problem description written in natural language. MBPP+ comprises 378 programming problems, each accompanied by around 100 test cases, allowing for rigorous functional evaluation of generated code.

These three domains exhibit differing levels of entropy, defined as the entropy of the token probability distribution during generation. Mathematical reasoning tasks typically ex-

hibit lower entropy than natural language generation, reflecting the deterministic nature of symbolic computation. Likewise, code generation tends to produce lower-entropy distributions due to the rigid syntactic and structural constraints of programming languages. We leverage these variations in entropy across tasks to comprehensively assess the performance of WaterMod.

Baselines. We benchmark WaterMod against five zero-bit schemes and one representative multi-bit scheme.

- **KGW** (Kirchenbauer et al. 2023) randomly partitions the vocabulary into green/red lists once per step and adds a soft logit bonus to green tokens.
- **EXPEdit & ITSEdit** (Kuditipudi et al. 2024) map a pseudorandom seed to a fixed token sequence by exponential minimum or inverse transform sampling.
- **LSH** (Guo et al. 2024) applies locality-sensitive hashing to word embeddings, ensuring that semantically similar tokens are placed in the same green list.
- **SynthID-Text** (Dathathri et al. 2024) leverages tournament sampling, a sampling technique that subtly aligns token choices with seeded random values.
- **MPAC** (Yoo, Ahn, and Kwak 2024) allocates each payload digit to a dedicated token position; at that position the decoder forces a token whose hash equals the digit.

EXPEdit, ITSEdit and SynthID-Text represent sampling-based watermarking approaches, whereas KGW, LSH, MPAC and WaterMod fall under logit-based watermarking methods. We evaluate our method and all baselines on the same Qwen-2.5-1.5B (Yang et al. 2024) model to ensure a fair and controlled comparison. Since these watermarking approaches, including our own, are model-agnostic, their principles are generalizable to other open-source LLMs.

Evaluation Metrics. We evaluate the quality of watermarked outputs using task-specific metrics across various datasets. On C4, we use perplexity, where lower values indicate more fluent text. For GSM8K, we assess accuracy by comparing predicted answers to reference solutions. On MBPP+, we measure pass@1, which calculates the proportion of problems where the generated code passes all test cases. Detection performance is measured by computing the AUROC from z-scores. The z-scores quantify how likely a text is to have been generated by an LLM. Higher z-scores indicate a stronger likelihood, and AUROC evaluates how well the watermarking method distinguishes LLM-generated content from human-written text.

Implementation Details. All methods, including our own and the baseline approaches, use the same configuration. We apply a deterministic decoding strategy, selecting at each time step the token with the highest probability. This eliminates stochastic variability and enables a direct analysis of how watermarking affects model outputs. We limit the maximum number of tokens generated per instance for each dataset as follows: 400 for C4, 600 for GSM8K and MBPP+.

In the zero-bit watermarking scenario, we set the entropy scaling factor $H_{\text{scale}} = 1.2$ when calculating the odd-ranked token selection probability p_{odd}. The entropy scaling factor could be further optimized according to the entropy level of the domain in which the watermark is embedded, potentially leading to performance improvements. We leave the automated discovery of the optimal entropy scaling factor for future work. We configure the watermarking bias δ as 1.0 for the zero-bit setting and 2.5 for the multi-bit setting. The higher bias in the multi-bit setting is necessary to compensate for its smaller target class ($1/k$ of the vocabulary versus $1/2$), ensuring sufficient statistical pressure for reliable embedding. For logit-based watermarking methods under the zero-bit setting, we fix the green list ratio to 0.5. In the multi-bit setting, we embed 16-bit payloads using base $k = 4$. A 16-bit payload was chosen as it offers a practical and expressive message size, capable of encoding over 65,000 unique identifiers for fine-grained provenance tracing. We conduct all experiments on a single NVIDIA RTX 3090 GPU with 24GB of memory.

