XMARK: Reliable Multi-Bit Watermarking for LLM-Generated Texts

Jiahao Xu[1,2] * **Rui Hu**[1] **Olivera Kotevska**[2] **Zikai Zhang**[1]

[1]University of Nevada, Reno
[2]Oak Ridge National Laboratory
{jiahaox, ruihu, zikaiz}@unr.edu kotevskao@ornl.gov

Abstract

Multi-bit watermarking has emerged as a promising solution for embedding imperceptible binary messages into Large Language Model (LLM)-generated text, enabling reliable attribution and tracing of malicious usage of LLMs. Despite recent progress, existing methods still face key limitations: some become computationally infeasible for large messages, while others suffer from a poor trade-off between text quality and decoding accuracy. Moreover, the decoding accuracy of existing methods drops significantly when the number of tokens in the generated text is limited, a condition that frequently arises in practical usage. To address these challenges, we propose XMARK, a novel method for encoding and decoding binary messages in LLM-generated texts. The unique design of XMARK's encoder produces a less distorted logit distribution for watermarked token generation, preserving text quality, and also enables its tailored decoder to reliably recover the encoded message with limited tokens. Extensive experiments across diverse downstream tasks show that XMARK significantly improves decoding accuracy while preserving the quality of watermarked text, outperforming prior methods. The code is at https://github.com/JiiahaoXU/XMark.

1 Introduction

The rapid advancement and widespread adoption of Large Language Models (LLMs), both closed-source (e.g., ChatGPT (Schulman et al., 2022)) and open-source (e.g., LLaMA (Touvron et al., 2023)), have endowed them with remarkable abilities to generate high-quality text. As a result, they have become integral to many text generation applications, such as question answering (Perkins, 2023). However, these powerful generative capabilities also raise significant security and ethical concerns, as malicious users can exploit LLMs to generate harmful content such as fake news, phishing emails, and fraudulent reviews (Zhang et al., 2025).

Recently, researchers have proposed watermarking methods that embed identifiable signals into text, enabling post hoc detection of AI-generated content, i.e., zero-bit watermarking (Liu et al., 2024b; Wu et al., 2025; Pan et al., 2024). However, zero-bit watermarking is insufficient to prevent misuse of AI. To address this limitation, multi-bit watermarking methods have been actively studied, which embed and extract binary messages within text. Such messages can convey richer information, like user IDs, timestamps, and other metadata (Jiang et al., 2025).

Multi-bit watermarking methods can be broadly categorized into two types: (1) *Distortion-free methods*, where the watermarked text follows the same logit distribution as the unwatermarked text (Boroujeny et al., 2024; Hu et al., 2024; Mao et al., 2025; Christ et al., 2024; Kuditipudi et al., 2024; Dathathri et al., 2024; Wu et al., 2024). (2) *Logit-perturbation methods* which instead encode messages by perturbing the logits of selected tokens. Compared with distortion-free methods, they can embed richer watermark information and produce watermarked text that is more robust to text editing (Jiang et al., 2025). These methods generally follow a common paradigm: at each token generation step of LLM, once the model logits are obtained, a hash seed, computed from a hash key and previously-generated token(s), is used to permute the model's vocabulary. From this permuted vocabulary, a subset of tokens (the green list) is selected, and their logits are boosted to increase their sampling probability. The model then samples the next token from the modified logits. During decoding, green-list tokens appear more frequently in the watermarked text, thereby providing the watermark signal needed to recover the encoded message (Kirchenbauer et al., 2023; Yoo et al., 2024;

*This work was done during Jiahao's internship at the Oak Ridge National Laboratory.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14747–14763
July 2-7, 2026

Qu et al., 2025; Li et al., 2024; Fernandez et al., 2023; Wang et al., 2024; Xu et al., 2025).

Early methods such as CycleShift (Fernandez et al., 2023), CTWL (Wang et al., 2024), and DepthW (Li et al., 2024) take the message to be encoded as input to the hash function, yielding a distinct hash seed for vocabulary permutation. However, their decoding process requires brute-force enumeration over all possible message candidates to identify the encoded one, which becomes computationally infeasible for longer messages. To address this, MPAC (Yoo et al., 2024) introduces a block-wise method that divides the message into multiple blocks and encodes/decodes one block per token. Nonetheless, its encoder suffers from degraded text quality because it heavily constrains the size of the green list, causing noticeable distortion in token sampling probabilities. A recent method, StealthInk (Jiang et al., 2025), improves text quality by directly boosting the sampling probabilities of tokens with larger logits while eliminating the chance of sampling tokens with smaller logits, thereby better preserving quality. However, this comes at the cost of weakening the watermark signal and decreasing decoding accuracy. Importantly, we observe that all existing methods rely on the availability of a sufficient number of tokens in the suspect text for reliable decoding. In practice, however, the length of the suspect text may be limited, which can cause a significant drop in decoding accuracy.

In this work, we propose XMARK, which leverages green lists a*cross* (X) multiple vocabulary permutations to improve the text quality and decoding accuracy of multi-bit waterMARKing. Specifically, XMARK follows the block-wise encoding and decoding scheme as in (Yoo et al., 2024). The core innovation of XMARK's encoding process is its use of k distinct hash keys to generate k permutations of the vocabulary. Each permutation is partitioned into 2^d shards, where d is the length of a message block. For each permutation, a green list is formed by unioning all shards except the one indexed by the decimal value of the message block to be encoded, a paradigm we refer to as Leave-one-Shard-out (LOSO). An *evergreen* list is constructed by intersecting all k green lists, and a positive bias is added to its tokens' logits to boost their probabilities to be sampled. This method ensures the size of the evergreen list is proportional to $(1-2^{-d})^k$ of the vocabulary, thereby largely preserving the quality of watermarked texts.

A novel decoder is designed with a *constrained token–shard mapping matrix* (cTMM) to enhance decoding accuracy, particularly when only a limited number of generated tokens are available. For each token, the decoder reconstructs the same k permutations and their corresponding shard partitions, incrementing by at most one the count of the shard that the token belongs to in the cTMM. Through this process, each token contributes to updating the cTMM up to k times, providing a more accurate estimation of the mapping between a token and its originating shard and amplifying the distinction between the boosted shards (from which tokens are more likely drawn) and the unboosted shard (which encodes the message). This design effectively enhances token–shard mapping construction and hence improves decoding reliability under limited-token conditions. Extensive evaluations demonstrate that XMARK consistently achieves higher decoding accuracy while maintaining text quality comparable to existing approaches, especially under limited-token settings.

2 Preliminaries and Background

2.1 Problem Formulation

We consider the widely studied multi-bit watermarking setting (Yoo et al., 2024; Qu et al., 2025; Xu et al., 2025), where a cloud-hosted LLM-as-a-Service provider stamps each model-generated response with a watermark. Given an LLM f, a user prompt $\mathbf{x}_p$, and a b[1]-bit binary message $\mathbf{m} \in \{0,1\}^b$ associated with identifying information (such as user ID, timestamp, or other metadata (Jiang et al., 2025)), the task is to encode $\mathbf{m}$ into the model's output to produce a watermarked text $\mathbf{x}'_g$ via an *encoder*: $\mathbf{x}'_g = \texttt{Enc}(f, \mathbf{x}_p, \mathbf{m})$, while preserving fluency and semantics relative to the unwatermarked output $\mathbf{x}_g$. When a suspicious text is later encountered, a *decoder* can be applied to recover the encoded message: $\mathbf{m}' = \texttt{Dec}(\mathbf{x}'_g)$, enabling provenance verification and attribution of misuse to the originating account. The objective of multi-bit watermarking is to maximize the probability of exact message recovery subject to a minimum text quality constraint:

$$\begin{aligned} \max \quad & \Pr\left[\mathbf{m}' = \mathbf{m} \mid \mathbf{m}' = \texttt{Dec}(\texttt{Enc}(f, \mathbf{x}_p, \mathbf{m}))\right] \\ \text{s.t.} \quad & \texttt{Quality}(\mathbf{x}'_g \mid \mathbf{x}_p) \geq \tau, \end{aligned}$$

[1]Throughout, we assume $b > 0$ and is an even number.

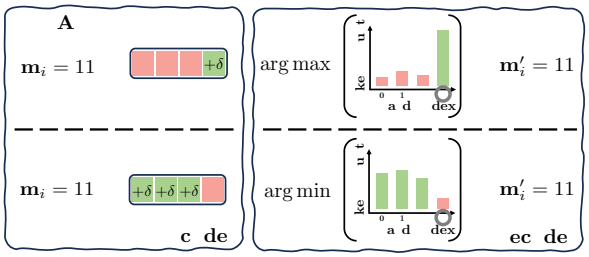

Figure 1: Illustration of the encoder and decoder for MPAC (top) and LOSO (bottom).

