# Lagrange Interpolation Based Multi-bit Watermark for Large Language Models

**Anonymous ACL submission**

## Abstract

The rapid advancement of LLMs (Large Language Models) has established them as a foundational technology for many AI- and ML-powered human–computer interactions. A critical challenge in this context is the attribution of LLM-generated text — for example, identifying the specific language model that generated it or the individual user who prompted the model. This capability is essential for combating misinformation, fake news, misinterpretation, and plagiarism. One of the key techniques for addressing this challenge is digital watermarking. This work presents a watermarking scheme for LLM-generated text based on Lagrange interpolation, enabling the recovery of a multi-bit watermark even when the text has been redacted by an adversary. The core idea is to embed a continuous sequence of points $(x, f(x))$ that lie on a single straight line. During extraction, the algorithm recovers the original points along with many spurious ones, forming an instance of the Maximum Collinear Points (MCP) problem, which can be solved efficiently. Experimental results demonstrate that the proposed method is scalable and effective, allowing the embedding of a multi-bit watermark.

## 1 Introduction

Artificial Intelligence (AI) and Machine Learning (ML) have advanced significantly in recent years and have given rise to powerful models based on deep learning. In particular, Deep Neural Networks (DNNs) have enabled machines to perform complex cognitive tasks such as image recognition, speech processing, and natural language understanding with accuracy almost equivalent to human performance. A pivotal advancement in this domain is the emergence of Large Language Models (LLMs). They can be seen as deep learning architectures trained on massive textual datasets, which can generate coherent, contextually relevant, and human-like text. LLMs operate by learning rich representations of language that capture its semantic, syntactic, and pragmatic patterns. These models, often based on the Transformer architecture, are pre-trained on diverse corpora and fine-tuned for specific tasks such as question answering, summarisation, translation, and dialogue. Their capabilities stem from their scale, both in terms of parameters and data, which allows them to generalise across a wide range of linguistic contexts.

The growing adoption of LLMs offers a wide array of practical benefits, including automated text correction, assistance with writing, and content refinement. However, alongside these advantages lies a more troubling aspect: the potential misuse of LLMs. One such concern arises when individuals present AI-generated content as their own, falsely claiming authorship. This challenge naturally raises an important question: how can we embed a watermark into LLM-generated text that allows a recipient to reliably trace its origin, whether to a specific user or to the LLM system itself? The objective of such watermarking is to make it possible to attribute the text to its true source, thereby establishing authorship beyond reasonable doubt. This capability is especially critical in combating misinformation, propaganda, automated plagiarism, and the spread of fake news.

## 2 Overview and Taxonomy of Watermarking in LLMs

The primary motivation behind watermarking has long been the need to verify that a document or a digital artifact originates from its rightful source (Cox et al., 2007). Classical watermarking enables creators to prove ownership of multimedia content beyond a reasonable doubt. In the era of generative AI and LLMs, this requirement is more urgent than ever, due to the growing difficulty in distinguishing between human and machine gener-

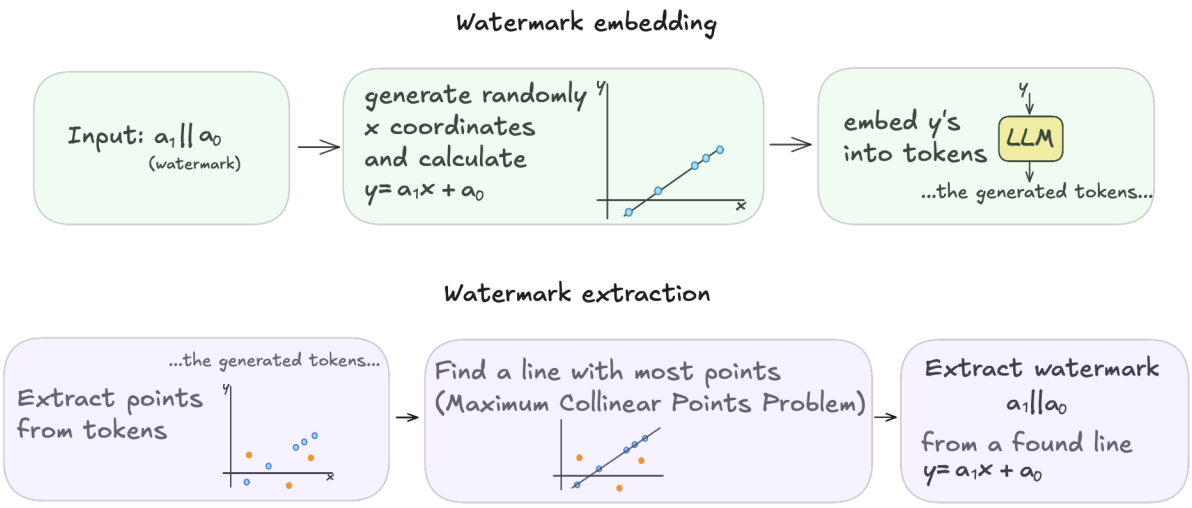


Figure 1: An overview of our watermarking scheme in its basic variant. A multi-bit watermark is represented by two numbers, $a_1$ and $a_0$, which are used as the coefficients of a linear function $y = a_1 x + a_0$. We pseudo-randomly generate many $x$-coordinates and compute the corresponding $y$-coordinates. These $y$-values are then embedded into a generated text by biasing the token selection. In the extraction phase, we recover the embedded points from the generated text. Due to sampling variability during generation and/or adversarial tampering with the text, some of the recovered points may be incorrect (orange points). Our goal is to find the line that passes through the largest number of points, a problem formally known as the Maximum Collinear Points Problem. Once this line is found, its coefficients $a_1$ and $a_0$ constitute the extracted watermark.

ated texts. Watermarking for LLMs serves several essential purposes: (1) authenticating the origin of text, (2) preventing unauthorised reuse or plagiarism, (3) identifying misuse (e.g., misinformation, impersonation), and (4) establishing accountability in high-stakes domains (e.g., education, journalism, code generation). Recent literature has explored a diverse range of strategies for embedding and detecting watermarks in LLM-generated content. Generally, these approaches can be divided into two categories:

*Watermarking During Training* – In this approach, the watermark is embedded by modifying the model training data or objectives, so that identifiable patterns are learned and can be later detected. In the work (Sander et al., 2024), the authors introduced the notion of text radioactivity. It allows to detect whether an LLM has memorised radioactive (marked) data points. TextMarker described in (Liu et al., 2023b) uses backdoor-based membership inference to embed private ownership information. The scheme (Cui et al., 2025) injects plausible yet fictitious knowledge into training corpora to create semantically valid but watermarkable patterns.

