LLM-Text Watermarking based on Lagrange Interpolation

Jarosław Janas[1], Paweł Morawiecki[1], and Josef Pieprzyk[1,2]

[1] Institute of Computer Science, Polish Academy of Sciences, Warsaw, Poland
[2] Data61, CSIRO, Sydney, Australia

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 – either to the specific language model that produced it or to the individual user who embedded their identity via a so-called multi-bit watermark. 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 author identity even when the text has been heavily 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. The x-coordinates are computed pseudorandomly using a cryptographic hash function H applied to the concatenation of the previous token's identity and a secret key s_k. Crucially, the x-coordinates do not need to be embedded into the text – only the corresponding $f(x)$ values are embedded. 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 highly effective, allowing the recovery of the author's identity even when as few as three genuine points remain after adversarial manipulation.

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 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.

Notable examples of LLMs include GPT-4 by OpenAI [16], which demonstrates strong reasoning and multitask capabilities; PaLM by Google [5], which emphasizes multilingual and multitask learning; and LLaMA by Meta [21], which explores high-performance language modeling with relatively modest computational resources. These models not only advance the state of natural language processing but also raise new challenges and opportunities for applications in domains such as content generation, code synthesis, and interactive AI systems.

The growing adoption of Large Language Models (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 digital artifact originates from its rightful source [8]. 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 generated text. 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. These approaches can be organised into four broad 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 [19], the authors introduced the notion of text radioactivity. It allows to detect whether an LLM has memorised radioactive (marked) data points. TextMarker described in [14] uses backdoor-based membership inference to embed private ownership information. The scheme [9] 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 [12] splits the collection of tokens into green and red subsets. The green tokens encode watermarks while red ones mark fake ones (zero-bit watermarking). Wong et. al. [22] describes their logits-to-text watermarking, which jointly optimises encoder-decoder architecture for high-fidelity and high-detectability watermarks. The GumbelSoft watermarking described in [10] improves sampling diversity while maintaining robust detection by refining Gumbel-based logits control.

Watermarking During Token Sampling – this approach injects watermarks during the token selection process, often by biasing token choices at generation time. In their work [12], Kirchenbauer et al. applies a stochastic green token sampling, which softly biases generation toward watermark-encoding tokens at each step. The work [23] proposes frequency-based watermarking that embeds periodic token patterns detectable via Fourier analysis. In their work [25], Zhu et al. discuss their dual-pattern embedding, which simultaneously encodes watermarks in both the logits and sampling distributions.

Post-Generation Detection and Attribution – the following works focus on detecting watermark patterns or attributing text to specific models or users after generation. Niess et al. [15] proposes a method for generating watermarks that alters token probabilities using stylometry. More precisely, it uses acrostica and sensorimotor norms to LLMs. Zhao et al. [24] introduce their Unigram-Watermark, which a theoretical framework for watermarking generation and correctness of watermarking detection.

For a comprehensive survey of LLM watermarking techniques, readers are referred to the review by Liu et al. [13].

2.1 Contributions

The paper makes the following contributions:

- We design a watermarking scheme that embeds points from a straight line $f(x)$ into LLM-generated text. The x-coordinates are computed using a cryptographic hash function $H()$ applied to the identity of the preceding token and a secret key s_k. Notably, there is no need to embed these coordinates explicitly, as the verifier can reconstruct them directly from the text. Only the corresponding $f(x)$ values are embedded. To extract the watermark, the verifier must solve an instance of the Maximum Collinear Points (MCP) problem. We show that several algorithms exist for this task, with the most efficient ones achieving quadratic complexity, i.e., $\mathcal{O}(N^2)$, where N is the number of candidate points.
- Our probabilistic analysis demonstrates that any adversarial modification of the points $(x, f(x))$ is likely to be detected, as the altered points are statistically scattered randomly in the 2D plane. This detection probability increases with the dimension n of the field $GF(2^n)$.
- Experimental results highlight the importance of choosing the field size carefully. For small fields such as $GF(2^6)$, natural structural interactions between points can invalidate the probabilistic analysis. However, for larger fields such as $GF(2^{32})$ and beyond, the analysis holds reliably.
- The proposed scheme can be naturally extended to support long multi-bit identities. We present two options: (1) using multiple lines, each encoding a distinct part of the identity, and (2) using higher-degree polynomials instead of lines. In the latter case, reconstructing the embedded polynomial corresponds to solving the Maximum Co-Polynomial Points problem. However, existing algorithms for this task have complexities that grow with the polynomial's degree.