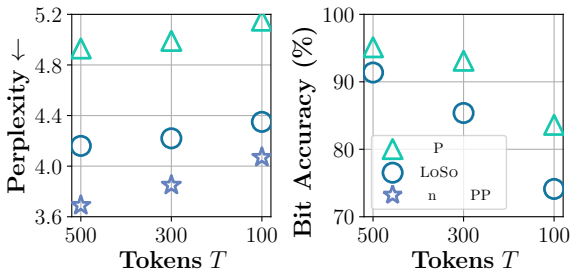

Figure 2: Left: Perplexity vs. T for MPAC, LOSO, and clean (unwatermarked) text. Right: Bit accuracy vs. T for MPAC and LOSO. Note that we do not report bit accuracy for clean text, since it corresponds to the false-positive case. Results are from the text completion task on LLaMA-2-7B (Touvron et al., 2023) using C4 dataset (Raffel et al., 2020), with message length $b = 16$ under varying output sequence lengths T.

where $\texttt{Quality}(\cdot)$ measures text quality and τ is a threshold calibrated to approximate the quality of $\mathbf{x}_g$. In practice, conditional perplexity $\texttt{PPL}(\cdot)$ is a common choice for $\texttt{Quality}(\cdot)$. This formulation ensures highly reliable message recovery with negligible degradation in text quality.

2.2 Classic Solution

To solve the above problem, the encoder of the classic multi-bit watermarking method MPAC (Yoo et al., 2024) divides the message $\mathbf{m}$ into r blocks $\{\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_{r-1}\}$, each containing $d \geq 2$ bits.

Encoder: For generating an output sequence of T tokens, in the t-th ($t \in [T]$) token generation step of the model f, the encoding process of MPAC includes the following three steps:

① Logits generation: The model computes the logits vector $\ell_t = [\ell_t^v]_{v \in \mathcal{V}}$, where $\ell_t^v = f(v \mid \mathbf{x}_p, \mathbf{x}_{:t-1})$ denotes the logit score assigned to token v in its vocabulary $\mathcal{V}$ of size V and $\mathbf{x}_{:t-1} = \{x_1, x_2, \ldots, x_{t-1}\}$ denotes the generated tokens.

② Logits perturbation: A message block $\mathbf{m}_i$ is pseudo-randomly selected (e.g., $i = x_{t-1} \mod r$). The encoder then computes a hash seed $\mathbf{s} = \texttt{Hash}(x_{t-1}, \mathbf{k})$, where $\texttt{Hash}(\cdot)$ is a hash function and $\mathbf{k}$ is a pre-defined hash key. This seed is used to permute the vocabulary, producing a permuted vocabulary $\mathcal{V}'$. Next, the encoder partitions $\mathcal{V}'$ into 2^d disjoint shards $\{\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{2^d-1}\}$, and designates the $[\mathbf{m}_i]_{10}$-th shard as the green list $\mathcal{G}$, where $[\cdot]_{10}$ denotes the decimal value of the binary message block. This results in a green list ratio of $\gamma = |\mathcal{G}|/|\mathcal{V}| = 2^{-d}$. Finally, the logits of tokens in $\mathcal{G}$ are boosted by adding a watermarking bias $\delta > 0$, yielding perturbed logits ℓ_t'.

A concrete example is shown in Figure 1. Suppose the block length is $d = 2$ and the selected message block is $\mathbf{m}_i = 11$. The permuted vocabulary $\mathcal{V}'$ is then divided into $2^d = 4$ shards $\{\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$, and since $[11]_{10} = 3$, $\mathcal{S}_3$ is selected as $\mathcal{G}$, and its logits are perturbed.

③ Token sampling: With the perturbed logits ℓ_t', the model completes the token generation by converting ℓ_t' into a probability distribution over $\mathcal{V}$ via $\texttt{softmax}(\cdot)$ and sampling a token x_t' from it. The encoder repeats these three steps for each generation step and finally obtains the watermarked sequence $\mathbf{x}_g' = \{x_t'\}_{t=1}^T$.

Decoder: Given a suspect text sequence $\mathbf{x}_s$ consisting of T tokens, the MPAC decoder examines each token $x_t' \in \mathbf{x}_s$. For each token, it first determines the index i of the message block that the token corresponds to, and reconstructs the shard partitions using the hash key $\mathbf{k}$. It then updates a token-shard mapping matrix (TMM) $\mathbf{A} \in \mathbb{N}^{r \times 2^d}$ by incrementing $\mathbf{A}[i, u]$ whenever x_t' belongs to the shard $\mathcal{S}_u$. After processing all tokens, the decoder recovers the decimal value of the i-th message block p_i' as

$$p_i' = \arg \max_{u \in \{0, \ldots, 2^d - 1\}} \mathbf{A}[i, u],$$

that is, the index of the shard with the highest token-shard mapping counts. This shard corresponds to $\mathcal{G}$, since $\mathcal{G} = \mathcal{S}_{[\mathbf{m}_i]_{10}}$, and logits of tokens in $\mathcal{G}$ were boosted during encoding, causing them to appear more frequently in the suspect text (see Figure 1). Finally, the block message is recovered by converting p_i' into its d-bit binary representation.

Empirically, MPAC achieves good decoding accuracy when the text sequence is sufficiently long for decoding, as more tokens lead to more accurate token-shard mapping estimates, thereby improving the reliability of correctly identifying the green list. As shown in Figure 2, when $T = 500$, MPAC

Algorithm 1: The Encoder of XMARK

Input : User prompt $\mathbf{x}_p$, LLM f, bias δ, vocabulary $\mathcal{V}$, maximum length T, message $\mathbf{m} \in \{0,1\}^b$, number of blocks r, hash keys $\{\mathbf{k}_j\}_{j=0}^{k-1}$.

Output : Watermarked text $\mathbf{x}'_g$.

1 $\mathbf{x}_{:2} \leftarrow \{x_{p,-2}, x_{p,-1}\}$
2 $\mathbf{m}_0, \ldots, \mathbf{m}_{r-1} \leftarrow \texttt{divide}(\mathbf{m}, r)$
3 $d \leftarrow b/r$
4 **for** $t \leftarrow 1$ **to** T **do**
5 $\quad \ell_t \leftarrow f(\mathbf{x}_p, \mathbf{x}_{3:})$
6 $\quad i \leftarrow (x_{t-2} + x_{t-1}) \bmod r$
7 $\quad p \leftarrow [\mathbf{m}_i]_{10}$
8 $\quad$ **for** $j \leftarrow 0$ **to** $k-1$ **do**
9 $\quad\quad \mathbf{s}_j \leftarrow \texttt{Hash}(x_{t-2}, x_{t-1}, \mathbf{k}_j)$
10 $\quad\quad \mathcal{V}'_j \leftarrow \texttt{permute}(\mathcal{V}, \mathbf{s}_j)$
11 $\quad\quad \mathcal{S}_{j,0}, \ldots, \mathcal{S}_{j,2^d-1} \leftarrow \texttt{part}(\mathcal{V}'_j, 2^d)$
12 $\quad\quad \mathcal{G}_j \leftarrow \mathcal{V}'_j \setminus \mathcal{S}_{j,p}$
13 $\quad$ **end**
14 $\quad \mathcal{E} \leftarrow \bigcap_{j=0}^{k-1} \mathcal{G}_j$
15 $\quad \ell'_t \leftarrow \ell_t$
16 $\quad$ **for** $v \in \mathcal{E}$ **do**
17 $\quad\quad \ell'^{,v}_t \leftarrow \ell'^{,v}_t + \delta$
18 $\quad$ **end**
19 $\quad$ Sample $x'_t \sim \texttt{Softmax}(\ell'_t)$
20 $\quad$ Append x'_t to $\mathbf{x}$
21 **end**
22 **Return** $\mathbf{x}'_g = \mathbf{x}_{3:}$

reaches a bit accuracy (i.e., the proportion of bits correctly decoded from the text) of 95.12%. However, when T decreases to 100, the bit accuracy drops significantly to 83.62%. Moreover, another major limitation of MPAC is that it largely degrades the quality of the generated texts: MPAC significantly increases the perplexity compared to the unwatermarked text. In fact, a larger green list ratio better preserves the quality of watermarked text, since it induces less distortion in the overall logits distribution (Qu et al., 2025; Xu et al., 2025). However, MPAC is constrained to $\gamma \leq 0.25$ for any $d \geq 2$ by design, leading to a sharp distortion.

3 Our Method: XMARK

We propose XMARK, a novel multi-bit watermarking method that incorporates three key designs (LOSO, evergreen list, and constrained TMM) to improve text quality and decoding accuracy.