*Watermarking at the Logit Level* – the output probability distribution (logits) is subtly perturbed before sampling tokens, embedding statistical patterns without altering the model weights. The work (Kirchenbauer et al., 2023) splits the collection of tokens into green and red subsets and biases the token sampling towards green tokens. Wong et. al. (Wong et al., 2025) describes their logits-to-text watermarking, which jointly optimises encoder-decoder architecture for high-fidelity and high-detectability watermarks. The GumbelSoft watermarking described in (Fu et al., 2024) improves sampling diversity while maintaining robust detection by refining Gumbel-based logits control.

For a comprehensive survey of LLM watermarking techniques, readers are referred to the review by Liu et al. (Liu et al., 2023a). In this paper we are particularly interested in multi-bit watermarks, where embedded watermark consists of more information than just a binary signal (watermarked vs. non-watermarked).

### 2.1 Contribution

We introduce a simple, training-free and model-agnostic watermark for LLM-generated text that encodes a multi-bit payload as the coefficients of a

polynomial over $GF(2^n)$. The watermark extraction reduces to finding a maximal collinear subset of the recovered points followed by Lagrange interpolation. We provide a transparent probabilistic analysis that upper-bounds the chance of spurious collinearities in $(2^n)^2$, yielding design guidance for field size and block length, and we quantify how many points must survive noise/adversarial edits for reliable recovery (we do not claim full robustness to arbitrary edits). To mitigate bit-level noise without changing the generator, we add a lightweight *bit-correction* step at extraction that enumerates candidates within a small Hamming radius, trading runtime for recall in a controlled way. We instantiate the scheme with linear $f$ and report an empirical study on two open LLMs. The results show that high match rates are achievable at modest bias and that runtime scales predictably. Finally, we outline extensions for composing longer payloads via multiple lines and for higher-degree polynomials. These are presented as design options and left for comprehensive empirical evaluation in future work.

## 3 Preliminaries

### 3.1 Lagrange Interpolation over $GF(2^n)$

In the proposed watermarking scheme, we use Lagrange interpolation to recover a function from a set of distinct points.

Formally, Lagrange interpolation over a finite field $GF(2^n)$ constructs a unique polynomial $f(x) \in GF(2^n)[x]$ of degree at most $t-1$ that passes through a given set of $t$ distinct points $(x_1, y_1), \ldots, (x_t, y_t)$, where all $x_i \in GF(2^n)$ and $x_i \neq x_j$ for $i \neq j$. The interpolating polynomial is given by:

$$f(x) = \sum_{j=1}^{t} y_j \cdot \ell_j(x),$$

where each $\ell_j(x) \in GF(2^n)[x]$ is a *Lagrange basis polynomial* defined as:

$$\ell_j(x) = \prod_{\substack{1 \leq m \leq t \\ m \neq j}} \frac{x - x_m}{x_j - x_m}.$$

All arithmetic operations are performed in the field $GF(2^n)$, including the inverses $(x_j - x_m)^{-1}$. The basis polynomials satisfy $\ell_j(x_i) = \delta_{ij}$, ensuring that $f(x_i) = y_i$ for all $i$, where $\delta_{ij}$ denotes the Kronecker delta ($\delta_{ij} = 1$ if $i = j$ and 0 otherwise). This form of interpolation is widely used in finite field cryptography, such as in Shamir's secret sharing, error-correcting codes, and information dispersal schemes.

### 3.2 Embedding Bits into Tokens

Kirchenbauer et al. (Kirchenbauer et al., 2023) introduced the concept of "Green" and "Red" tokens, where increasing the logit values of Green tokens leads to a strong preference for them — a hallmark of watermarked text. In this paper, we use a similar method; however, instead of preferring only green tokens, we select either a green or red token depending on a bit value (green corresponds to 1, red to 0). In this way, we embed a binary vector (watermark) into the stream of generated tokens.

## 4 Watermarking scheme based on Lagrange Interpolation

In the proposed watermarking scheme, the watermark $W$ constitutes coefficients of the polynomial $f(x)$. We focus on the simplest nontrivial case where $f(x)$ is a linear function $f(x) = a_0 + a_1 x \in GF(2^n)$, where the watermark is encoded as $W = (a_0 || a_1) \in GF(2^n)$. We generate a set of points, each of which satisfies the polynomial $f(x)$. The $x$-coordinates are generated pseudo-randomly and the corresponding $y$-coordinates are embedded into tokens (as described in the previous section).

During the extraction, we solve Maximum Collinear Points Problem to identify the largest set of points, which are collinear (i.e., lie on the same line). Then we apply the Lagrange interpolation to those points to recover the original $f(x)$.

A desirable feature of the proposed scheme is that even if only a fraction of embedded points survive tampering, they suffice to recover the watermark. The scheme is naturally resilient to partial modifications, making it suitable for adversarial environments where text may be edited with the intent to obscure provenance.

**Watermark Embedding** Algorithm 1 describes the process of embedding a multi-bit watermark into text generated by an LLM. The watermark is encoded as a linear polynomial $f(x) = a_1 x + a_0$, where the coefficients $a_0$ and $a_1$ together form the watermark. The algorithm begins by initializing the text generation with a given prompt, and then iteratively embeds the watermark across the generated tokens. In each iteration, a pseudo-random $x$-coordinate is computed using a cryptographic

**Algorithm 1** Watermark Embedding

**Input:** large language model $LLM$, vocabulary $\mathcal{V}$, $2n$-bit watermark $W = (a_0||a_1)$, bias $\delta$, prompt $s^{(-N_p)}, \ldots, s^{(-1)}$, secret key $\mathcal{K}$, hash function $H$
**Output:** generated tokens $\mathcal{T}$ with embedded watermark $W$

$f(x) = a_1 x + a_0$ ▷ specify a polynomial based on the watermark
$\mathcal{T} = s^0 = LLM(s^{(-N_p)}, \ldots, s^{(-1)})$ ▷ generate the first token
$t = 1$ ▷ token counter
**while** (end of text generation) **do**
  $x = (x_0, x_1, \ldots, x_{n-1}) = H(s^{t-1}, \mathcal{K})$ ▷ generate pseudo-random $x$-coordinate
  $y = f(x) = (y_0, y_1, \ldots, y_{n-1})$ ▷ compute $y$-coordinate
  **for** $j = 0$ to $n$ **do** ▷ a loop embedding $y$-coordinate into $n$ consecutive tokens
    $seed = H(s^{t-1+j}, \mathcal{K})$
    Using a $seed$, pseudo-randomly split vocabulary $\mathcal{V}$ into two sets $\mathcal{V}_0$, $\mathcal{V}_1$
    Apply $LLM$ to prior tokens $s^{(-N_p)} \ldots s^{(t-1+j)}$ to get a logit vector $v$ over the vocabulary $\mathcal{V}$:
    $v = LLM(s^{(-N_p)}, \ldots, s^{(t-1+j)})$
    **if** $y_j == 0$ **then** $\hat{v}[w] = v[w] + \delta$ for all $w \in \mathcal{V}_0$ ▷ conditionally bias the logits
    **else** $\hat{v}[w] = v[w] + \delta$ for all $w \in \mathcal{V}_1$
    **end if**
    Sample the token $s^{(t+j)}$ from biased logits $\hat{v}$ and append it to $\mathcal{T}$
  **end for**
  $t = t + n$
**end while**
**return** $\mathcal{T}$