Table 2 compares some multi-bit watermarking schemes with the one presented in this work. The scheme by Qu et al. [18] proposes a robust multi-bit watermarking that encodes user identity by mapping segments of the message to selected token positions using pseudo-random assignment, reinforced with Reed–Solomon error correction. The paper [11] by Haoyu Jiang et al. proposes a multi-bit watermarking framework designed to credibly identify texts generated by LLMs. The core idea involves a trusted third party (TTP) coordinating with multiple LLM vendors to embed and later extract identity-carrying watermarks.

Table 1. Some LLM-generated text multi-bit watermarking schemes, where b is the number of bits in user identity, k is the number of segments, T is the number of tokens in the text and N is the number of points

Authors	Tools	Extraction Algorithm	Complexity
Qu et al [18]	RS Codes, Hash	RS decoding	$\mathcal{O}(2^b)$ or $\mathcal{O}(k2^{b/k})$
Jiang et al [11]	TTP (digital signatures), Hash	Deterministic mapping	$\mathcal{O}(T + b)$
This work	Lagrange Interpolation, Hash	MCP	$\mathcal{O}(N^2)$

The rest of the paper is structured as follows. Section 3 focuses on the quality and security properties required from a watermarking scheme. Section 4 describes the tools and building blocks used in the design of our watermarking schemes. Section 5 introduces our basic watermarking scheme along with

algorithms for recovering watermarking bits. Section 6 presents probabilistic analysis evaluating the likelihood that an adversary can implant a fake point that would be accepted as genuine. Section 7 details our experimental results. Section 8 demonstrates how our watermarking scheme can be extended to support long multi-bit identities. Finally, Section 9 concludes the paper.

3 Quality and Security Requirements for LLM-text Watermarking

The main requirements for LLM-text watermarking ensure that the watermark is effective (i.e. easy to embed and extract) while preserving text quality. These include: imperceptibility, textual coherence and readability, scalability and efficiency, easy and reliable detection, and adaptability to different LLMs. Typical adversarial models for LLM-text watermarking assume that the owner of the text (either the LLM itself or its agent) embeds a watermark into the generated content. Once the watermarked text is published, an adversary with access to the text may attempt the following attacks:

- Recovery of the watermark from the text to analyze or reverse-engineer the embedding method.
- Removal of the watermark to evade detection and attribution.
- Impersonation, where the adversary embeds a watermark to mimic a different AI model or author.
- Modification of the watermark through insertion, repetition, substitution, or deletion, with the goal of making the changes undetectable while disrupting verification.
- Collusion attack, where the adversary, having access to multiple watermarked text samples, combines them into a single text that passes watermark verification.
- Extraction of secret keys by applying cryptanalysis techniques to break the watermarking scheme.

The required security features for LLM-text watermarking are enumerated below:

- Resistance against adversarial editing – the watermark should remain detectable even if the text undergoes modifications such as paraphrasing, truncation, or reformatting.
- Resilience to modifications, substitutions, and removal of watermarked text.
- Authentication – the watermark should provide strong proof that the text was generated by a specific source (either by LLM or author who holds a secret key).
- Low false positive (FP) and false negative (FN) rates – the detection method should minimize errors, ensuring that non-watermarked text is not falsely flagged and that genuine watermarks are not missed.
- Security against collusion – the watermark should remain secure even if multiple watermarked texts are combined or analysed to disclose the secret elements (keys).

In this work, we focus on a specific security goal: enabling the owner of an LLM-generated text to prove that the text contains a substantial portion of their original content, from which a multi-bit identity K can be reliably recovered. This identity serves as a cryptographic witness of authorship or ownership. Our approach is grounded in the use of Lagrange interpolation over points in a two-dimensional plane. Each point encodes a fragment of the identity. To ensure unpredictability and prevent pattern leakage, we employ a cryptographic hash $H()$ to determine the x-coordinates of the interpolation points.

4 Components for Watermark Designs

4.1 Lagrange Interpolation over $GF(2^n)$

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 δ_{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.