3.1 Leave-one-shard-out Watermarking

Recall that in MPAC, the vocabulary shard indexed by the block value $[\mathbf{m}_i]_{10}$ is selected as the green list. In contrast, we propose a method called Leave-one-Shard-out (LOSO) watermarking, which reverses this choice by marking *all* shards *except* the $[\mathbf{m}_i]_{10}$-th as green (leaving $[\mathbf{m}_i]_{10}$-th shard unperturbed). For example (see Figure 1), when $d = 2$ and $\mathbf{m}_i = 11$, $\mathcal{S}_0$, $\mathcal{S}_1$, and $\mathcal{S}_2$ form the green list while $\mathcal{S}_3$ is excluded. During decoding, the block value is inferred as the index of the shard with *fewest* token-shard mapping counts, i.e., the excluded one. This design achieves a larger green list ratio of $\gamma = 1 - 2^{-d} \geq 0.75$ for any $d \geq 2$, compared to $\gamma = 2^{-d}$ in MPAC, thereby substantially improving text quality. As shown in Figure 2, LOSO yields significantly lower perplexity than MPAC and is closer to the quality of unwatermarked text. However, this gain in text quality comes with a cost of reduced decodability, since enlarging the green list weakens the watermark signal. As shown in Figure 2, when $T = 500$, LOSO attains 91.38% bit accuracy versus 95.12% for MPAC (a 3.74% gap). This gap goes up to 9.50% when $T = 100$.

In the following, based on LOSO, we propose a novel method, named XMARK, which preserves the quality of watermarked text while ensuring high decoding accuracy, even when the number of tokens available for decoding is limited. We provide the detailed algorithm of the encoder of XMARK in Algorithm 1. The algorithm of the decoder can be found in the Appendix A.4.

3.2 Encoder of XMARK

We first describe the encoding process of XMARK, and then analyze its advantages in preserving text quality. Specifically, before text generation starts, XMARK first divides the binary message $\mathbf{m}$ into r blocks $\{\mathbf{m}_0, \mathbf{m}_1, \ldots, \mathbf{m}_{r-1}\}$, where each block has $d \geq 2$ bits (Line 2-3). It also prepares k hash keys $\{\mathbf{k}_0, \mathbf{k}_1, \ldots, \mathbf{k}_{k-1}\}$.

At the t-th generation step for token x_t, the encoder of XMARK obtains the model logits ℓ_t based on the user prompt and previous generated tokens (Line 5). It samples a message block $\mathbf{m}_i$ to be embedded, where $i = (x_{t-2} + x_{t-1}) \bmod r$ (Line 6). Given a hash function that takes as input the two previously generated token IDs and a hash key, it obtains k distinct hash seeds $\{\mathbf{s}_j\}_{j=0}^{k-1}$, i.e., $\mathbf{s}_j = \texttt{Hash}(x_{t-2}, x_{t-1}, \mathbf{k}_j), \forall j \in \{0, 1, \ldots, k-1\}$

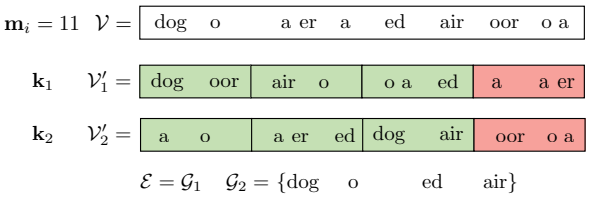

Figure 3: Illustration showing that tokens in $\mathcal{E}$ may originate from different shards across multiple permutations.

(Line 9). Using these seeds, the encoder permutes the vocabulary $\mathcal{V}$ to generate k different permutations: $\{\mathcal{V}'_j\}_{j=0}^{k-1}$ (Line 10). Each permuted vocabulary $\mathcal{V}'_j$ is then evenly partitioned into 2^d shards: $\mathcal{V}'_j = \{\mathcal{S}_{j,0}, \mathcal{S}_{j,1}, \ldots, \mathcal{S}_{j,2^d-1}\}$ (Line 11). For $\mathcal{V}'_j$, the corresponding green list is constructed via LOSO as $\mathcal{G}_j = \mathcal{V}'_j \setminus \mathcal{S}_{j,p}$ (Line 12), where $p = [\mathbf{m}_i]_{10}$ (Line 7). The intersection of all k green lists constructs an *evergreen list* $\mathcal{E} = \bigcap_{j=0}^{k-1} \mathcal{G}_j$ (Line 14). Finally, the logits of all tokens in $\mathcal{E}$ are perturbed by a watermarking bias $\delta > 0$, and the token x'_t is sampled correspondingly (Line 16-19). This process continues until all T tokens are generated.

By using the LOSO strategy, the encoder of XMARK achieves an improved green list ratio, thereby preserving the quality of the watermarked texts. The following theorem characterizes the expected green list ratio of XMARK.

Theorem 1 (γ of XMARK)**.** *The evergreen list $\mathcal{E}$ constructed by* XMARK *satisfies $\mathbb{E}[|\mathcal{E}|] \approx (1 - 2^{-d})^k |\mathcal{V}|$, i.e., $\mathbb{E}[\gamma] = \mathbb{E}[|\mathcal{E}|/|\mathcal{V}|] \approx (1 - 2^{-d})^k$.*

Proof. The proof is in the Appendix A.9. □

Remark 1. *(1) Discussion on d: For a fixed k,* XMARK *produces an exponentially increasing γ with d, thereby improving text quality. In contrast, if* XMARK *adopts MPAC's encoding strategy instead of* LOSO*, the expected γ becomes $\mathbb{E}[\gamma] = (2^{-d})^k$, which decreases exponentially with d and thus severely degrades text quality. In practice, a large d is not preferred, as it increases the number of candidate shards during decoding and thus requires substantially more tokens to estimate token-shard mapping reliably. In this work, we set $d = 2$ for* XMARK*. (2) Discussion on k: When $k = 1$, the encoder of* XMARK *reduces to that of* LOSO *(i.e., $\mathcal{E} = \mathcal{G}_0$). When $k \geq 2$, the γ of* XMARK *gradually decreases. However, the evergreen list design can improve decoding accuracy (see Section 3.3); thus, k serves as a hyperparameter that balances the trade-off between text quality and decoding accuracy.*

3.3 Decoder of XMARK

Here, we first present the details of the decoding process of XMARK and then discuss the importance of the evergreen list and constrained TMM for decoding accuracy.

Given a suspect text $\mathbf{x}_s$, the decoder first initializes a *constrained token-shard mapping matrix* (cTMM) $\mathbf{A} \in \mathbb{N}^{r \times 2^d}$ as all zeros, i.e., $\mathbf{A}^0 = \mathbf{0}$. For each token $x'_t \in \mathbf{x}_s$, the decoder of XMARK first computes current message block index i as $i = (x'_{t-2} + x'_{t-1}) \bmod r$. Given each hash key used in the encoder $\mathbf{k}_j, \forall j \in \{0, 1, \ldots, k-1\}$, it reconstructs each permuted vocabulary $\mathcal{V}'_j$ with the seed $\mathbf{s}_j = \texttt{Hash}(x'_{t-2}, x'_{t-1}, \mathbf{k}_j)$ and partitions $\mathcal{V}'_j$ into 2^d shards $\{\mathcal{S}_{j,u}\}_{u=0}^{2^d-1}$. If $x'_t \in \mathcal{S}_{j,u}$, the decoder updates the cTMM $\mathbf{A}^t \in \mathbb{N}^{r \times 2^d}$ by incrementing $\mathbf{A}[i, u]$ by 1, subject to the *constraint* $\mathbf{A}^t[i,:] - \mathbf{A}^{t-1}[i,:] \in \{0,1\}^{2^d}$. This process continues until all tokens in $\mathbf{x}_s$ are enumerated.

For each block $i \in \{0, \ldots, r-1\}$, the decoder determines the decimal value of i-th block p'_i by

$$p'_i = \arg \min_{u \in \{0,\ldots,2^d-1\}} \mathbf{A}[i, u],$$

since the shard with the *fewest* token–mapping counts is more likely unboosted during encoding. Finally, it converts each p'_i to its d-bit binary representation and concatenates them to recover the full message.

Evergreen List vs LOSO Green List. Recall the decoding challenge of existing methods when given a short suspect text sequence, which cannot provide sufficient tokens for extracting the message. The evergreen list in XMARK provides more observations for the token-shard mapping. Specifically, when $k = 1$, the evergreen list reduces to the LOSO's green list, $\sum \mathbf{A} = T$ indicates exactly T observations that construct the mapping between the token and its originating shard. When $k \geq 2$, the number of observations increases to at most kT. An example is given in Figure 3. With $k = 2$ and $\mathbf{m}_i = 11$, the token "*dog*" in $\mathcal{E}$ belongs to $\mathcal{S}_{1,0}$ of $\mathcal{V}'_1$ and $\mathcal{S}_{2,2}$ of $\mathcal{V}'_2$. This mitigates the problem of inaccurate construction of token-shard mapping due to the limited tokens.

cTMM vs TMM. In existing methods and also our naive LOSO method, a TMM is used for decoding. While intuitive, the TMM results in each token–shard mapping being counted up to k times

in the extreme case, when the evergreen list is used during encoding. For instance, consider a generated token belonging to $\mathcal{V} \setminus \bigcup_{j=0}^{k-1} \mathcal{G}_j$, i.e., it does not appear in any green list across all permutations. In this case, the token would be counted k times into the same unboosted shard (like the shard 3 of LoSo in Figure 1), making it difficult to distinguish between the boosted shards and the unboosted shard, especially when the limited number of tokens does not provide sufficient observations for reliable token-shard mapping construction. cTMM addresses this issue by constraining each token to contribute at most once to any shard, effectively preventing the explosion of counts for the unboosted shard.