hash function applied to the previous token and a secret key. This $x$-value is then used to evaluate the watermark polynomial, producing a corresponding $y$-value that represents the next set of bits to embed. For each bit in this $y$-value, the vocabulary is pseudo-randomly partitioned into two subsets, one corresponding to a bit value of 0 and the other to 1. The LLM generates a probability distribution (logits) over the vocabulary, which is then selectively biased toward the appropriate subset based on the bit being embedded. Then, a new token is sampled (greedy sampling) from the biased logits and added to a sequence of the generated tokens $\mathcal{T}$. The process repeats for each $y$-value, with each one embedded across a fixed number of consecutive tokens, until the desired length of text is generated.

**Watermark Extraction** Algorithm 2 describes the procedure for extracting a watermark from a sequence of tokens generated by a large language model (LLM), assuming the text was previously watermarked using the embedding method outlined in Algorithm 1.

For each group of $n$ consecutive tokens, a pseudo-random $x$-coordinate is derived using a cryptographic hash function $H$ applied to the previous token and a secret key $K$. This process mimics the original embedding and ensures that the same $x$-values are regenerated during extraction.

For each such $x$-value, the algorithm recovers a corresponding $y$-value by processing $n$ consecutive tokens. Using a deterministic hash-based seed, the vocabulary is split into two subsets: $V_0$ and $V_1$. The current token is compared against these subsets to infer a single bit of the $y$-coordinate. If the token belongs to $V_0$, the corresponding bit is set to zero; otherwise, it is set to one. This procedure is repeated for $n$ tokens to fully reconstruct the $n$-bit $y$-coordinate.

Each recovered point $(x, y)$ is added to a collection $P$. After processing the entire token sequence in this manner, the algorithm attempts to identify the original watermark by solving the Maximum Collinear Points (MCP) problem on the set $P$. This step finds the largest subset of points that lie on a single straight line in the two-dimensional plane.

Once this maximal collinear subset is identified, Lagrange interpolation is applied to these points to reconstruct the polynomial $f(x)$. The coefficients of this polynomial, namely $a_0$ and $a_1$, are then returned as the extracted watermark $W = (a_0||a_1)$.

**Algorithm 2** Watermark Extraction

**Input:** sequence of tokens $s^0, s^1, \ldots, s^N$, size of the watermark (in bits) $2n$, vocabulary $\mathcal{V}$, secret key $\mathcal{K}$, hash function $H$, threshold $\tau$
**Output:** watermark $W = (a_0 \| a_1)$ or $\emptyset$ (non-watermarked text)

$\mathbb{P} = \varnothing$ ▷ Create an empty list of points
$t = 1$ ▷ token counter
**while** (end of tokens) **do**
  $x = (x_0, x_1, \ldots, x_{n-1}) = H(s^{t-1}, \mathcal{K})$ ▷ extract $x$-coordinate
  **for** $j = 0$ to $n$ **do** ▷ a loop extracting $y$-coordinate of a single point
    $seed = H(s^{t-1+j}, \mathcal{K})$
    Using a $seed$, pseudo-randomly split vocabulary $\mathcal{V}$ into two sets $\mathcal{V}_0$, $\mathcal{V}_1$
    **if** $s^t \in \mathcal{V}_0$ **then** $y_j = 0$ ▷ extract $y_j$ bit of $y$-coordinate
    **else** $y_j = 1$
    **end if**
  **end for**
  $y = (y_0, y_1, \ldots, y_{n-1})$
  Append $(x, y)$ point to the list $\mathbb{P}$
  $t = t + n$
**end while**
Find the maximum collinear points (MCP) in $\mathbb{P}$
Apply the Lagrange interpolation to maximum collinear points to obtain $f(x) = a_0 + a_1 x$
**if** MCP $\geq \tau$ **then return** $W = (a_0 \| a_1)$
**else return** $\emptyset$

### 4.1 Maximum Collinear Points Problem

During the watermark embedding, a selected set of tokens is biased depending on the value of a watermark bit. However, due to the stochastic nature of sampling, there is no guarantee that a token from the desired set will be selected. As a result, the embedding process may introduce errors. Another potential source of errors is adversarial tampering with the text, such as adding or deleting tokens. Consequently, the verifier is faced with the task of identifying the genuine points (i.e., valid $x$ and $y$ coordinates), which can be formally framed as an instance of the Maximum Collinear Points (MCP) problem. The MCP problem is defined as follows: *Instance*: A set $\mathbb{S}$ of $N$ points in the 2D plane, where each point $p_i = (x_i, y_i)$; $i = 1, 2, \ldots, N$. *Question*: What is the largest subset $\mathbb{S}' \subset \mathbb{S}$ of points that are collinear (i.e., lie on a single line)?

There are many efficient algorithms that solve the problem. The basic ones are as follows:

- Brute force approach — the algorithm checks all triplets for collinearity. Its time and space complexities are $\mathcal{O}(N^3)$ and $\mathcal{O}(1)$, respectively (see (Cormen et al., 2022)).
- Hashing-based approach — the algorithm is similar to the sorting-based one, except that instead of sorting, it applies hashing to store slopes, reducing the time complexity to $\mathcal{O}(N^2)$ (see (Cormen et al., 2022)).

Also there is a convex-hull approach (Preparata and Shamos, 1985) and sorting-based method (Cormen et al., 2022).

Let us consider the Hashing-based algorithm (see Algorithm 4). It determines the largest subset of collinear points among $N$ given points in a 2D plane. The algorithm iterates through each point, treating it as a reference. For every other point, it computes the slope relative to the reference and stores the slope count in a hash map. This hash map allows for quick lookups and updates, ensuring that collinear points are efficiently counted. The maximum count of any slope within the hash map, plus the reference point itself, gives the largest number of collinear points for that reference. By iterating through all points, the algorithm determines the global maximum. Note that vertical lines can be ignored as they are not expressed as a linear function $y = a_1 x + a_0$.

**Bit correction.** In practice, recovering the $n$-bit $y$ from $n$ consecutive tokens is noisy due

to (i) stochastic sampling that can select a token outside the intended subset, (ii) local edits/insertions/deletions that perturb nearby bits. To increase the chance that a true point $(x, y)$ survives verification, we apply a lightweight bit-correction step at extraction, see Appendix for details.