4.2 Embedding Bits into Tokens

Large language models (LLMs) employ token vocabularies of varying sizes, with recent models like GPT-3 and Llama 2 using over 30,000 tokens, while Llama 3 supports up to 128,000. The trend toward larger vocabularies improves tokenisation and model performance but comes at the cost of increased computational demands. To embed watermarks in generated text, we build upon the method of partitioning the vocabulary into two disjoint subsets $\mathcal{V}_0$ and $\mathcal{V}_1$ representing binary marks 0 and 1, respectively. This concept extends the idea of "Green" and "Red" tokens introduced by Kirchenbauer et al. [12], which designate allowable and disallowed tokens.

Token partitioning can be done efficiently using pseudorandom bit generators seeded with a secret key or derived using cryptographic hashes. However, regenerating subsets frequently can become inefficient. Instead, a token's membership in $\mathcal{V}_0$ or $\mathcal{V}_1$ can be computed on the fly using methods such as linear equations over token ID bits, encryption (e.g., AES in ECB mode), or cryptographic hashes (e.g., SHA3). These approaches aim to maintain balanced partitions and are especially effective in high-entropy text, where the natural variety of token usage allows for seamless watermarking without degrading the quality or fluency of the output.

Consider an example where we apply the SHA-3 cryptographic hash function to token IDs, optionally combined with a secret key s_k. In this case, we compute $h = (h_1, \ldots, h_n) = \text{SHA3}(s_k, ID_v)$. The watermark assigned to the token v is then defined as $m = \bigoplus_{i=1}^{n} h_i$. A well-designed hash function ensures that the subsets $\mathcal{V}_i; i \in \{0, 1\}$, are balanced with high probability, providing a fast and efficient solution.

5 Watermark based on Lagrange Interpolation

In many real-world scenarios, the owner of an LLM-generated text may wish to assert authorship by demonstrating that the embedded watermark allows recovery of a multi-bit identity K. For this to be effective, the extraction algorithm must be able to retrieve the identity even if the text has been altered or partially redacted by an adversary. To support robust and verifiable authorship attribution, any sufficiently long fragment of the watermarked text should carry enough information to reconstruct K, serving as undeniable proof of origin.

To achieve this, we propose using Lagrange interpolation to recover a secret polynomial $f(x)$ whose coefficients encode the identity K. Each watermarking point is of the form $(x, f(x))$, where the x-coordinates are generated pseudorandomly using a cryptographic hashing. This design ensures that even if only a fraction of these points survive tampering, they suffice to recover K. 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.

As previously mentioned, we focus on the simplest nontrivial case where $f(x)$ is a degree-1 polynomial, i.e., a straight line of the form $f(x) = a_0 + a_1 x \in GF(2^n)[x]$. The secret identity is encoded as $K = (a_0 \| a_1) \in GF^2(2^n)$. A straightforward strategy would embed N watermark points directly onto this line, which requires a text of at least $2nN$ tokens. To improve efficiency, we propose a dense embedding scheme where each value $y_i = f(x_i)$ is embedded using n tokens, and the x-coordinate x_i is derived from the identity of the preceding token. Formally, given a sequence of tokens $S_{i-1}, S_i, \ldots, S_{i+n}$, we define:

$$x_i = H(ID_{S_{i-1}}, s_k),$$

and use the subsequent block $(S_i, \ldots, S_{i+n})$ to encode $f(x_i)$. This construction requires only $nN + 1$ tokens to embed N points, reducing overhead compared to the naive $2nN$-token requirement. The first token S_0 is not watermarked; it serves solely to generate the initial x-coordinate.

Algorithm 1 outlines the watermark embedding process based on Lagrange interpolation. Our approach diverges from common designs where the token vocabulary is repartitioned for each embedding step. While dynamically splitting $\mathcal{V}_0$ and $\mathcal{V}_1$ per token can enhance security, it introduces unnecessary computational overhead, particularly on the verifier's side.