4 Empirical Evaluations

4.1 Experimental Settings

General Settings. By default, we consider a total of 50 users, corresponding to 50 randomly generated messages to be encoded. Each user submits two prompts, and each prompt is used to generate a text of $T/2$ tokens, resulting in T tokens per user for decoding. Unless otherwise specified, we set $T = 150$ by default. We compare our proposed methods with five state-of-the-art methods, including DepthW (Li et al., 2024), CycleShift (Fernandez et al., 2023), MPAC (Yoo et al., 2024), RSBH (Qu et al., 2025), and StealthInk (Jiang et al., 2025). For DepthW and CycleShift, we only evaluate the case of $b = 8$, as their decoding time grows exponentially and becomes impractical for larger message lengths. For XMark, we set $d = 2$ and $k = 2$ by default. These settings result in an expected green list ratio of $\mathbb{E}[\gamma] \approx 0.5625$, which is larger than those of existing methods (0.25 for MPAC and CycleShift and 0.5 for DepthW and RSBH). We set the watermarking bias to $\delta = 2$ for all methods by default, except for StealthInk, which does not require δ.

Datasets and Models. We evaluate our method on three widely studied LLM downstream tasks: *text completion* on C4 news (Raffel et al., 2020), *story generation* on WritingPrompts (Fan et al., 2018), and *text summarization* on CNN/DailyMail (Hermann et al., 2015). Unless otherwise specified, we adopt the widely-used open-source LLaMA-2-7B (Touvron et al., 2023) for the text completion task, and the LLaMA-2-7B-chat for the story generation and text summarization tasks. The system prompts used for these tasks are provided in the Appendix A.5.

Metrics. We evaluate the quality of generated text using perplexity (PPL), computed by a larger LLaMA-2-13B model. In addition, we consider the semantic metric BERTScore (BSc.) (Zhang et al., 2020) and the lexical metrics ROUGE-1 (R.-1) and ROUGE-Lsum (R.-Lsum) (Lin, 2004). Following prior practice, these quality metrics are computed by comparing the watermarked text with the corresponding unwatermarked text generated by the same model. For decoding performance, we adopt bit accuracy (BA), following prior works (Zhu et al., 2018; Qu et al., 2025; Yoo et al., 2024; Xu et al., 2025; Jiang et al., 2025), which is defined as the proportion of correctly decoded bits with respect to the ground-truth message. A desirable method should achieve a low PPL and a high BA, BSc., R.-1, and R.-Lsum.

4.2 Results

Main Results. Table 1 reports the comprehensive BA and PPL results of representative multi-bit watermarking methods on three tasks with $b = 8$ and different values of $T \in \{150, 200, 250, 300\}$. As expected, increasing T generally improves BA across all methods and tasks. Notably, our method XMark consistently achieves higher BA than all baselines for every T and task. On the text summarization and story generation tasks, XMark reaches average BAs of 82.75% and 80.94%, outperforming the second-best method MPAC by $+5.81\%$ and $+6.81\%$, respectively. On the text completion task, which is inherently more challenging due to higher token entropy, XMark already achieves a near-perfect BA of 98.75% with only $T = 150$ tokens, a $+3.5\%$ improvement over CycleShift. These results demonstrate XMark's robustness in ensuring reliable decoding with limited generated tokens. This advantage stems from the co-design of the evergreen list and cTMM that enhances the construction of accurate token-shard mapping during decoding.

In addition to improving BA, XMark preserves the quality of generated texts at a level comparable to or better than existing methods. For the text completion task, XMark achieves an average PPL of 4.61, outperforming all methods except StealthInk. However, the high fluency of StealthInk comes at the cost of significantly lower BA. For the text summarization and story generation tasks, XMark attains PPLs of 4.66 and 4.03, which are only 0.07 and 0.34 higher than the unwatermarked baselines,

Table 1: Comparison of methods on three downstream tasks with $b = 8$ under different token budgets $T \in \{150, 200, 250, 300\}$. The average PPL of unwatermarked texts is 3.97, 4.59, and 3.69 for the three tasks, respectively, serving as the lower bounds for the PPLs of watermarked texts.

Method	$T = 150$		$T = 200$		$T = 250$		$T = 300$		Avg.	Avg.
	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓
Text Completion										
CycleShift	95.25	5.17	98.00	5.02	97.00	5.07	98.25	4.96	97.13	5.06
DepthW	79.50	4.62	89.00	4.67	90.50	4.63	95.75	4.62	88.69	4.64
StealthInk	85.00	4.13	89.50	4.21	92.25	4.13	92.50	4.04	89.81	4.13
MPAC	94.00	5.19	94.00	5.00	96.25	5.03	98.25	5.10	95.63	5.08
RSBH	92.75	4.83	96.00	4.83	95.50	4.67	97.75	4.83	95.50	4.79
XMARK	98.75	4.68	99.00	4.61	99.75	4.60	100.00	4.56	99.38	4.61
Text Summarization										
CycleShift	54.50	5.57	59.25	4.82	64.75	4.36	62.25	4.10	60.19	4.71
DepthW	58.50	5.60	55.00	4.76	57.25	4.29	54.00	3.95	56.19	4.65
StealthInk	62.75	5.55	65.00	4.60	69.50	4.26	71.50	3.93	67.19	4.59
MPAC	71.75	5.71	75.75	4.94	78.25	4.36	82.00	4.13	76.94	4.79
RSBH	62.50	5.60	65.00	5.04	63.75	4.56	64.75	4.13	64.00	4.83
XMARK	74.75	5.50	79.00	4.85	87.75	4.33	89.50	3.97	82.75	4.66
Story Generation										
CycleShift	66.00	4.93	66.00	4.21	66.75	3.80	74.00	3.58	68.19	4.13
DepthW	56.25	4.78	59.75	3.99	55.75	3.60	54.75	3.36	56.63	3.93
StealthInk	59.00	4.55	62.50	3.88	69.00	3.44	71.25	3.22	65.44	3.77
MPAC	71.00	4.98	71.25	4.24	76.25	3.83	78.00	3.61	74.13	4.17
RSBH	58.00	4.96	62.75	4.33	65.75	3.85	77.00	3.47	65.88	4.15
XMARK	75.50	4.86	79.75	4.12	82.50	3.68	86.00	3.44	80.94	4.03

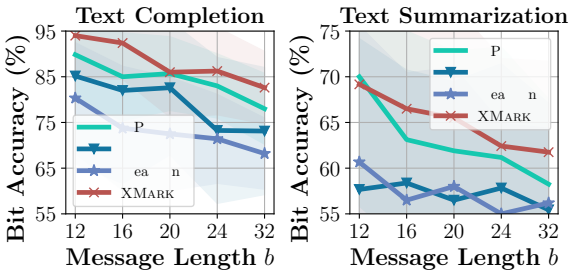

Figure 4: Impact of message length b on BA for MPAC, RSBH, StealthInk, and XMARK on the text completion and text summarization tasks.

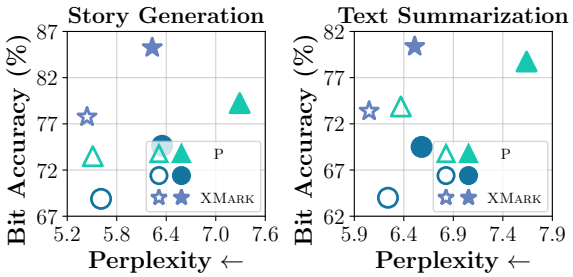

Figure 5: Impact of bias δ on BA and PPL for MPAC, RSBH, and XMARK on the text summarization and story generation tasks. Hollow markers indicate results with $\delta = 3$, while solid markers correspond to $\delta = 4$.

respectively. This strong quality preservation stems from the design of LOSO, which allows XMARK to maintain a longer green list than existing methods.

Impact of Message Length. In practical applications of watermarking, a binary message length of $b = 8$ can encode at most 256 distinct values (e.g., user IDs), which is often insufficient. To evaluate scalability, we extend the message length up to $b = 32$ and examine how different methods perform with longer messages. Figure 4 reports the BA of MPAC, RSBH, StealthInk, and XMARK on the text completion and text summarization tasks. As shown, all methods experience a decrease in BA as b increases. Nevertheless, XMARK consistently outperforms the baselines, maintaining relatively higher BA across different message lengths. On the text completion task, even with $b = 32$, XMARK achieves a BA of 82.62%, which is +4.62% higher than the second-best method, MPAC. On the text summarization task, although the absolute BA of all methods becomes low for large b, XMARK still attains the highest performance, achieving a +3.5% improvement over MPAC when $b = 32$.