4.2 Experimental Results

Zero-bit Watermarking. Table 1 compares the performance of WaterMod with five existing watermarking methods in the zero-bit watermarking scenario. On the natural language continuation task (C4), WaterMod achieves the lowest perplexity, indicating the most fluent generation among all watermarking methods. It ranks third in detection performance (AUROC 87.09), trailing only SynthID-Text and LSH. However, the higher AUROC of LSH (88.03) comes at a steep cost in fluency: its perplexity is more than twice as high as that of WaterMod. SynthID-Text emerges as a strong baseline, attaining the second-best perplexity and the highest AUROC on C4. On the GSM8K mathematical reasoning benchmark, WaterMod attains the best accuracy and simultaneously achieves perfect detection performance. Compared to SynthID-Text, the strongest competing method in terms of AUROC, WaterMod improves accuracy by 13.06%, suggesting that the modular operation over token probability ranks ensures that the most probable or second most probable token is consistently selected, even under watermark embedding. For code generation on MBPP+, WaterMod delivers the best watermark detectability and the second-best pass@1 score. It outperforms the next-best AUROC baseline, KGW, by a substantial 14.12% while simultaneously improving pass@1 by 23.07%. Compared to LSH, which yields the highest pass@1 score while exhibiting an extremely weak watermarking signal (AUROC 30.72), WaterMod improves AUROC by an impressive 169.07%, indicating that the LSH-based method fails to reliably embed the watermark in this low-entropy case.

Practical deployment of watermarking methods requires the embedded signal to remain resilient to adversarial rewriting of the text. We therefore test WaterMod against a paraphrasing attack carried out by ChatGPT (`gpt-4o-2025-04-14`). For every GSM8K sample we supply the WaterMod-marked solution to the assistant with the prompt *"Please paraphrase the following text:"*. The human–written solutions are left untouched, and the same z-score detector is applied to all passages. Table 2 shows that paraphrasing reduces the mean z-score of WaterMod outputs from 14.89 to 9.95, yet the margin relative

Method	C4		GSM8K		MBPP+	
	Perplexity	AUROC	Accuracy	AUROC	Pass@1	AUROC
EXPEdit	36.35	36.90	10.84	37.37	22.80	34.38
ITSEdit	31.03	11.29	11.75	35.44	20.10	27.40
KGW	21.96	80.83	51.78	44.38	29.90	<u>72.43</u>
LSH	26.19	<u>88.03</u>	<u>53.07</u>	52.63	**41.30**	30.72
SynthID-Text	<u>12.77</u>	**94.36**	47.61	<u>97.65</u>	27.80	66.90
WaterMod	**12.58**	87.09	**53.83**	**100**	<u>36.80</u>	**82.66**

Table 1: A comparative evaluation of different watermarking methods under the zero-bit watermarking scenario. We highlight the best-performing result for each evaluation metric in bold, and the second-best result with underlining.

Source	**Mean z-score**	**AUROC (%)**
Human-written text	0.09	—
WaterMod (no attack)	14.89	100.00
WaterMod (ChatGPT paraphrase)	9.95	99.95

Table 2: Detection robustness on GSM8K under a CHATGPT paraphrasing attack.

Method	C4		GSM8K		MBPP+	
	Perplexity	AUROC	Accuracy	AUROC	Pass@1	AUROC
MPAC	10.88	97.78	31.77	95.05	20.60	48.40
WaterMod	**10.87**	**98.02**	**40.33**	**96.94**	**26.20**	**98.29**

Table 3: A comparative evaluation of different watermarking methods under the multi-bit watermarking scenario.

to human text (0.09) remains large. The AUROC consequently drops by only 0.05 absolute to 99.95%. Because the paraphraser must preserve mathematical correctness, many high-rank tokens remain unreplaced. As a result, most rank-adjacent alternatives retain their intended color, allowing the watermark to survive. WaterMod thus maintains near-perfect detection even under strong rewriting attacks.