To embed a watermark bit $i \in \{0, 1\}$, the LLM is queried to produce token logits. The probabilities of top-ranked tokens are then adjusted so that tokens from the appropriate subset $\mathcal{V}_i$ are favored subject to a parity constraint. This constraint can be defined either as the XOR of token ID bits or as the parity of the output of a cryptographic hash function $H(ID_{\text{token}}, s_k)$ (see Section 4.2). In this construction, the sets $\mathcal{V}_0$ and $\mathcal{V}_1$ are kept static across the embedding process, ensuring efficient and consistent watermark detection. The algorithm processes text block by block. For each block (lines 6,7), two binary sequences are prepared. The first is a block generated by the hash, which consists of an x-coordinate. The second block is $y_i = f(x_i)$. The loop from line 8 to 15 injects bits y_i into the LLM text. Algorithm 2 outlines the process of watermark extraction. The procedure begins by initializing the list $\mathbb{P}$ of candidate points. Two token subsets, $\mathcal{V}_0$ and $\mathcal{V}_1$, are then constructed. Once these are prepared, the watermark sequence $\mathbb{W} = (w_1, \ldots, w_{nN})$ is extracted from the modified text. The watermark bits and their corresponding tokens are processed sequentially. For each bit, the algorithm computes the coordinate $x_i = H(ID_{\tilde{S}_{i-1}}, s_k)$ using the identity of the preceding token. It then assigns the value y_i as the sequence $(w_{i+1}, \ldots, w_{i+1+n})$, and the resulting point (x_i, y_i) is appended to the list $\mathbb{P}$.

As a result, the extraction algorithm builds a list of N genuine points that lie on the original line $f(x)$, along with $n(N-1) + 1$ additional points that are scattered across the 2D plane. An adversary may both insert fake points and remove genuine ones, attempting to obscure the original structure. Nonetheless, as long as a sufficient number of authentic points remain on the correct straight line, the identity K can still be reliably recovered – assuming the adversary does not manage to create a competing line with a comparable number of points. The key computational challenge becomes identifying the straight line that contains the largest subset of points from the entire set – a classic geometric problem known as the Maximum Collinear Points (MCP) problem.

Algorithm 1: Watermark Embedding Secret Identity K

1: **Input:** Autoregressive language model **LLM** used to generate a text of length $T = nN + 1$, prompt sequence PR, token set $\mathcal{V}$, n-bit secret key s_k, cryptographic hash $H()$ and straight line $f(x) = a_0 + a_1 x$ with the secret identity $K = (a_0 \| a_1)$, where computations are done in $GF(2^n)$.

2: **Output:** Public LLM text $\mathcal{T} = S_{0,n}, S_{1,1}, \ldots, S_{N,n}$.

3: Using secret key s_k, create two lists $\mathcal{V}_0, \mathcal{V}_1$, where $\mathcal{V}_i$ are tokens carrying watermark $i \in \{0,1\}$ ($\mathcal{V}_0 \cup \mathcal{V}_1 = \mathcal{V}$)

4: Initialize $\mathcal{T} = S_{0,n} \leftarrow LLM(PR)$

5: **for** $i = 1$ to N **do**

6:      $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}) \leftarrow H(ID_{S_{i-1,n}}, s_k)$ ▷ *x- coordinate*

7:      $y_i = f(x_i) = (y_{i,1}, y_{i,2}, \ldots, y_{i,n})$ ▷ *value $y = f(x)$*

8:      **for** $j = 1$ to n **do**

9:          Compute logits: $v = \text{LLM}(PR, S_{0,n}, S_{1,1}, \ldots, S_{i,j-1})$

10:          Bias the logits:

11:          **if** $y_{i,j} = 0$ **then** add bias to $\hat{v}[w] = v + \delta$ for all $w \in \mathcal{V}_0$

12:          **else** add bias to $\hat{v}[w] = v + \delta$ for all $w \in \mathcal{V}_1$

13:          **end if**

14:          Append $S_{i,j}$ to $\mathcal{T}$ by sampling from biased logits $\hat{v}$

15:      **end for**

16: **end for**

17: **return** $\mathcal{T}$

Algorithm 2: Watermark Extraction

1: **Input:** LLM text $\widetilde{\mathcal{T}} = (\tilde{S}_0, \tilde{S}_1, \ldots, \tilde{S}_{nN})$, token set $\mathcal{V}$, n-bit secret key s_k, n-bit cryptographic hash $H()$, which generates x-coordinates of unknown $f(x)$, where points $(x_i, f(x_i))$ are embedded into the text according to Algorithm 1; $i = 1, \ldots, N$, computations are done in $GF(2^n)$.