Impact of Watermarking Bias. We next investigate the impact of different watermarking biases δ on performance. Theoretically, a larger δ can

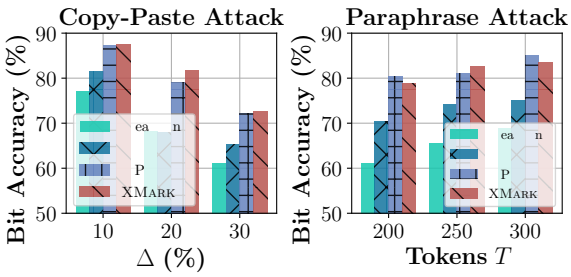

Figure 6: Robustness of methods against text editing attacks on the text completion task with $b = 16$.

increase decoding accuracy, but at the cost of degrading text quality since stronger perturbations are applied to the token logits. We report the BA and PPL of MPAC, RSBH, and XMARK on the text summarization and story generation tasks with $\delta = 3$ and $\delta = 4$, as shown in Figure 5. Overall, XMARK consistently achieves a better trade-off between BA and PPL across both settings. With $\delta = 3$, XMARK attains the highest BA while also preserving the lowest PPL on the story generation task. When δ increases to 4, all methods show improved BA but at the expense of higher PPL, verifying the theoretical analysis of the impact of δ. For instance, on the text summarization task, MPAC's BA increases from 73.88% to 78.55%, while its PPL degrades sharply from 6.37 to 7.64. In contrast, XMARK continues to provide the better trade-off, achieving the highest BA (80.38) with low PPL (6.51) in this setting.

Robustness to Text Editing Attacks. When users receive watermarked texts generated by LLMs, they may edit them to improve fluency or, more intentionally, to attempt watermark removal. We evaluate the robustness of different methods against two widely studied text editing attacks: *Copy-Paste* and *Paraphrase* (Zhang et al., 2024), using the text completion task with $b = 16$. For the Copy-Paste attack, we mix watermarked and human-written texts by randomly interleaving unwatermarked text segments into watermarked texts, following prior work (Yoo et al., 2024; Qu et al., 2025). The proportion of unwatermarked text is controlled by Δ, while keeping the total length fixed. For the Paraphrase attack, we employ the strong paraphraser *Dipper* (Krishna et al., 2023). We report the BA of StealthInk, RSBH, MPAC, and XMARK under varying Δ for Copy-Paste attack and varying number of generated tokens T under Paraphrase attack in Figure 6.

As shown in Figure 6, for Copy-Paste attack, as Δ increases, all methods exhibit a decline in BA, as expected. Despite this degradation, XMARK consistently achieves the highest BA, demonstrating superior robustness. Interestingly, MPAC achieves BA comparable to XMARK, largely because its design fixes the green list ratio at a small level 0.25, resulting in a stronger watermark signal that confers additional robustness against editing. For Paraphrase attack, which remains a significant open challenge in the literature (Jiang et al., 2025), we observe that BA slightly improves for all methods as T increases. Here, MPAC performs better than StealthInk and RSBH, while XMARK achieves a BA comparable to MPAC.

Table 2: Results on the machine translation task using WMT14 German-to-English dataset.

Method	BA↑	PPL↓	R.-1↑	R.-Lsum↑	BLEU↑
RSBH	58.62	7.60	0.61	0.58	38.96
MPAC	60.88	7.54	0.65	0.61	43.21
XMARK	64.62	7.26	0.64	0.61	44.52

Table 3: Performance comparison with long messages ($b = 64$) and long generated texts ($T = 1000$) on the text completion task.

Method	BA↑	PPL↓	R.-1↑	R.-Lsum↑	BSc.↑
RSBH	84.47	4.46	0.36	0.33	0.8266
MPAC	85.81	4.71	0.35	0.32	0.8254
XMARK	93.00	4.50	0.38	0.35	0.8304

Results on Machine Translation Task. We further evaluate XMARK on a downstream machine translation task. Specifically, we use the WMT14 German-to-English dataset (Bojar et al., 2014). In addition to BA and PPL, we also report BSc., R.-1, R.-Lsum, and BLEU (Papineni et al., 2002), where BLEU serves as a standard task-specific metric for translation quality. The results under message length $b = 16$, bias $\delta = 2$, and generation length $T = 150$ are summarized in Table 2. We observe that XMARK achieves the highest BLEU score of 44.52, outperforming the strongest baseline MPAC, which obtains 43.21. This suggests that XMARK imposes the smallest negative impact on the functional utility of the translation task. Meanwhile, XMARK also achieves the highest BA of 64.62%, showing that it provides the best trade-off between decodability and task performance.

Longer Message, Longer Text, and More Semantic Metrics. We further evaluate all methods under a larger-scale setting with a long message

Table 4: Comparison of runtime (in seconds) and BA under the default setting.

Method	Encoding	Decoding	BA↑
DepthW	11.51	16.16	79.50
MPAC	10.93	0.06	94.00
RSBH	11.80	8.45	92.75
StealthInk	11.47	0.07	85.00
XMARK	11.54	0.08	98.75

length ($b = 64$) and a long generated text length ($T = 1000$) on the text completion task. In addition to BA and PPL, we also report BSc. as well as R.-1 and ROUGE-Lsum (R.-Lsum) (Lin, 2004). The results are summarized in Table 3. We can see that XMARK achieves the highest BA by a clear margin, improving from 85.81% with MPAC to 93.00%. At the same time, it also provides the best overall text quality, achieving the highest BSc., R.-1, and R.-Lsum, while maintaining a low PPL comparable to the baselines. These results show that XMARK remains highly effective for long messages, while better preserving both the semantic meaning and lexical content of watermarked texts.

Runtime Analysis. We report the average encoding and decoding time under the default setting, with results shown in Table 4. The encoding time of all methods is around 11.5 seconds, suggesting that the overall latency is mainly dominated by the LLM inference process itself. This also shows that XMARK introduces negligible additional overhead during generation. For decoding, XMARK remains highly efficient, requiring only 0.08 seconds, which is comparable to the fastest baseline, MPAC (0.06 seconds). By contrast, DepthW and RSBH are much slower, taking 16.16 and 8.45 seconds, respectively, even for the short message length of $b = 8$, mainly because they rely on candidate enumeration. These results show that XMARK effectively overcomes the computational inefficiency of prior methods while still achieving substantially higher BA (98.75%).

Ablation Study. We conduct an ablation study on XMARK by varying the number of hash keys k from 1 to 4. Note that when $k = 1$, XMARK reduces to LOSO. By default, XMARK employs the cTMM in decoding. For comparison, we also evaluate XMARK with the TMM in decoding, denoted as XMARK$^-$. Experiments are conducted on the text completion task with $b = 32$, with T set to 150 and 250, and the results are summarized in Table 5. We ignore the PPL results for XMARK$^-$ as they are the same as XMARK's PPL. The findings are clear: as k increases, XMARK achieves consistently higher BA, since additional keys allow more observations to accurately estimate the mapping between each generated token and its originating shard during decoding. However, this also slightly increases PPL due to the mildly reduced green list ratio. Compared with XMARK, XMARK$^-$ exhibits marginally lower BA due to the over-counting of the unboosted shard in TMM, especially when the number of generated tokens is small. Overall, the results indicate that cTMM is a more effective token-shard mapping construction method for XMARK.

Table 5: Ablation study of XMARK with different numbers of hash keys (k) and different token–shard mapping matrices: cTMM in XMARK vs. TMM in XMARK$^-$.

Method	$T = 150$		$T = 250$		Avg.	Avg.
	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓
LOSO	74.06	4.45	81.38	4.46	77.72	4.46
XMARK (k=2)	82.62	4.84	88.00	4.60	85.31	4.72
XMARK (k=3)	82.94	5.01	89.38	4.97	86.16	4.99
XMARK (k=4)	84.25	5.07	92.06	5.14	88.16	5.11
XMARK$^-$ (k=2)	81.31	–	88.06	–	84.69	–
XMARK$^-$ (k=3)	80.69	–	87.75	–	84.22	–
XMARK$^-$ (k=4)	80.31	–	89.56	–	84.94	–

Additional Results. Results on more widely-used public LLMs and datasets are provided in Appendix A.7. We also discuss how XMARK handles *false positive cases* in Appendix A.6.

5 Conclusion

We propose a novel multi-bit watermarking method, XMARK, for LLM-generated texts. During encoding, XMARK leverages distinct hash keys to generate multiple permutations of the vocabulary, from which an evergreen list is constructed by intersecting the green lists across permutations. Tokens in the evergreen list are then boosted to increase their sampling probability. During decoding, each token can update multiple shard observations under the cTMM constraint, thereby providing a more accurate estimation of the mapping between a generated token and its originating shard. This enhanced counting mechanism ensures a more reliable decoding process, especially when the number of tokens for decoding is limited. Extensive experiments on diverse LLM downstream tasks and settings demonstrate that XMARK substantially improves decoding accuracy compared with other methods, while preserving the quality of watermarked texts.

Limitations

We now discuss the limitations and potential future directions of our work.