Multi-bit Watermarking. Table 3 compares WaterMod against MPAC, a recent state-of-the-art multi-bit watermarking approach. On all three tasks, WaterMod achieves superior performance in both generation quality and detection. In the C4 task, WaterMod slightly improves over MPAC in perplexity and achieves a higher AUROC, underscoring its ability to hide payload bits without degrading fluency. In GSM8K, WaterMod significantly enhances accuracy by 26.94%, while maintaining strong watermark detectability, exceeding that of MPAC. This difference is most pronounced on MBPP+, where WaterMod achieves an AUROC of 98.29, more than double that of MPAC (48.40). This 103.07% relative improvement in detectability demonstrates that WaterMod remains highly effective even in low-entropy, syntactically rigid settings like code. Additionally, its pass@1 score improves over MPAC by 27.18%, indicating that watermark insertion does not compromise program correctness.

5 Related Work

LLM-integrated watermarking encodes watermark signals directly within the text generation process. Early zero-bit approaches introduce subtle biases into the model logits or sampling distributions to probabilistically flag LLM-generated outputs (Kirchenbauer et al. 2023; Dathathri et al. 2024). Subsequent semantic-aware methods improve fluency and detection robustness by refining the partitioning of the vocabulary—typically into green and red token sets—based on semantic properties (Hou et al. 2024). More recent multi-bit extensions enable richer payload encoding within model outputs. These include bit-string allocation and nested-list biasing strategies, as employed in depth watermarking (Yoo, Ahn, and Kwak 2024; Li, Bai, and Cheng 2024). These methods, however, continue to exhibit inherent trade-offs between watermark capacity and text quality.

6 Conclusion

WaterMod unifies zero-bit attribution and multi-bit payload embedding through a simple rank $\mathrm{mod}\ k$ rule applied to probability-sorted token ranks. Experiments across natural language, mathematical reasoning, and code generation tasks demonstrate that this probability-balanced bias preserves output quality while achieving state-of-the-art detection performance.

Acknowledgments

This work was supported by the NRF grant (RS-2025-00562134) and the AI Graduate School Program (RS-2020-II201361) funded by the Korean government.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*.

Chen, L.; Bian, Y.; Deng, Y.; Cai, D.; Li, S.; Zhao, P.; and Wong, K.-F. 2024. WatME: Towards Lossless Watermarking Through Lexical Redundancy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*.

Cobbe, K.; Kosaraju, V.; Bavarian, M.; Chen, M.; Jun, H.; Kaiser, L.; Plappert, M.; Tworek, J.; Hilton, J.; Nakano, R.; et al. 2021. Training Verifiers to Solve Math Word Problems. *arXiv preprint arXiv:2110.14168*.

Cordeiro, J.; Noei, S.; and Zou, Y. 2024. An Empirical Study on the Code Refactoring Capability of Large Language Models. *arXiv preprint arXiv:2411.02320*.

Dathathri, S.; See, A.; Ghaisas, S.; Huang, P.-S.; McAdam, R.; Welbl, J.; Bachani, V.; Kaskasoli, A.; Stanforth, R.; Matejovicova, T.; et al. 2024. Scalable Watermarking for Identifying Large Language Model Outputs. *Nature*.

EUAIAct. 2024. EU AI Act Compliance Checker. https://artificialintelligenceact.eu/assessment/eu-ai-act-compliance-checker/.

Fellbaum, C. 1998. *WordNet: An Electronic Lexical Database*. MIT press.

Fu, Y.; Xiong, D.; and Dong, Y. 2024. Watermarking Conditional Text Generation for AI Detection: Unveiling Challenges and a Semantic-aware Watermark Remedy. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Giboulot, E.; and Furon, T. 2024. WaterMax: breaking the LLM watermark detectability-robustness-quality trade-off. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Golowich, N.; and Moitra, A. 2024. Edit Distance Robust Watermarks via Indexing Pseudorandom Codes. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Goyal, T.; Li, J. J.; and Durrett, G. 2022. News Summarization and Evaluation in the Era of GPT-3. *arXiv preprint arXiv:2209.12356*.