2: **Output:** The secret user identity $K = (a_0 \| a_1)$

3: Create an empty list of points $\mathbb{P} = \varnothing$ of $(x, f(x))$

4: Using secret key s_k, create two lists $\mathcal{V}_0, \mathcal{V}_1$, where $\mathcal{V}_i$ are tokens carrying watermark $i \in \{0,1\}$ ($\mathcal{V}_0 \cup \mathcal{V}_1 = \mathcal{V}$)

5: Extract watermark $\mathbb{W} = (w_1, \ldots, w_{nN})$ using $\mathcal{V}_0$ and $\mathcal{V}_1$

6: **for** $i = 0$ to $n(N-1)$ **do**

7:      $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n}) \leftarrow H(ID_{\tilde{S}_i}, s_k)$

8:      $y_i = (w_{i+1}, \ldots, w_{i+1+n})$

9:      $\mathbb{P} \leftarrow \mathbb{P} \cup (x_i, y_i)$

10: **end for**

11: Solve the maximum collinear points instance for $\mathbb{P}$ and find the line $f(x) = a_0 + a_1 x$, which contains them

12: **return** $\tilde{K} = (a_0 \| a_1)$

5.1 Maximum Collinear Points Problem

The adversarial model is illustrated in Figure 1. In this model, the adversary has full control over the text and may apply arbitrary modifications, including insertions and deletions. However, extensive editing is often counterproductive, as it effectively amounts to rewriting the entire text from scratch. In practice, it is reasonable to assume that the adversary will aim to preserve as much of the original content as possible to minimise effort – after all, the motivation behind plagiarism is typically to save time by reusing someone else's work. Interestingly, if the original watermark is available, the identity K can be recovered directly using Lagrange interpolation. However, a verifier who only knows the key s_k faces a more challenging task, as the extracted watermark may have been tampered with by an adversary. Effectively, the verifier deals with an instance of the MCP problem.

Fig. 1. Adversarial model

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 straight 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 [6]).
- Sorting-based approach — the algorithm takes each point as a reference and computes slopes between the reference point and the other points. The slopes are sorted to group collinear points. Finally, it counts the maximum number of collinear points for each reference point. Its time and space complexities are $\mathcal{O}(N^2 \log N)$ and $\mathcal{O}(N)$, respectively (see [2]).
- 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 [6]).
- Convex-hull approach — the algorithm uses a convex hull (such as Graham's scan or Andrew's monotone chain) to preprocess points and then adds points, checking them for collinearity. Its time and space complexities are $\mathcal{O}(N^2)$ and $\mathcal{O}(N)$, respectively (see [17]).

The algorithms presented above show that the MCP problem can be efficiently solved for the number of points $N < 10^5$. Table 2 compares the algorithms. Let us consider the Hashing-based algorithm (see Algorithm 3). 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

Algorithm	Time Complexity	Space Complexity	Approach
Brute force [6]	$\mathcal{O}(N^3)$	$\mathcal{O}(1)$	Check triplets
Sorting-based [2]	$\mathcal{O}(N^2 \log N)$	$\mathcal{O}(N)$	Sort slopes
Hashing-based [6]	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	Hashing for slopes
Convex hull [17]	$\mathcal{O}(N^2)$	$\mathcal{O}(N)$	Geometric hull

Table 2. Comparison of algorithms for the Maximum Collinear Points Problem

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 parts of secret sharing.

Algorithm 3: Hashing-based MCP (Maximum Collinear Points)

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

3: maxCount $\leftarrow 1$
4: **for** each point $p_i \in \mathbb{S}$ **do**
5:     Create an empty hash map H
6:     **for** each point $p_j \in \mathbb{S}$ such that $p_j \neq p_i$ **do**
7:         Compute slope $s = \frac{y_j - y_i}{x_j - x_i}$ ▷ *Handle vertical lines separately*
8:         Increment $H[s]$ by 1
9:     **end for**
10:     maxCount $\leftarrow \max(\text{maxCount}, \max_{s \in H}(H[s] + 1))$
11: **end for**
12: **return** maxCount