## 5 Security Evaluation

Once a text is generated, an adversary can insert, delete or substitute some tokens. Instead of analyzing each attack separately, we approach the problem in a more analytical way. We aim to determine the threshold beyond which a watermark cannot be extracted from the LLM-generated text. During extraction, we obtain $N = F + R$ number of points of, where $F$ are valid points that lie on the original straight line $f(x)$ and $R$ are spurious points resulting from two sources: (1) noise introduced by the extraction process itself, and (2) modifications introduced by the adversary. It is reasonable to assume that the $R$ points are randomly scattered on the plane $GF(2^n) \times GF(2^n)$. Clearly, the watermark extraction is successful if any subset of random points forming a straight line is no larger than $F$.

First, consider the following trivial facts about straight lines on the $GF(2^n) \times GF(2^n)$ plane:

- Any two points uniquely determine a straight line. This means that there are $2^{2n}$ straight lines.
- Any two straight lines can either intersect at a single point or be parallel with no points in common.
- For each point, there are $2^n$ straight lines that contain that point.

The above properties show that for the number of random points $R$ larger than $2^n$, the correlation among the straight lines grows. Consequently, finding the precise probability distribution of the number of random points that belong to different straight lines is a challenging problem.

**Theorem 1 (Random collinearity in $\mathbf{GF(2^n)^2}$)** *Let $q = 2^n$. Let $S = \{p_1, \ldots, p_R\}$ be $R$ points sampled i.i.d. uniformly from $\mathrm{GF}(q) \times \mathrm{GF}(q)$, and consider only non-vertical lines of the form $y = ax + b$ with $a, b \in \mathrm{GF}(q)$. For any integer $k \geq 3$, the probability that there exists a non-vertical line containing at least $k$ points of $S$ satisfies*

$$\Pr\left[\exists\, k\text{-collinear points}\right] \;\leq\; \binom{R}{k} q^{-(k-2)}.$$

*A proof of the theorem is given in Appendix.*

Theorem 1 provides us with a tool to evaluate the probability that Algorithm 2 extracts the watermark correctly. For a *fixed* non-vertical line $\ell$, the probability that it contains exactly $k$ of the $R$ points is

$$\Pr\left[|\ell \cap S| = k\right] \;=\; \binom{R}{k}\left(\frac{1}{q}\right)^k\left(1 - \frac{1}{q}\right)^{R-k}.$$

**Example 1** *Consider a toy example, where calculations are done in $GF(2^8)$, i.e. $n = 8$. We have $F = 4$ points lying on a straight line and $R = 16$ random points. The probability that four random points belong to a straight line is*

$$\binom{16}{4}\left(\frac{1}{256}\right)^2\left(1 - \frac{1}{256}\right)^{12} \;\approx\; 0.0265,$$

*Consequently, the probability of correct recovery of the watermark is $\approx 0.98$.*

## 6 Experimental Results

We compared our approach with other top five methods. The results of the experiments are given in Table 1. The the experiment setting is identical to the one used in (Qu et al., 2024). A watermarked text length is around 200 tokens. Texts were generated with Llama 2-7B model and prompts were taken from the OpenGen dataset (Krishna et al., 2023). All experiments used a consistent logit bias of $\delta = 6$. For longer watermarks (24 and 32 bits), we use the bit correction mechanism with $c = 1$. The threshold $\tau$ was selected to obtain False Positive Rate FPR$\leq 1\%$.

For two baseline methods, extraction time grows exponentially with watermark size and becomes infeasible in practice (denoted as 'NA').

In essence, the match rates of the presented method are very close to the state-of-the-art (Qu et al., 2024; Fernandez et al., 2023) with the extraction time taking only a fraction of a second.

Additionally, we evaluated five models from the `Llama` and `Mistral` families using two datasets *Essays* (Schuhmann, 2023) and *HC3* (Guo et al., 2023). The watermark's performance was tested across different watermark sizes and bit correction levels ($c$); all experiments used a consistent logit

Table 1: Comparison of methods on match rate(%) for different watermark lengths (in bits).

| Watermark size | 16 | 20 | 24 | 32 |
|---|---|---|---|---|
| (Fernandez et al., 2023) | 99.6 | 99.2 | 98.0 | NA |
| (Wang et al., 2024) | 98.8 | 98.4 | 97.2 | NA |
| (Yoo et al., 2024) | 73.6 | 49.2 | 30.4 | 8.4 |
| (Cohen et al., 2025) | 88.8 | 78.4 | 65.6 | 27.2 |
| (Qu et al., 2024) | 98.0 | 97.6 | 96.0 | 94.0 |
| Ours | 98.6 | 96.0 | 93.9 | 85.9 |

bias of $\delta = 6$. Each experiment was set up to generate 208 watermarked tokens using 100 prompts per dataset. All the generations with fewer than 208 tokens were filtered out.

The results of our experiments are shown in Table 2. For both datasets and models, applying bit correction (Appendix C) at the $c = 1$ level usually shows some improvement in watermark recovery. However, there exists a delicate balance between the number of meaningful watermark blocks and random ones that are introduced by bit correction. As this threshold is crossed, the new blocks become detrimental to the MCP algorithm, as can be seen in the results for level 2 bit correction. The extraction time (MCP computation) with correction $c = 1$ is still only a fraction of a second on a single PC (without any parallelization).

| $W$ size | Model | Bit Correction $c$ | | |
|---|---|---|---|---|
| | | 0 | 1 | 2 |
| 16 | Llama 2 7B | 98.3 | 94.2 | 76.9 |
| | Llama 3.2 3B | 98.7 | 95.7 | 88.2 |
| | Llama 3.2 3B Instruct | 89.8 | 88.0 | 56.8 |
| | Mistral 7B | 98.2 | 94.4 | 84.4 |
| | Mistral 7B Instruct | 91.2 | 96.2 | 54.8 |
| 32 | Llama 2 7B | 83.9 | 91.1 | 60.2 |
| | Llama 3.2 3B | 94.8 | 91.6 | 80.0 |
| | Llama 3.2 3B Instruct | 27.1 | 48.9 | 33.8 |
| | Mistral 7B | 90.2 | 95.2 | 65.7 |
| | Mistral 7B Instruct | 14.4 | 46.6 | 24.1 |

Table 2: Match rates obtained with our method. Prompts taken from Essays and HC3 datasets.

## 7 Adversarial Attacks

Once a watermarked text is released, an adversary may attempt to obscure or destroy the embedded watermark by modifying the generated text. In this section, we analyze the robustness of our scheme against three fundamental classes of adversarial interventions: *substitution*, *insertion*, and *deletion*.

### 7.1 Attack Model and Scope

We consider an adversary with full access to the generated text but without knowledge of the secret key used for watermark embedding. The adversary can perform a bounded number of token-level interventions, affecting at most a fixed fraction of the text (10% in our experiments). The goal of the adversary is to prevent reliable recovery of the embedded watermark.