First, during the decoding process of XMARK, each row of the cTMM has a size of 1×2^d, resulting in a total of $r \times 2^d$ shards available for token–shard mappings. In our experiments, we fix $d = 2$. Increasing d substantially expands the number of shard candidates, which in turn reduces the number of observations/counts for the token–shard mappings per shard, given a fixed k and T, and decreases the accuracy of estimation, thereby degrading the decoding accuracy. Adapting XMARK to larger d values is an important future direction, as increasing d can significantly enhance the quality of watermarked text, as discussed in Remark 1.

Second, although XMARK demonstrates robustness against text editing attacks in our evaluation, its design does not explicitly incorporate mechanisms to enhance robustness against attacks. Introducing such mechanisms represents another promising avenue for future research.

Third, as discussed in Remark 1, the hyperparameter k controls the trade-off between text quality and decoding accuracy. Although we conducted an ablation study on k, in practice, the optimal value still requires manual tuning. Designing an adaptive strategy to automatically determine the optimal k is another valuable direction for future exploration.

Finally, in our experiments, all methods are evaluated under a fixed δ. In fact, δ also plays an important role in balancing text quality and decoding accuracy. Specifically, a larger δ yields more reliable decoding but may degrade the quality of the generated text. Jointly optimizing XMARK with respect to δ constitutes an additional important direction for future work.

Acknowledgments

We thank the anonymous reviewers and the area chair for their constructive comments. The work of Jiahao, Rui, and Zikai was supported in part by the National Science Foundation under Grant No. 2511989. This material is based upon work co-supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Contract No. DE-AC05-00OR22725. This manuscript has been co-authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

References

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Massieh Kordi Boroujeny, Ya Jiang, Kai Zeng, and Brian Mark. 2024. Multi-bit distortion-free watermarking for large language models. *arXiv preprint arXiv:2402.16578*.

Yapei Chang, Kalpesh Krishna, Amir Houmansadr, John Frederick Wieting, and Mohit Iyyer. 2024. PostMark: A robust blackbox watermark for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8969–8987, Miami, Florida, USA. Association for Computational Linguistics.

Miranda Christ, Sam Gunn, and Or Zamir. 2024. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1125–1139. PMLR.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, and 1 others. 2024. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

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. IEEE.

Eva Giboulot and Teddy Furon. 2024. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. *Advances in Neural Information Processing Systems*, 37:18848–18881.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2024. Unbiased watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Ya Jiang, Chuxiong Wu, Massieh Kordi Boroujeny, Brian Mark, and Kai Zeng. 2025. StealthInk: A multi-bit and stealthy watermark for large language models. In *Forty-second International Conference on Machine Learning*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*, pages 17061–17084. PMLR.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. On the reliability of watermarks for large language models. In *The Twelfth International Conference on Learning Representations*.

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. *Advances in Neural Information Processing Systems*, 36:27469–27500.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*.

Liying Li, Yihan Bai, and Minhao Cheng. 2024. Where am I from? identifying origin of LLM-generated content. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12218–12229, Miami, Florida, USA. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024a. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip Yu. 2024b. A survey of text watermarking in the era of large language models. *ACM Computing Surveys*, 57(2):1–36.

Minjia Mao, Dongjun Wei, Zeyu Chen, Xiao Fang, and Michael Chau. 2025. Watermarking large language models: An unbiased and low-risk method. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7939–7960, Vienna, Austria. Association for Computational Linguistics.

Travis Munyer, Abdullah All Tanvir, Arjon Das, and Xin Zhong. 2024. Deeptextmark: a deep learning-driven text watermarking approach for identifying large language model generated text. *IEEE ACCESS*, 12:40508–40520.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, and 1 others. 2024. Markllm: An open-source toolkit for llm watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–71.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Mike Perkins. 2023. Academic integrity considerations of ai large language models in the post-pandemic era: Chatgpt and beyond. *Journal of University Teaching and Learning Practice*, 20(2):1–24.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. 2025. Markmywords: Analyzing and evaluating language model watermarks. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pages 68–91. IEEE.

Wenjie Qu, Wengrui Zheng, Tianyang Tao, Dong Yin, Yanze Jiang, Zhihua Tian, Wei Zou, Jinyuan Jia, and Jiaheng Zhang. 2025. Provably robust multi-bit watermarking for AI-generated text. In *34th USENIX Security Symposium (USENIX Security 25)*, pages 201–220.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Christoph Schuhmann. 2023. Essays with instructions dataset. https://huggingface.co/datasets/ChristophSchuhmann/essays-with-instructions. Accessed: 2025-07-14.

John Schulman, Barret Zoph, Christina Kim, Jacob Hilton, Jacob Menick, Jiayi Weng, Juan Felipe Ceron Uribe, Liam Fedus, Luke Metz, Michael Pokorny, and 1 others. 2022. ChatGPT: Optimizing language models for dialogue. *OpenAI blog*, 2(4).

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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 *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zongqi Wang, Tianle Gu, Baoyuan Wu, and Yujiu Yang. 2025. Morphmark: Flexible adaptive watermarking for large language models. *arXiv preprint arXiv:2505.11541*.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Lidia Sam Chao, and Derek Fai Wong. 2025. A survey on llm-generated text detection: Necessity, methods, and future directions. *Computational Linguistics*, 51(1):275–338.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. 2024. A resilient and accessible distribution-preserving watermark for large language models. In *Forty-first International Conference on Machine Learning*.

Jiahao Xu, Rui Hu, and Zikai Zhang. 2025. Majority bit-aware watermarking for large language models. *arXiv preprint arXiv:2508.03829*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

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: Human Language Technologies (Volume 1: Long Papers)*, pages 4031–4055, Mexico City, Mexico. Association for Computational Linguistics.

Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2024. REMARK-LLM: A robust and efficient watermarking framework for generative large language models. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 1813–1830.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zhaoxi Zhang, Xiaomei Zhang, Yanjun Zhang, He Zhang, Shirui Pan, Bo Liu, Asif Qumer Gill, and Leo Yu Zhang. 2025. Character-level perturbations disrupt llm watermarks. *arXiv preprint arXiv:2509.09112*.

Jiren Zhu, Russell Kaplan, Justin Johnson, and Li Fei-Fei. 2018. Hidden: Hiding data with deep networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 657–672.

Table 6: Notation table.

Symbol	Description
γ	The green list ratio
$\mathbf{m}$	The binary multi-bit message
b	The length of message $\mathbf{m}$
f	The large language model
$\mathbf{x}_p$	The input prompt
$\mathbf{x}_s$	The suspect text
x_t	The t-th token generated by f
$\mathcal{V}$	The vocabulary of f
$\mathbf{x}_g$	The generated text of f without watermark
$\mathbf{x}'_g$	The generated text of f with watermark
T	The generation step of f and length of $\mathbf{x}_g$
$\mathcal{G}$	The green list
$\mathcal{S}$	The token shard
δ	The watermarking bias
$\texttt{Enc}(\cdot)$	The encoding function
$\texttt{Dec}(\cdot)$	The decoding function
$\mathbf{m}'$	The decoded message via $\texttt{Dec}(\cdot)$
$\texttt{Quality}(\cdot)$	The quality function
$\texttt{PPL}(\cdot)$	The conditional perplexity
τ	The tolerance margin on text quality
$\mathbf{k}$	The secret hash key
k	The number of hash keys
$\texttt{Hash}(\cdot)$	The Hash function
$\mathbf{s}$	The pseudo-random seed generated by Hash
r	The number of blocks
d	The number of bits in a message block
$\mathbf{A}$	The token-shard mapping matrix
p	The decimal value of a message block
$\mathcal{E}$	The evergreen list

A Appendix

A.1 Notation Table

We present the detailed notation table in Table 6 for the reader's convenience.

A.2 Hardware Settings.

All experiments were carried out on a self-managed Linux-based computing cluster running Ubuntu 20.04.6 LTS. The cluster is equipped with eight NVIDIA RTX A6000 GPUs (each with 48 GB of memory) and AMD EPYC 7763 CPUs featuring 64 cores. Model inference leveraged GPU acceleration extensively. In total, the experiments accumulated roughly two weeks of GPU compute time.

A.3 Related Work

Zero-bit Watermarking for LLMs. The first zero-bit watermarking approach, KGW, was proposed by Kirchenbauer et al. (2023). At each generation step t, after the language model f produces the logit vector for the next token, a pseudo-random seed is deterministically derived using a hash function that takes as input a hash key $\mathbf{k}$ and the previously generated token x_{t-1}. This seed governs a fixed permutation and subsequent partitioning of the vocabulary $\mathcal{V}$ into two disjoint subsets: the green list $\mathcal{G}$ and the red list $\mathcal{R}$. A positive bias δ is then added to the logits corresponding to tokens in $\mathcal{G}$, increasing their likelihood under the $\texttt{softmax}(\cdot)$ distribution. While a larger δ strengthens the watermark signal, it also amplifies the distortion in the token sampling distribution, potentially degrading the fluency and naturalness of the generated text. Repeating this process at every generation step produces a watermarked sequence $\mathbf{x}'_g$. For verification, given a suspect text $\mathbf{x}_s$, the decoder reconstructs the green list for each token position using the same key and hashing procedure, and then conducts a statistical z-test to evaluate whether the observed proportion of tokens from $\mathcal{G}$ significantly exceeds the expected baseline, thereby determining the presence of the watermark. A statistically significant excess over the expected frequency indicates the presence of the watermark. Several subsequent works have improved this zero-bit watermarking method to achieve higher decoding accuracy and better utility of the watermarked text (Kirchenbauer et al., 2024; Kuditipudi et al., 2024; Wang et al., 2025; Chang et al., 2024; Giboulot and Furon, 2024; Liu et al., 2024a; Piet et al., 2025; Christ et al., 2024; Munyer et al., 2024).