---

6 Security Evaluation

According to our security model (see Figure 1), the adversary has full control over the LLM-generated text. They may freely modify, delete, edit, replace, or duplicate any part of it. In this setting, unconditional security is unattainable, as the adversary can always rewrite the content from scratch using the original as a guide. A more realistic goal is to ensure that an adversary – seeking to save time by reusing parts of the original text – is likely to leave behind fragments of the watermark that, with high probability, still allow recovery of the original author's identity. Therefore, the primary security objective is to preserve authorship attribution. Indirectly, this means (1) maximizing the probability of recovering the embedded identity, and (2) increasing the effort required to fully erase authorship to a level comparable with rewriting the entire text from scratch.

Let us consider the extent to which an adversary must tamper with the watermarked text to successfully destroy the watermark. In other words, we aim to determine the threshold beyond which the original owner can no longer prove authorship or extract their identity K from the LLM-generated text. During extraction, we obtain a total of $N = F + R$, 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 K 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 R of random points 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. *Given a plane $GF(2^n) \times GF(2^n)$, R random points and the assumption that $R \ll 2^n$, then the probability that there is a straight line that contains k points is approximated by*

$$\binom{R}{k} \left(\frac{1}{2^n}\right)^{k-2} \left(1 - \frac{1}{2^n}\right)^{R-k},$$

where $k > 2$.

Proof. We start tossing R points. The first two tosses give a unique straight line. The event that the tosses produce the same point occurs with negligible probability so we ignore it. Now, we expect that among the $(R-2)$ other tosses, $(k-2)$ belong to the line (with probability $1/2^n$) and $(R-k)$ fall outside the line (with probability $(1 - 1/2^n)$). The binomial coefficient provides the number of possible choices of k out of R possibilities. $\square$

Theorem 1 provides us with a tool to evaluate the probability that Algorithm 2 extracts the identity K correctly. The algorithm fails to give the true K if there is a subset of F or more random points (whose total number is R) sitting on the same straight line. The probability of this event is

$$P(\text{fail}) = \sum_{k=F}^{R} \binom{R}{k} \left(\frac{1}{2^n}\right)^{k-2} \left(1 - \frac{1}{2^n}\right)^{R-k}$$

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

$$\binom{16}{4} \left(\frac{1}{64}\right)^2 \left(\frac{63}{64}\right)^{12} \approx 0.37$$

It is easy to check that probabilities that five and more points lie on a straight line vanish exponentially. Consequently, the probability of correct recovery of K is ≈ 0.62. The probability of the correct K recovery improves significantly if $F = 5$ and $R = 15$ as it increases to ≈ 0.99.

7 Experiments

We conducted experiments to analyze the maximum number of collinear points that can be generated by an adversary who randomly selects points on a 2D plane. The adversary's goal is to deceive the verifier by creating a dense enough cluster of points that form a fake straight line, which the verifier might mistake for a genuine watermark. These experiments were carried out using the PARI software package - see Table 3. The results confirm our intuition: in smaller fields, the number of collinear points tends to grow, but not significantly. In contrast, for larger fields such as $GF(2^{32})$ and beyond, the maximum number of collinear points observed is typically just two – indicating that the points

Random Points #	$GF(2^6)$	$GF(2^8)$	$GF(2^{12})$	$GF(2^{16})$	$GF(2^{32})$
8	3	2	2	2	2
16	3	3	2	2	2
32	5	3	2	2	2
64	6	4	3	2	2
128	6	5	3	2	2
256	10	5	3	2	2
512	14	8	3	2	2
1024	20	11	4	2	2

Table 3. Number of collinear points detected in random sets of varying sizes across different Galois fields

are largely scattered across the plane, forming only trivial two-point lines. Under these conditions, the presence of three or more collinear watermarking points is a strong indicator of intentional embedding. Moreover, the sender can embed a larger number of such points to make the watermark more resilient against adversarial interference.