### 7.2 Substitution Attacks

Substitution attacks replace existing tokens with alternative tokens. Crucially, substitutions preserve the overall token sequence length and therefore do not disrupt block alignment. As a result, each watermark block still corresponds to the intended group of tokens, although individual bits within a block may be corrupted.

This type of attack manifests primarily as *bit noise* in the recovered $(x, y)$ points rather than structural damage. Since MCP-based recovery is inherently tolerant to a moderate number of spurious points, the watermark can still be recovered as long as enough valid points remain collinear. Consequently, substitution attacks are the least destructive class of interventions for our scheme, and the watermark exhibits partial robustness even without additional protection.

### 7.3 Insertion and Deletion Attacks (Desynchronization)

Insertion and deletion attacks are significantly more damaging for our method because they alter the token stream length and thereby destroy block alignment. Even a single insertion or deletion can cause all subsequent watermark blocks to be misaligned, producing a cascade of invalid $(x, y)$ points during extraction. This effect is referred to as *desynchronization*.

While MCP recovery can tolerate random noise,

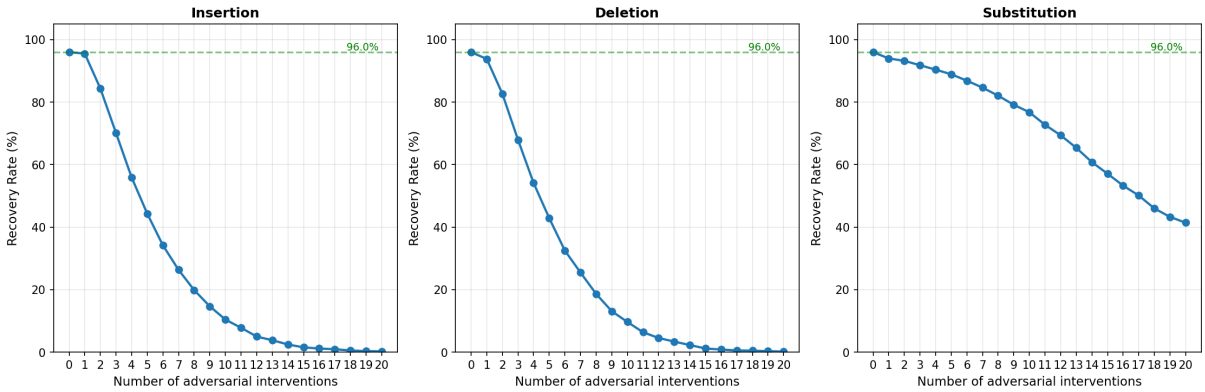


Figure 2: Effect of adversarial token-level interventions on watermark recovery. Recovery rates are shown for substitution, insertion, and deletion attacks with a total budget of 10% modified tokens and a text length of 200 tokens. The number of interventions controls the fragmentation of the attack (i.e., more interventions result in the attack being more evenly distributed throughout the text). Baseline performance without adversarial modifications is shown for reference.

large-scale desynchronization rapidly increases the number of spurious points and may prevent the true collinear structure from dominating. Therefore, the basic scheme is inherently fragile with respect to insertion and deletion attacks and offers limited robustness against them.

### 7.4 Experimental Evaluation

We evaluate robustness under adversarial interventions affecting 10% of the tokens, considering substitution, insertion, and deletion attacks. Figure 2 reports the watermark recovery rate as a function of the number of *adversarial interventions* applied to the text.

Across all attack types, recovery performance degrades gracefully as the number of interventions increases. As expected, substitution attacks exhibit the slowest degradation due to preserved synchronization whereas insertion and deletion attacks are more harmful. These results confirm that, while the proposed scheme is not fully robust against arbitrary editing, it can tolerate a non-trivial level of adversarial manipulation.

## 8 Conclusion

This work introduces a novel watermarking scheme for LLM-generated texts. The core idea is to represent the watermark as coefficients of a straight line $f(x)$ over a finite field, and embed a sequence of points $(x_i, f(x_i))$ lying on this line into the LLM-generated text. A watermark is recoverable even after partial editing or adversarial manipulation. We demonstrate that the scheme is lightweight, straightforward to implement, and well-suited for real-world applications.

The scheme is naturally extendable to support longer multi-bit watermarks. One straightforward approach is to embed multiple lines, each corresponding to a part of the watermark. We also propose the use of higher-degree polynomials instead of lines. In this approach, the watermark is encoded as the coefficients of a polynomial $f(x)$ and watermark points are sampled from this polynomial.

Overall, the proposed watermarking framework offers a flexible balance between efficiency and security, and opens the door to new watermarking paradigms based on algebraic structures. Future work may focus on optimising variants with higher-degree polynomials, enhancing resilience against editing attacks, and exploring hybrid schemes combining lines and polynomials to balance robustness and performance.

## Limitations

Our scheme assumes that enough $(x, f(x))$ points survive generation noise and light edits to dominate the MCP search; it is not designed to be robust against heavy paraphrasing, translation, aggressive summarization, or large insertion/deletion cascades that desynchronize blocks.

Theoretical bounds rely on independence assumptions and union bound arguments (and ignore vertical lines), which are informative but loose when $R$ grows relative to $q = 2^n$.

Our implementation and experiments instantiate only linear $f$ and a single-line (or conceptual multi-line) payload. Higher-degree variants and "top-$t$ lines" composition are presented as design options

rather than validated systems.

## References


D. H. Ballard. 1981. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122.

Jean-Paul Berrut and Lloyd N. Trefethen. 2004. Barycentric lagrange interpolation. *SIAM Review*, 46(3):501–517.

Aloni Cohen, Alexander Hoover, and Gabe Schoenbach. 2025. Watermarking language models for many adaptive users. In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2022. *Introduction to Algorithms*, 4-th edition. The MIT Press.

Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Jessica Fridrich, and Ton Kalker. 2007. *Digital Watermarking and Steganography*, 2nd edition. Morgan Kaufmann.

Xinyue Cui, Johnny Tian-Zheng Wei, Swabha Swayamdipta, and Robin Jia. 2025. Robust data watermarking in language models by injecting fictitious knowledge. ArXiv eprint 2503.04036.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, Nürnberg, Germany. IEEE.

Jiayi Fu, Xuandong Zhao, Ruihan Yang, Yuansen Zhang, Jiangjie Chen, and Yanghua Xiao. 2024. Gumbelsoft: Diversified language model watermarking via the gumbelmax-trick. ArXiv eprint 2402.12948.

Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. 2023. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arxiv:2301.07597*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. ArXiv eprint 2301.10226.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *Preprint*, arXiv:2303.13408. Dataset: `https://huggingface.co/datasets/jaroslawjanas/open-generation-data`.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip S. Yu. 2023a. A survey of text watermarking in the era of large language models. ArXiv eprint 2312.07913.