Guo, Y.; Tian, Z.; Song, Y.; Liu, T.; Ding, L.; and Li, D. 2024. Context-aware Watermark with Semantic Balanced Green-red Lists for Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hou, A.; Zhang, J.; He, T.; Wang, Y.; Chuang, Y.-S.; Wang, H.; Shen, L.; Van Durme, B.; Khashabi, D.; and Tsvetkov, Y. 2024. SemStamp: A Semantic Watermark with Paraphrastic Robustness for Text Generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Hu, Y.; Gan, L.; Xiao, W.; Kuang, K.; and Wu, F. 2025. Fine-tuning Large Language Models for Improving Factuality in Legal Question Answering. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*.

Hu, Z.; and Huang, H. 2024. Inevitable Trade-off between Watermark Strength and Speculative Sampling Efficiency for Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Kirchenbauer, J.; Geiping, J.; Wen, Y.; Katz, J.; Miers, I.; and Goldstein, T. 2023. A Watermark for Large Language Models. In *International Conference on Machine Learning (ICML)*.

Kuditipudi, R.; Thickstun, J.; Hashimoto, T.; and Liang, P. 2024. Robust Distortion-free Watermarks for Language Models. *Transactions on Machine Learning Research*.

Lee, J.; Le, T.; Chen, J.; and Lee, D. 2023. Do Language Models Plagiarize? In *Proceedings of the ACM Web Conference (WWW*, 3637–3647.

Li, L.; Bai, Y.; and Cheng, M. 2024. Where Am I From? Identifying Origin of LLM-generated Content. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Li, X.; Li, G.; and Zhang, X. 2024. Segmenting Watermarked Texts From Language Models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Liu, J.; Xia, C. S.; Wang, Y.; and Zhang, L. 2023. Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Pan, Y.; Pan, L.; Chen, W.; Nakov, P.; Kan, M.-Y.; and Wang, W. Y. 2023. On the Risk of Misinformation Pollution with Large Language Models. *arXiv preprint arXiv:2305.13661*.

Panaitescu-Liess, M.-A.; Che, Z.; An, B.; Xu, Y.; Pathmanathan, P.; Chakraborty, S.; Zhu, S.; Goldstein, T.; and Huang, F. 2025. Can Watermarking Large Language Models Prevent Copyrighted Text Generation and Hide Training Data? In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Pang, Q.; Hu, S.; Zheng, W.; and Smith, V. 2024. No Free Lunch in LLM Watermarking: Trade-offs in Watermarking Design Choices. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*.

Sander, T.; Fernandez, P.; Durmus, A.; Douze, M.; and Furon, T. 2024. Watermarking Makes Language Models Radioactive. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sander, T.; Fernandez, P.; Mahloujifar, S.; Durmus, A.; and Guo, C. 2025. Detecting Benchmark Contamination Through Watermarking. *arXiv preprint arXiv:2502.17259*.

WilmerHale. 2024. Navigating Generative AI under the European Union's Artificial Intelligence Act. https://www.wilmerhale.com/en/insights/blogs/wilmerhale-privacy-and-cybersecurity-law/20241002-navigating-generative-ai-under-the-european-unions-artificial-intelligence-act.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; et al. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

Yoo, K.; Ahn, W.; and Kwak, N. 2024. Advancing Beyond Identification: Multi-bit Watermark for Large Language Models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*.

Zhou, T.; Zhao, X.; Xu, X.; and Ren, S. 2024. Bileve: Securing Text Provenance in Large Language Models Against Spoofing with Bi-level Signature. In *Advances in Neural Information Processing Systems (NeurIPS)*.