Given a short paragraph consisting of 100 tokens. We can embed a 100-bit watermark. In practice, a key challenge lies in selecting the appropriate dimension of the Galois field used for the computation. If we choose $GF(2^6)$, each point occupies 6 bits. This means the text can accommodate 16 watermark points. According to Table 4, only 8 of these points need to survive for the correct watermark line to be recovered, while the remaining 75 points may be replaced by fake or random ones without affecting recoverability. If we instead use $GF(2^8)$, the total number of watermark points decreases to 12. In this case, as shown in Table 4, just 6 genuine points are enough to reconstruct the correct line, while the remaining 77 random points form lines with a maximum of only 5 points. With $GF(2^{12})$, the number of watermark points drops further to 8. Still, only 4 genuine watermark points are needed to reconstruct the original line, as 77 random points with a maximum of only 3 points. A broader discussion considering different text lengths – such as a short paragraph, a paragraph, and a 1-page report – is provided in Table 4. It is evident that for longer documents, using Galois fields of higher dimension is advantageous, as the appearance of a three-point line becomes strong evidence of the watermark's presence and authenticity.

Document	# of Tokens (bits)	Arithmetics GF	Total # of Points $n(N-1)+1$	# of Watermark Points N	# of Fake Points	# of Points Needed to Recover K
Short		$GF(2^6)$	91	16	75	8
Paragraph	100	$GF(2^8)$	89	12	77	6
		$GF(2^{12})$	85	8	77	4
		$GF(2^6)$	295	50	245	12
Paragraph	300	$GF(2^8)$	289	37	252	6
		$GF(2^{12})$	289	25	264	4
		$GF(2^6)$	691	116	575	18
Report	700	$GF(2^8)$	689	87	602	10
(1 page)		$GF(2^{12})$	685	58	627	4
		$GF(2^{16})$	673	43	630	3

Table 4. Watermarking points for different documents and arithmetics

8 Long Multi-bit Watermarking

8.1 Multiple Lines

Using a single line $f(x) = ax + b \in GF(2^n)$, it is possible to obtain the identity $K = (a\|b) \in \{0,1\}^{2n}$. 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 $\{f_i(x) = a_i x + b_i \mid i = 1, \ldots, t\}$ and their corresponding identities $\{K_i = (a_i\|b_i) \mid i = 1, \ldots, t\}$. To form a single $2tn$-bit identity, we must determine an appropriate order for concatenation. A simple method proceeds as follows:

- For each identity K_i, where $i = 1, \ldots, t$, compute its digest: $d_i = H(K_i)$, where $H()$ is a cryptographic hashing.
- Sort the digests $\{d_1, \ldots, d_t\}$ in increasing order and reorder the corresponding identitys K_i accordingly.
- Concatenate the reordered identitys into a single sequence.

8.2 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-know Lagrange Interpolation to reconstruct $f(x)$. The identity $K = (a_0 \parallel a_1 \parallel \ldots \parallel a_{t-1}\}$. 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 given below.

- Brute force polynomial interpolation — the algorithm iterates over all $\binom{N}{t}$ subsets of t points, interpolates a $(t - 1)$-degree polynomial and counts how many other points lie on the curve $f(x)$. Its complexity is $\mathcal{O}(N^{t+1})$, which quickly becomes impractical for large N and t [4].
- Random Sampling + Verification – the algorithm selects many subsets of t points, interpolates a polynomial for the given subset and verifies how many points lie on it. Its complexity is $\mathcal{O}(T(t^2+N)$, where T is the number of trials.
- Hough Transform for Polynomials – the algorithm uses the classic Hough Transform (used in computer vision for line detection) to parametrise t-degree polynomials. Each point votes for all curves passing through it – curves with most votes are detected. Its complexity $\mathcal{O}(Nq^{t-1})$, where q is the space size parameter [1].
- Gröbner Basis – the algorithm encodes the point constraints as a system of polynomial equations and use Gröbner bases to simplify and solve. Its complexity is exponential in t although often better in practice [20].

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 2. *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 $f = \#\mathbb{F}$.

Proof. 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 $1/p$. $\qquad\square$

Example 2. 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 $(1000/200)^4 = 625$.