Yixin Liu, Hongsheng Hu, Xun Chen, Xuyun Zhang, and Lichao Sun. 2023b. Watermarking text data on large language models for dataset copyright. arXiv preprint arXiv:2302.13971.

Franco P. Preparata and Michael Ian Shamos. 1985. *Computational Geometry*. Springer New York.

Wenjie Qu, Wengrui Zheng, Tianyang Tao, Dong Yin, Yanze Jiang, Zhihua Tian, Wei Zou, Jinyuan Jia, and Jiaheng Zhang. 2024. Provably robust multi-bit watermarking for AI-generated text. *ArXiv, https://arxiv.org/pdf/2401.16820.pdf*.

Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. 2024. Watermarking makes language models radioactive. ArXiv eprint.

Tomás Sauer. 2006. Polynomial interpolation in several variables: Lattices, differences, and ideals. In Kurt Jetter, Martin D. Buhmann, Werner Haussmann, Robert Schaback, and Joachim Stöckler, editors, *Topics in Multivariate Approximation and Interpolation*, volume 12 of *Studies in Computational Mathematics*, pages 191–230. Elsevier.

Christoph Schuhmann. 2023. essays-with-intructions: Essays paired with instructions (hugging face dataset. Hugging Face, Dataset at ChristophSchuhmann/essays-with-instructions. Accessed Aug 4, 2025.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2024. Towards codable watermarking for injecting multi-bits information to llms. In *Proceedings of the Twelfth International Conference on Learning Representations (ICLR)*. OpenReview.

Ka Him Wong, Jicheng Zhou, Jiantao Zhou, and Yain-Whar Si. 2025. An end-to-end model for logits based large language models watermarking.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 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 (NAACL)*, pages 4031–4055. Association for Computational Linguistics.


## A Proof of Theorem 1

[Proof of Theorem 1] Let $q = 2^n$ and let $S = \{p_1, \ldots, p_R\}$ be $R$ points sampled i.i.d. uniformly from $(q) \times (q)$. We consider only non-vertical lines of the form $y = ax + b$ with $a, b \in (q)$.

**Counting non-vertical lines.** Each ordered pair $(a, b) \in (q) \times (q)$ determines a unique non-vertical line $\ell_{a,b} = \{(x, y) : y = ax+b\}$, and distinct pairs give distinct lines. Hence the set $\mathcal{L}$ of non-vertical lines has cardinality $|\mathcal{L}| = q^2$.

**Membership probability and the per-line distribution.** Fix any $\ell \in$. For a single random point $X$ uniformly distributed on $(q)^2$, we have

$$\Pr[X \in \ell] \;=\; \frac{|\ell|}{|(q)^2|} \;=\; \frac{q}{q^2} \;=\; \frac{1}{q}.$$

Because $p_1, \ldots, p_R$ are sampled i.i.d., the random variable

$$N_\ell \;\; |\ell \cap S|$$

is binomial: $N_\ell \sim \mathrm{Binomial}(R, 1/q)$. In particular,

$$\Pr\big[N_\ell = k\big] \;=\; \binom{R}{k}\left(\frac{1}{q}\right)^k\left(1-\frac{1}{q}\right)^{R-k},$$

which proves the "per-line" part of the theorem.

**Existence of a $k$-collinear subset on some non-vertical line (union bound).** For a line $\ell \in$ and a $k$-subset $T \subseteq [R]$ (indexing the points in $S$), let

$$E_{\ell,T} \;=\; \{\text{all points } \{p_i\}_{i\in T} \text{ lie on } \ell\}.$$

Then $\Pr[E_{\ell,T}] = (1/q)^k$, since the $p_i$ are independent and each falls on $\ell$ with probability $1/q$. The event that there exist at least $k$ collinear points on some non-vertical line is contained in

$$\bigcup_{\ell\in}\;\bigcup_{T\in\binom{[R]}{k}} E_{\ell,T}.$$

Applying the union bound,

$$\Pr\big[\exists\, k\text{-collinear points on a non-vertical line}\big] \;\le$$

$$\sum_{\ell\in}\;\sum_{T\in\binom{[R]}{k}} \Pr[E_{\ell,T}] \;=\; q^2\cdot\binom{R}{k}\cdot\left(\frac{1}{q}\right)^k \;=$$

$$\binom{R}{k}\, q^{-(k-2)}.$$

This is exactly the claimed bound.

## B Tradeoff between Block Size and Probability of Success

In watermarking that rely on identifying structures (such as collinear points) over finite fields like $GF(2^t)$, the designer is confronted with a tradeoff in choosing the block size $t$. This tradeoff arises when the underlying communication channel is modeled as a *Binary Symmetric Channel* (BSC) with bit-flip (crossover) probability $p$, representing a large language model (LLM) channel or a natural error-prone medium. On one hand, using a larger field (larger $t$) improves the geometric strength of the scheme—fewer collinear points are needed to identify a unique line, thus increasing robustness to adversarial actions. On the other hand, larger $t$ increases the likelihood that at least one bit in a block is corrupted, which degrades the reliability of receiving correct points.

To find a practical "sweet spot" between error resilience and field size, we examine the following result.

**Theorem 2** *Let $t$ be the size (in bits) of a block transmitted over a Binary Symmetric Channel (BSC) with bit-flip probability $p$, and let $P = (1-p)^t$ be the probability that a $t$-bit word is transmitted without error. Suppose $N$ such words are transmitted independently. Then, the probability that at least $n$ of them are received correctly is bounded below by:*

$$\mathbb{P}[\textit{at least } n \textit{ correct blocks}] = \sum_{k=n}^{N}\binom{N}{k}P^k(1-P)^{N-k} \ge L.$$

Each $t$-bit block is correctly received with probability $P = (1-p)^t$, and corrupted with probability $1-P$. Let $X$ be the number of correctly received blocks among $N$ independent transmissions. Then $X$ follows the binomial distribution $X \sim \mathrm{Binomial}(N, P)$. The probability that at least $n$ blocks are correct is:

$$\mathbb{P}[X \ge n] = \sum_{k=n}^{N}\binom{N}{k}P^k(1-P)^{N-k}.$$

This completes the proof.

This theorem allows designers to tune the parameters of their schemes. A larger $t$ means fewer collinear points are needed to identify a line in $GF(2^t) \times GF(2^t)$, but the likelihood of receiving a block without error drops exponentially in $t$.

Hence, the selection of $t$ depends on the acceptable communication overhead (total number of transmitted tokens), the crossover probability $p$ (determined by channel quality), and the minimum number of correct blocks $n$ required to successfully identify the embedded structure (e.g., a watermark or secret line).