Multi-bit Watermarking for LLMs. While zero-bit watermarking enables text verification, it is insufficient for traceable watermarking, motivating the development of multi-bit watermarking methods. CycleShift (Fernandez et al., 2023) cyclically shifts vocabulary permutations according to the embedded message and biases the tokens within the resulting green list. However, overlapping shifts can introduce interference among message bits, reducing distinctiveness and weakening statistical separability (Jiang et al., 2025). DepthW (Li et al., 2024) directly encodes the message as input to the hash function and further sets a dark green list inside of the green list, to which stronger perturbations are applied. Despite their effectiveness, both CycleShift and DepthW rely on brute-force search over all possible message candidates, rendering them impractical for long messages.

To improve decoding efficiency, MPAC (Yoo et al., 2024) partitions the message into multiple blocks and uses each block's value to guide green

Algorithm 2: The Decoder of XMARK

Input : Vocabulary $\mathcal{V}$, hash keys $\{\mathbf{k}_0, \mathbf{k}_1, \ldots, \mathbf{k}_{k-1}\}$, number of blocks r, length of message block d, suspect text $\mathbf{x}'$ of length T

Output : Decoded message $\mathbf{m}'$

1 Initialize $\mathbf{A} \leftarrow \mathbf{0}^{r \times 2^d}$
2 **for** $t = 3$ **to** T **do**
3 $i \leftarrow (x'_{t-2} + x'_{t-1}) \mod r$
4 $\mathbf{cc} \leftarrow \mathbf{0}^{1 \times 2^d}$
5 **for** $j \leftarrow 0$ **to** $k-1$ **do**
6 $\mathbf{s}_j \leftarrow \texttt{Hash}(x'_{t-2}, x'_{t-1}, \mathbf{k}_j)$
7 $\mathcal{V}' \leftarrow \texttt{permute}(\mathcal{V}, \mathbf{s}_j)$
8 $\mathcal{S}_{j,0}, \ldots, \mathcal{S}_{j,2^d-1} \leftarrow \texttt{part}(\mathcal{V}', 2^d)$
9 **if** $x'_t \in \mathcal{S}_{j,u}$ **then**
10 $\mathbf{cc}[u] \leftarrow \mathbf{cc}[u] + 1$
11 **end**
12 **end**
13 $\mathbf{update} \leftarrow \texttt{sign}(\mathbf{cc})$
14 $\mathbf{A}[i,:] \leftarrow \mathbf{A}[i,:] + \mathbf{update}$
15 **end**
16 **for** $i \leftarrow 0$ **to** $r-1$ **do**
17 $p'_i \leftarrow \arg\min \mathbf{A}[i:]$
18 $\mathbf{m}'_i \leftarrow \texttt{binary}(p'_i)$
19 **end**
20 **Return** $\mathbf{m}' = \texttt{concat}(\mathbf{m}'_0, \ldots, \mathbf{m}'_{r-1})$

list construction. However, as discussed in this paper, MPAC substantially degrades text quality. RSBH (Qu et al., 2025) enhances MPAC by incorporating the message block value into the hash seed computation and adopting a larger green list ratio $\gamma = 0.5$. This design improves text utility but increases decoding complexity exponentially with 2^d. More recently, StealthInk (Jiang et al., 2025) improves text quality by directly amplifying the sampling probabilities of high-logit tokens while suppressing low-logit ones, thereby preserving fluency. Nevertheless, this approach weakens the watermark signal and consequently reduces decoding accuracy.

A.4 Decoder Algorithm

We present the detailed decoding procedure of XMARK in Algorithm 2. Note that we start from the third token during decoding because computing the block index and hash seed requires x'_{t-2} and x'_{t-1}. In Line 13, the function $\texttt{sign}(\cdot)$ operates element-wise, setting any value greater than 0 to 1 while keeping 0 unchanged, thereby producing the constrained token-shard mapping matrix (cTMM).

A.5 System Prompt Settings

Recall that in our work, we use the LLaMA-2-7B-chat model (Touvron et al., 2023) for story generation and text summarization tasks. The chat prompt for story generation is designed to encourage coherent and imaginative story generation and is defined as follows:

> `[System]` *You are a helpful assistant that writes engaging and coherent stories.*
>
> `[User]` *Please write a detailed and imaginative short story based on the following prompt: [prompt].*

For the text summarization task, the chat prompt is designed to instruct the model to generate concise and faithful summaries given an input article, using the following format:

> `[System]` *You are a helpful assistant specialized in summarization. You take a document and write a concise, faithful summary.*
>
> `[User]` *Please summarize the following article in a few sentences: [article].*

A.6 Discussion on False Positive Cases

False positives are a common issue across all multi-bit watermarking methods. Specifically, even when applied to unwatermarked text, a decoder may still output a message that could be mistakenly interpreted as a valid watermark. However, this problem has received little attention in the existing literature. We observe that our method, XMARK, is inherently robust to false positives. Recall that during decoding, XMARK leverages the cTMM to recover the embedded message. Our recovery strategy identifies the shard with the fewest token–shard mapping counts, and this statistical property naturally serves as an indicator of false positives: if no shard exhibits a noticeably small count, the text is likely unwatermarked. In practice, one can set a threshold based on the entropy of the token–shard count distribution for each message block. If the entropy exceeds this threshold, the decoder concludes that the text is unwatermarked; otherwise, it proceeds with message recovery as usual.

We further conduct empirical experiments to evaluate XMARK's ability to detect false positives,

Table 7: Performance comparison on text completion tasks with message length $b = 16$ under different numbers of available tokens $T \in \{150, 200, 250, 300\}$ on the Essays and OpenGen datasets.

Task (Dataset)	Method	$T = 150$		$T = 200$		$T = 250$		$T = 300$		Avg.	Avg.
		BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓
Text Completion (Essays)	StealthInk	71.62	4.12	75.00	3.90	78.88	3.75	79.25	3.66	76.19	3.86
	MPAC	85.88	5.37	89.12	5.09	91.75	4.93	92.88	4.78	89.91	5.04
	RSBH	80.12	4.97	85.00	4.80	89.75	4.66	91.00	4.53	86.47	4.74
	XMARK	92.62	4.88	95.62	4.68	97.12	4.50	97.75	4.40	95.78	4.62
Text Completion (OpenGen)	StealthInk	72.75	4.11	75.75	3.98	77.88	4.04	80.12	3.85	76.63	3.99
	MPAC	84.88	5.01	89.25	4.98	90.38	4.97	92.62	4.84	89.28	4.95
	RSBH	77.00	4.73	81.75	4.75	88.25	4.72	87.12	4.56	83.53	4.69
	XMARK	90.62	4.70	91.12	4.68	95.25	4.66	95.88	4.67	93.22	4.68

Table 8: Performance comparison on text completion tasks with message length $b = 32$ under different numbers of available tokens $T \in \{150, 200, 250, 300\}$ on the Essays and OpenGen datasets.

Task (Dataset)	Method	$T = 150$		$T = 200$		$T = 250$		$T = 300$		Avg.	Avg.
		BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓
Text Completion (Essays)	StealthInk	65.94	4.17	67.76	3.96	71.12	3.84	72.12	3.74	69.24	3.93
	MPAC	77.81	5.03	80.38	4.89	80.94	4.76	83.19	4.65	80.58	4.83
	RSBH	75.06	5.07	80.38	4.77	82.00	4.67	84.06	4.54	80.38	4.76
	XMARK	82.38	4.89	85.38	4.65	89.31	4.44	91.25	4.32	87.08	4.58
Text Completion (OpenGen)	StealthInk	67.12	4.22	67.88	4.03	69.94	4.07	71.69	4.01	69.16	4.08
	MPAC	76.06	4.97	79.31	4.91	82.69	4.97	84.94	4.99	80.75	4.96
	RSBH	70.88	4.67	76.06	4.82	78.62	4.62	77.56	4.71	75.78	4.71
	XMARK	79.50	4.81	83.88	4.52	85.81	4.75	90.62	4.68	84.95	4.69

and compare it with CycleShift, MPAC, and StealthInk under the setting of $b = 8$, $T = 150$, and $\delta = 2$. For each baseline, we follow the detection metric used in its original paper, such as the p-value for CycleShift and the z-score for MPAC, and vary the corresponding decision threshold to measure detection performance. For XMARK, given the well-updated cTMM $\mathbf{A}^T$, we compute the standard deviation of each row (corresponding to each message block) and then take the average. We report the TPR at a fixed 10% FPR, together with the F1 score, in Table 9. Note that RSBH and DepthW are not included in this comparison, as their original papers do not provide explicit false-positive detection modules or statistical thresholds for unwatermarked text detection. From the results, XMARK achieves strong detection performance, with a TPR of 94% at 10% FPR and an F1 score of 0.92. It substantially outperforms CycleShift and StealthInk, and remains competitive with MPAC, which achieves slightly higher detection metrics. We note that MPAC benefits from a stricter green list ratio ($\gamma = 0.25$), which produces a stronger detection signal but typically comes at the cost of text quality. Overall, these results show that XMARK provides a favorable balance between detection reliability and generation quality.