9   Conclusions

This work introduces a novel watermarking scheme for LLM-generated text that enables the recovery of a identity author identity through geometric encoding. The core idea is to represent the identity as 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 text. These points serve as the watermark, 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 multiple identities. One straightforward approach is to embed multiple lines, each corresponding to an individual identity. While effective, this method increases the probability that an adversary may generate a fake point that aligns with one of the embedded lines, potentially resulting in false positives. Moreover, if the goal is to produce a single unified identity, a method for determining the correct order of concatenation among the individual identities must be established. To address this, we propose the use of higher-degree polynomials instead of lines. In this approach, the identity is encoded as the coefficients of a polynomial $f(x)$ and watermark points are sampled from this polynomial. This not only reduces the adversary's chances of fabricating a point that lies on the curve—since a random point lies on a polynomial of the degree t with probability only $1/2^n$ – but also enables straightforward concatenation of the coefficients into a single, longer identity. The trade-off, however, is an increase in computational complexity: reconstructing a polynomial of the degree t requires solving a more resource-intensive interpolation problem, especially as t or the field size grows.

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 multi-identity recovery, enhancing resilience against editing attacks, and exploring hybrid schemes combining lines and polynomials to balance robustness and performance.

ACKNOWLEDGMENT

References

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

[2] Bentley and Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, C-28(9):643–647, 1979.

[3] Daniel J. Bernstein, Ted Krovetz, and Peter Schwabe. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. Online, https://competitions.cr.yp.to/caesar.html, 2019.

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

[5] Aakanksha Chowdhery, Anish Vaswani, Steven J. Rennie, Mikhail Pavlov, Jacob Devlin, Sanjay Aggarwal, Mike Lewis, Neil Houlsby, Colin Raffel, Barbara Plank, Lee Howard, Martin D. Riley, Michael Swietojanski, Mo Yu, Dipanjan Das, Mike Schuster, Yiming Yang, Jakob Uszkoreit, and Yonghui Wu. PaLM: Scaling language modeling with pathways. In *Proceedings of the 39th International Conference on Machine Learning (ICML 2022)*, 2022.

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

[7] Nicolas T. Courtois and Willi Meier. Algebraic attacks on stream ciphers with linear feedback. In *Advances in Cryptology — EUROCRYPT 2003*, pages 345–359. Springer Berlin Heidelberg, 2003.

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

[9] Xinyue Cui, Johnny Tian-Zheng Wei, Swabha Swayamdipta, and Robin Jia. Robust data watermarking in language models by injecting fictitious knowledge. ArXiv eprint 2503.04036, 03 2025.

[10] Jiayi Fu, Xuandong Zhao, Ruihan Yang, Yuansen Zhang, Jiangjie Chen, and Yanghua Xiao. Gumbelsoft: Diversified language model watermarking via the gumbelmax-trick. ArXiv eprint 2402.12948, 02 2024.

[11] Haoyu Jiang, Xuhong Wang, Ping Yi, Shanzhe Lei, and Yilun Lin. Credid: Credible multi-bit watermark for large language models identification. 12 2024.

[12] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. ArXiv eprint 2301.10226, 01 2023.

[13] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip S. Yu. A survey of text watermarking in the era of large language models. ArXiv eprint 2312.07913, 12 2023.

[14] Yixin Liu, Hongsheng Hu, Xun Chen, Xuyun Zhang, and Lichao Sun. Watermarking text data on large language models for dataset copyright. arXiv preprint arXiv:2302.13971, 05 2023.

[15] Georg Niess and Roman Kern. Stylometric watermarks for large language models. 05 2024.

[16] OpenAI. GPT-4 Technical Report. `https://openai.com/research/gpt-4`, 2023.

[17] Franco P. Preparata and Michael Ian Shamos. *Computational Geometry*. Springer New York, 1985.

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

[19] Tom Sander, Pierre Fernandez, Alain Durmus, Matthijs Douze, and Teddy Furon. Watermarking makes language models radioactive. ArXiv eprint, 02 2024.

[20] Tomas Sauer. 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, 2006.

[21] Hugo Touvron, Thibault Louvrier, Matthieu Cord, Piotr Bojanowski, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

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

[23] Zhenyu Xu, Kun Zhang, and Victor S. Sheng. Freqmark: Frequency-based watermark for sentence-level detection of llm-generated text. ArXiv eprint 2410.10876, 10 2024.

[24] Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. 06 2023.

[25] Chaoyi Zhu, Jeroen Galjaard, Pin-Yu Chen, and Lydia Y. Chen. Duwak: Dual watermarks in large language models. ArXiv eprint 2403.13000, 03 2024.