**Example 2** *Suppose the channel has a bit-flip probability of $p = 0.2$ (i.e., each bit is transmitted correctly with probability $0.8$). The table below*

*shows the probability that an entire block of $t$ bits is received without error, i.e., $P = (1-p)^t$, for various values of $t$.*

| Block size $t$ | $P$ for $p = 0.1$ | $P$ for $p = 0.2$ |
|---|---|---|
| 2 | 0.81 | 0.64 |
| 3 | 0.729 | 0.512 |
| 4 | 0.6561 | 0.4096 |
| 5 | 0.59049 | 0.32768 |
| 6 | 0.53144 | 0.26214 |
| 7 | 0.47830 | 0.20972 |
| 8 | 0.43047 | 0.16777 |

Table 3: Probability of a $t$-bit block being received without error over BSC($p$)

*Now assume that $t = 6$ and $p = 0.2$, so the probability that a block is received error-free is $P = 0.26214$. We are interested in determining the minimum number of transmitted blocks $N$ such that the probability of receiving at least $n$ correct blocks is at least $L = 0.9$. The results are shown in Table 4.*

| $n$ (Correct) | Minimum $N$ | #Tokens ($N \cdot t$) |
|---|---|---|
| 4 | 15 | 90 |
| 8 | 34 | 204 |
| 12 | 50 | 300 |
| 16 | 66 | 396 |
| 32 | 132 | 792 |
| 64 | 258 | 1548 |

Table 4: Minimum number of transmitted blocks needed to achieve 90% confidence of receiving at least $n$ correct blocks with $t = 6$, $p = 0.2$

# C Bit Correction

When decoding a watermarked text, each $n$-bit block represents a $y$ coordinate in the Galois field $GF(2^n)$; however, the errors introduced during encoding can corrupt the blocks, making them random and not useful for the MCP algorithm. To combat this issue, a $c$-bit error correction algorithm is employed that generates all possible block variants within $c$ bit flips of the recovered block. Mathematically, given a decoded $n$-bit sequence $y$, this produces $\binom{n}{c} = \frac{n!}{c!(n-c)!}$ alternative sequences, each representing a potential correction for up to $c$ simultaneous bit errors. During verification, all new variant $y$ coordinates and their corresponding $x$ values contribute to the MCP solver. This increases both the chances of having a correct sequence in the bag of the coordinates and the required computational power to calculate the new variants and solve the MCP problem.

# D Watermarking with Multiple Lines and High Degree Polynomials

## D.1 Multiple Lines

Using a single line $f(x) = ax + b \in GF(2^n)$, it is possible to obtain the watermark $W = (a\|b) \in \{0,1\}^{2n}$. Further expansion of its length is possible by embedding $t$ distinct lines instead of just one. In this case, our MCP algorithm requires a minor adjustment to track the top $t$ straight lines that yield the highest point counts. This results in a collection of lines $\{f_i(x) = a_i x + b_i \mid i = 0, \ldots, t-1\}$ and their corresponding watermarks $\{W_i = (a_i\|b_i) \mid i = 0, \ldots, t-1\}$. To form a single $2tn$-bit watermark, we must determine an appropriate order for concatenation.

The method of concatenation depends on how the watermark is generated by the owner. The owner may either (1) generate individual pieces $W_i$ and concatenate them freely if the watermark is a random string created at the time of watermark embedding, or (2) follow a predetermined order if the watermark already exists and its structure is imposed by the application. In the case where the author is free to create a new watermark, a simple and deterministic method can be used:

- For each watermark component $W_i$, where $i = 0, \ldots, t-1$, compute its digest: $d_i = H(W_i)$, where $H()$ is a cryptographic hash function.
- Sort the digests $d_0, \ldots, d_{t-1}$ in increasing order and reorder the corresponding watermark components $W_i$ accordingly.
- Concatenate the reordered $W_i$ values into a single sequence.

If the watermark is fixed in advance, we can encode the order of components by indexing lines based on the least significant bits (LSB) of their $x$-coordinates. For simplicity, assume there are $t = 2^e$ lines, where $e \in \mathbb{N}$. Then:

- Any point whose $x$-coordinate has an $e$-bit LSB equal to $d$ belongs to the line $f_d(x)$.
- The lines are ordered by increasing values of $d$, i.e., $f_0(x), f_1(x), \ldots, f_{2^e-1}(x)$.

- The watermark $W = W_0 \| W_1 \| \ldots \| W_{2^e-1}$.

### D.2 High Degree Polynomials

An obvious generalisation is a choice of higher degree polynomials for our $f(x)$ instead of straight lines. This means that we deal with the polynomial over $GF(2^n)$ of the degree $t-1$ of the following form

$$f(x) = a_0 + a_1 x +, \ldots, a_{t-1} x^{t-1} \text{ where } t \geq 3$$

Points $\{(x_i, f(x_i) \mid i = 1, \ldots, N\}$ are translated into appropriate watermarking bits. Clearly, to determine the polynomial $f(x)$, it is enough to know $t$ points. We can apply the well-known Lagrange Interpolation to reconstruct $f(x)$. The watermark $W = (a_0 \parallel a_1 \parallel \ \ldots \ \parallel a_{t-1})$. The problem of recovery, however, becomes interesting when some points are corrupted by our adversary. The verifier in this case faces a task to identify a polynomial of degree $(t-1)$, which contains the maximum number of points. This is to say that we face the Maximum Co-Polynomial Points (MCPP) problem.

The MCPP problem is defined as follows:
*Instance*: A set $\mathbb{S}$ of $N$ points in the 2D plane, where each point $p_i = (x_i, y_i)$; $i = 1, 2, \ldots, N$.
*Question*: What is the largest subset $\mathbb{S}' \subset \mathbb{S}$ of points that lie on a single univariate polynomial $f(x)$ of degree at most $(t-1)$?

Some basic algorithms for solving the problem are: brute force polynomial interpolation (Berrut and Trefethen, 2004), Hough Transform for Polynomials (Ballard, 1981) or Gröbner Basis (Sauer, 2006).

Finding a polynomial $f(x)$ of degree $(t-1)$ from $t$ points becomes increasingly challenging as $t$ grows. However, our scenario is slightly different. If the watermarks remain unaltered by the adversary, any collection of $t$ such points will correctly determine the polynomial. In contrast, if the text has been modified, the points extracted from the tampered sections will not lie on the original polynomial $f(x)$. The verifier must distinguish between two types of points: good ones that lie on $f(x)$ and bad ones, which are scattered randomly in the plane $GF^2(2^n)$. The following theorem evaluates the probability of success when the verifier randomly selects $t$ points, hoping they all lie on $f(x)$.