A.7 More Results

More Results on Additional Text Completion Tasks. We further evaluate MPAC, RSBH, StealthInk, and our proposed XMARK on the text completion task using the OpenGen (Krishna et al., 2023) and Essays (Schuhmann, 2023) datasets. Results for message lengths $b = 16$ and $b = 32$ with $T \in \{150, 200, 250, 300\}$ are summarized in Table 7 and Table 8, respectively. Across all T settings, XMARK consistently achieves higher BA than the compared methods. When $b = 16$, on Essays and OpenGen, XMARK attains the highest average BA of 95.78% and 93.22%, yielding gains of $+5.87\%$ and $+3.94\%$ over the second-best method (MPAC). In addition to BA improvements, XMARK also demonstrates clear PPL reductions, achieving PPL values of 4.62 and 4.68, respectively. When $b = 32$, the advantage of XMARK becomes even more pronounced, as the BA gaps over MPAC further widen to $+6.50\%$ on Essays and $+4.20\%$ on OpenGen.

More Results on Additional LLMs. We further evaluate our method on two widely used language models: Qwen2.5-7B (Yang et al., 2024) and

Table 9: False positive detection performance measured by TPR at 10% FPR (TPR@10%FPR) and F1 score.

Method	Detection Mechanism and Threshold	TPR@10%FPR ↑	F1 Score ↑
CycleShift	p-value ($p < 2e^{-2}$)	80.00	0.85
MPAC	z-score ($z > 4.10$)	98.00	0.94
StealthInk	p-value ($p < 8e^{-4}$)	60.00	0.70
XMARK	Standard Deviation ($\sigma > 3.30$)	94.00	0.92

Table 10: Performance comparison of MPAC and XMARK on Qwen2.5-7B and LLaMA-3.1-8B models with message lengths $b \in \{8, 16\}$ and number of available tokens $T \in \{150, 200, 250, 300\}$.

Message Length	Model	Method	$T = 150$		$T = 200$		$T = 250$		$T = 300$		Avg.	Avg.
			BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓	BA↑	PPL↓
$b = 8$	Qwen2.5-7B	MPAC	96.50	8.25	98.00	7.69	98.50	7.57	98.50	7.66	97.88	7.79
		XMARK	97.75	7.66	100.00	7.19	100.00	7.07	100.00	7.00	99.44	7.23
	LLaMA-3.1-8B	MPAC	97.25	6.55	98.25	6.44	98.75	6.38	99.50	6.33	98.44	6.43
		XMARK	100.00	6.37	99.75	6.05	100.00	5.98	100.00	5.94	99.94	6.09
$b = 16$	Qwen2.5-7B	MPAC	91.12	7.97	91.00	7.90	94.75	7.54	95.38	7.55	93.06	7.74
		XMARK	96.38	7.62	97.12	7.07	97.75	6.92	99.25	6.76	97.63	7.09
	LLaMA-3.1-8B	MPAC	91.00	7.03	92.25	6.53	95.00	6.69	96.62	6.39	93.72	6.66
		XMARK	95.50	6.26	98.00	6.07	97.88	5.93	98.62	6.01	97.50	6.07

LLaMA-3.1-8B (Grattafiori et al., 2024). The BA and PPL results with message lengths $b \in \{8, 16\}$ and $T \in \{150, 200, 250, 300\}$ are reported in Table 10. For PPL evaluation, we employ the larger models Qwen2.5-32B and LLaMA-3.1-70B to obtain more accurate text quality measurements. As shown, XMARK consistently achieves higher BA and lower PPL than MPAC across all configurations, confirming its strong generalization ability across different LLM architectures. Specifically, when $b = 8$, XMARK attains nearly perfect BA while maintaining average PPLs of 7.23 and 6.09 on Qwen2.5-7B and LLaMA-3.1-8B, respectively. When b increases to 16, XMARK still delivers very high BA (97.63% and 97.50%), outperforming MPAC by +4.57% and +3.78%, respectively. Moreover, XMARK achieves lower PPLs of 7.09 and 6.07, corresponding to improvements of +0.65 and +0.59 compared with MPAC.

A.8 Generated Text Example

Table 11 presents examples of generated texts produced by XMARK for a given prompt, with watermarking bias $\delta = 2$ and message length $b = 16$.

A.9 Proof of the Expected Green List Size of XMARK

Setup and notation. Let $\mathcal{V}$ denote the vocabulary, and fix integers $d \geq 1$ and $k \geq 1$, representing the length of the message block and the number of hash keys, respectively. For each hash key, we independently (i) generate a uniformly random permutation of $\mathcal{V}$, and (ii) partition the permuted vocabulary into 2^d disjoint shards of (nearly) equal size[2]. For a given key $j \in \{0, \ldots, k-1\}$, we define the *green list* $\mathcal{G}_j$ as the complement of one designated shard. Consequently, a uniformly chosen token belongs to $\mathcal{G}_j$ with probability $1 - 2^{-d}$. The *evergreen list* is then defined as the intersection

$$\mathcal{E} = \bigcap_{j=0}^{k-1} \mathcal{G}_j.$$

In the following, we prove that the expected size of $\mathcal{E}$ is $(1 - 2^{-d})^k |\mathcal{V}|$.

Proof. For each token $v \in \mathcal{V}$, define the indicator variable

$$X_v = \mathbf{1}\{v \in \mathcal{E}\} = \prod_{j=0}^{k-1} \mathbf{1}\{v \in \mathcal{G}_j\}.$$

By construction and independence across keys, we have $\Pr[v \in \mathcal{G}_j] = 1 - 2^{-d}$, and $\{\mathbf{1}\{v \in \mathcal{G}_j\}\}_{j=0}^{k-1}$ are independent. Therefore, we have

$$\mathbb{E}[X_v] = \Pr[v \in \bigcap_{j=0}^{k-1} \mathcal{G}_j]$$
$$= \prod_{j=0}^{k-1} \Pr[v \in \mathcal{G}_j] = (1 - 2^{-d})^k.$$

[2]We assume that $|\mathcal{V}|$ is divisible by 2^d; otherwise, there is an $O(1/|\mathcal{V}|)$ discrepancy due to the near-equal partition.

Table 11: Example of generated texts of XMARK.

Prompt: *MIDDLETOWN, NJ - The Middletown Township Public Library and the Township of Middletown are pleased to announce the appointment of Kim Rinaldi as Provisional Library Director, beginning on February 24, 2014. [...] choice for the residents of the Township.*

Method	Generated Text	PPL↓
Nonwatermarked	The Middletown Board of Trustees interviewed three highly qualified candidates for the Library Director position. The Board of Trustees voted to appoint Ms. Rialdi to the position of Provisional Library director at a meeting held on February 10, 2004. The MTPL is a member of the Monmouth County Library	3.17
XMARK	The provisional library director contract will be ratified by the Middletown Township Public Library Board of Trustees on Tuesday, February 25, 2024. Ms Rinaldi will begin on Monday, February 24th. Previous MTPL Receives 2013	3.92

Prompt: *Belying expectations, Prasar Bharti has earned only Rs 58.19 crore (Rs 581.9 million) as revenue during the Commonwealth Games [...] Rs 90,000 for the same in the opening of the CWG.*

Method	Generated Text	PPL↓
Nonwatermarked	The advertisements were supposed to be telecasted live and the revenue was supposed to be earned in real time. But after the complaints, the channel was not allowed to telecast advertisements during closing ceremony. The opening ceremony was telecasted live on DD and was deferred live for the evening slot to accommodate advertising.	4.61
XMARK	But the channel had to forego the rate hike as advertisers did not want to be associated with a sports event, sources said. Prasar Bharati's revenue from the last CWG Games was Rs 59.19 cr (Rs 403.8 million), a sharp drop from Rs	5.49

Since $|\mathcal{E}| = \sum_{v\in\mathcal{V}} X_v$, by linearity of expectation, we have

$$\mathbb{E}[|\mathcal{E}|] = \sum_{v\in\mathcal{V}} \mathbb{E}[X_v] = (1-2^{-d})^k|\mathcal{V}|.$$

Dividing both sides by $|\mathcal{V}|$ gives

$$\mathbb{E}[\gamma] = \mathbb{E}\left[\frac{|\mathcal{E}|}{|\mathcal{V}|}\right] = (1-2^{-d})^k,$$

which concludes the proof. □