**Theorem 3** *Given a collection $\mathbb{P}$ of all watermarks/points recovered from LLM-generated text, where $\mathbb{P} = \mathbb{F} \cup \mathbb{A}$, and $\mathbb{F}$ are points lying on $f(x)$ while $\mathbb{A}$ consists of random points introduced by the adversary. Suppose the verifier selects $t$ points at random, computes $\widetilde{f(x)}$, and checks whether other points are consistent with it. Then, the expected number of trials required to succeed (i.e., to find at least one set of $t$ points entirely from $\mathbb{F}$) is*

$$\left(\frac{\ell}{f}\right)^t,$$

*where $\ell = \#\mathbb{P}$ and $f = \#\mathbb{F}$.*

The probability of success in a single trial is $p = (f/\ell)^t$. If a trial fails, the next has probability $(1-p)p$ of being successful. More generally, the probability that the $k$-th trial is the first success is $(1-p)^{k-1}p$. This defines the geometric probability distribution $P(X = k) = (1-p)^{k-1}p$, whose expected value is $1/p$. □

**Example 3** *Let $\ell = 1000$, and let the number of good points be $f = 200$. If the watermark is defined by a polynomial of degree $t = 3$, then the expected number of trials is $(1000/200)^4 = 625$.*

## E False Positive Rate (FPR) from Theorem 1

Let

$$P = \{(x_i, y_i)\}_{i=1}^{N} \subset (2^n) \times (2^n)$$

be the set of extracted points used by the detector (one point per decoded block). The detector declares a watermark present iff

$$\mathrm{MCP}(P) \geq \tau,$$

where $\mathrm{MCP}(P)$ is the maximum number of points in $P$ lying on a single (non-vertical) line and $\tau$ is the detection threshold.

**Number of blocks and $n$.** If the verification segment contains $T$ tokens and each watermark block encodes $n$ bits, then the number of decoded blocks (hence extracted points) is typically

$$N \approx \left\lfloor \frac{T}{n} \right\rfloor. \tag{1}$$

**Definition of FPR.** Under the null hypothesis (no watermark), $P$ behaves like $N$ random points in $(2^n)^2$. The false positive rate is

$$\mathrm{FPR}(\tau; N, n) = \Pr[\mathrm{MCP}(P) \geq \tau \,|\, \text{no watermark}]\,. \tag{2}$$

**FPR bound via Theorem 1.** Let $q = 2^n$. By Theorem 1, for $\tau \geq 3$,

$$\begin{aligned} \mathrm{FPR}(\tau; N, n) &\leq \binom{N}{\tau} q^{-(\tau-2)} \\ &= \binom{N}{\tau} 2^{-n(\tau-2)}. \end{aligned} \tag{3}$$

**Choosing $\tau$ for a target FPR.** Given a desired upper bound $\alpha$ (e.g., $\alpha = 0.01$), choose the smallest integer $\tau \geq 3$ such that

$$\binom{N}{\tau} 2^{-n(\tau-2)} \leq \alpha. \tag{4}$$

Equivalently (taking $\log_2$),

$$\log_2 \binom{N}{\tau} - n(\tau - 2) \leq \log_2 \alpha. \tag{5}$$

### E.1 Worked example

Assume $n = 8$ (so $q = 256$), $N = 25$ blocks, and threshold $\tau = 5$. Then

$$\begin{aligned} \mathrm{FPR} &\leq \binom{25}{5} 256^{-3} = 53130 \cdot 256^{-3} \\ &= 53130 \cdot \frac{1}{16777216} \approx 3.17 \times 10^{-3}. \end{aligned} \tag{6}$$

Thus, the bound predicts an FPR below $0.4\%$ for these parameters.

It is worth noting that the bound in (3) is conservative (union bound style) and an empirical FPR is often smaller.

## F Quality of Watermarked Texts

Our watermarking scheme, like other existing approaches, modifies logit values by adding a small bias to a subset of logits (green tokens). We use the same logit bias as the state-of-the-art method proposed by (Qu et al., 2024), and their analysis can be directly applied to our approach. Specifically, they show that using a watermark with bias $\delta = 6$ only slightly degrades text quality, as measured by perplexity, and that the perplexity of LLM-generated text is similar with and without watermarking. In addition, they conducted a small qualitative study with human judges, concluding that any text quality degradation caused by the watermark is subtle according to human evaluation.

## G Computational complexity of MCP-based detection and practical optimizations

**Baseline complexity.** A standard approach to MCP in the plane over a finite field is the "anchor-and-count-slopes" method: for each anchor point $p_i$, compute slopes to all other points and count duplicates (points sharing the same slope with $p_i$ lie on the same line through $p_i$). Using hashing for slope counts, this runs in $\mathcal{O}(N^2)$ time and $\mathcal{O}(N)$ extra memory, which is already efficient for moderate $N$ (e.g., tens to a few hundreds of points), but could become expensive if $N$ is allowed to scale with very long texts.

**Optimization 1: fixed verification window (constant-time in text length).** A simple and effective mitigation is to cap the number of blocks inspected by running detection only on a fixed-length token window of size $T_0$ (e.g., $T_0 = 200$ tokens). This bounds the number of points to

$$N_0 \approx \left\lfloor \frac{T_0}{n} \right\rfloor,$$

and thus bounds MCP computation to $\mathcal{O}(N_0^2)$, which is constant with respect to total document length. The remainder of a long text can be treated as redundancy: it does not increase worst-case inference cost, while still being useful for robustness (see below).

**Optimization 2: multi-window / redundancy for robustness.** Using only a single fixed window may reduce detection robustness if that segment is heavily edited. A robust variant keeps inference efficient by evaluating MCP on $K$ non-overlapping (or randomly sampled) windows of length $T_0$ and taking the maximum score:

$$S = \max_{j=1,\ldots,K} \mathrm{MCP}(P_j),$$

where $P_j$ are points extracted from window $j$. This costs $\mathcal{O}(KN_0^2)$, still constant in total text length for fixed $K$ and $T_0$. In practice, small $K$ (e.g., $K \in \{2, 3, 5\}$) provides strong robustness while remaining fast.

**Optimization 3: early stopping with threshold $\tau$.** Detection only requires deciding whether $\mathrm{MCP}(P) \geq \tau$. The slope-counting procedure can stop early as soon as any anchor point yields a line with at least $\tau$ points. Conversely, if an anchor

point has fewer than $\tau - 1$ remaining points, it cannot create a $\tau$-point line and can be skipped. These checks often reduce runtime substantially in both positive and negative cases.

**Algorithm 4** Hashing-based MCP (Maximum Collinear Points)

**Input:** A set of $N$ points $\mathbb{S} = \{p_1, p_2, \ldots, p_N\}$ in $GF(2^n) \times GF(2^n)$
**Output:** Maximum number of collinear points

```
maxCount ← 1
for each point p_i ∈ S do
    Create an empty hash map H
    for each point p_j ∈ S such that p_j ≠ p_i do
        Compute slope s = (y_j − y_i)/(x_j − x_i)        ▷ Handle vertical lines separately
        Increment H[s] by 1
    end for
    maxCount ← max(maxCount, max_{s∈H}(H[s] + 1))
end for
return maxCount =0
```