ADAPTIVE PAYLOAD ALLOCATION FOR MULTI-BIT LLM TEXT WATERMARKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Watermarking has emerged as a promising technique for tracing the provenance of content generated by large language models (LLMs). Compared to single-bit watermarking, multi-bit watermarking could embed richer provenance information, such as user identifiers. Most existing multi-bit methods adopt a fixed-payload design, embedding the same number of message bits at every generation step. However, we argue that the payload should be adaptively selected to maximize watermark detectability. In this paper, we characterize detectability using the mutual information since it quantifies the information about the embedded message after observing the sampled tokens. We further show that the achievable mutual information varies with payload size, which motivates formulating per-step payload selection as a mutual-information optimization problem. Based on this, we introduce MAXIMIZED MUTUAL-INFORMATION-BASED PAYLOAD (MMIP), an oracle allocator that exhaustively evaluates candidate payload sizes and selects the one achieving the largest mutual information. However, MMIP requires constructing multiple message-conditioned watermarked distributions for each candidate payload, resulting in substantial inference overhead. To enable efficient online allocation, we further propose ENTROPY-BASED WATERMARK PAYLOAD (EWP). Inspired by findings from MMIP, EWP derives the payload directly from the entropy of the original next-token distribution, thereby avoiding both payload enumeration and explicit construction of candidate watermarked distributions. MMIP and EWP are compatible with representative logit-based and distortion-free multi-bit watermarking methods. Experiments show that EWP achieves comparable performance with MMIP with negligible additional latency and both methods consistently improves the message-recovery–generation-quality trade-off over fixed-payload baselines.

1 INTRODUCTION

Large language models (LLMs) are now widely used to generate natural language content, code, and other forms of text. As model-generated outputs become increasingly difficult to distinguish from human-written content, reliable provenance tracing has become important for mitigating risks such as misinformation, phishing, impersonation, and other forms of misuse. This concern is increasingly reflected in regulation: the EU AI Act introduces transparency obligations for certain AI-generated content Union (2024), while China has introduced labeling requirements for AI-generated synthetic content and explicitly encourages digital watermarks as a form of implicit identificationCyberspace Administration of China and others (2025). LLM watermarking provides a lightweight approach to provenance tracing by embedding hidden statistical signals directly into generated text during generation. In practice, major LLM providers have begun deploying text watermarking in production systems: Google has deployed SynthID-Text in Gemini (Dathathri et al., 2024), while Anthropic has announced watermarking for future Claude models (Anthropic, 2026).

Existing watermarking methods can be broadly divided into single-bit and multi-bit schemes. While single-bit watermarking primarily determines whether a text is watermarked, multi-bit watermarking embeds explicit messages that can encode richer provenance information such as user identifiers or timestamps (Yoo et al., 2024; Jiang et al., 2025; Xu et al., 2026). For many practical provenance applications, binary detection alone is insufficient for source attribution, misuse tracing, and accountability (Jiang et al., 2025; Xu et al., 2026). Multi-bit watermarking therefore extends water-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

marking from binary content identification to fine-grained provenance tracing. A central objective of multi-bit watermarking is *message detectability*, which is the ability of the detector to correctly recover the embedded message from a watermarked text sequence Xu et al. (2026).

To embed a multi-bit message M, existing methods commonly divide it into equal-sized message blocks and embed one block at each generation step (Yoo et al., 2024; Fernandez et al., 2023; Xu et al., 2026; Jiang et al., 2025; Qu et al., 2025). The final message is recovered by aggregating the information provided by multiple sampled tokens about their corresponding message blocks. Consequently, the amount of message information embedded by each token contributes to message detectability.

However, we argue that payload should be selected adaptively for embedding the most amount of information in every generation step.

To formalize this, we view watermark detection at each step as an inference problem. At message decoding step t, the detector observes a sampled token x_t and attempts to infer the embedded message block. Since the particular token x_t is unknown during watermark embedding, we characterize the expected information about the embedded message block after observing the sampled token for this inference using the mutual information $I_t(k)$. We show that the achievable mutual information depends on the current next-token distribution, the watermarking algorithm, and the payload size. Therefore, the payload that provides the most information may vary across generation steps.

This raises the following question: *which payload should be selected to maximize the message information embedded at each generation step?*

To answer this question, we formulate per-step payload selection as a mutual-information optimization problem. Based on this formulation, we introduce MAXIMIZED MUTUAL-INFORMATION-BASED PAYLOAD (MMIP), an oracle allocator that exhaustively evaluates candidate payload sizes and selects the payload with the largest mutual information.

Although principled, MMIP is expensive for online inference because evaluating a k-bit payload generally requires constructing 2^k message-conditioned watermarked distributions to compute mutual information. To enable efficient online payload allocation, we investigate whether the payload selected by MMIP can be predicted directly from properties of the original next-token distribution. Our empirical analysis reveals a clear relationship between the entropy of the original distribution and the payload selected by MMIP: higher-entropy distributions tend to receive larger payloads.

Motivated by these observations, we propose ENTROPY-BASED WATERMARK PAYLOAD (EWP), which directly determines the payload without explicitly evaluating mutual information. As a result, EWP avoids both payload enumeration and repeated construction of message-conditioned watermarked distributions, making it suitable for inference-time use.

Our contributions are summarized as follows:

- We propose to formulate per-step payload selection in multi-bit LLM watermarking as a mutual-information optimization problem.
- We introduce MAXIMIZED MUTUAL-INFORMATION-BASED PAYLOAD (MMIP), an oracle allocator that directly solves the proposed optimization problem through exhaustive payload evaluation.
- We propose ENTROPY-BASED WATERMARK PAYLOAD (EWP), a lightweight entropy-based approximation to MMIP with negligible online overhead.
- Experiments show that EWP achieves performance comparable to MMIP while substantially reducing inference overhead. And both methods consistently improves the trade-off between watermark detectability and generation quality over fixed-payload baselines.

2 RELATED WORKS

Recent large language model families, including Qwen2.5 Qwen et al. (2025), Gemma 2 Team et al. (2024), and DeepSeek-V3 DeepSeek-AI et al. (2025), have continued to advance the capabilities and efficiency of LLMs. Alongside these model advances, LLMs have been increasingly applied to and evaluated on demanding tasks such as mathematical reasoning Cobbe et al. (2021a), instruc-

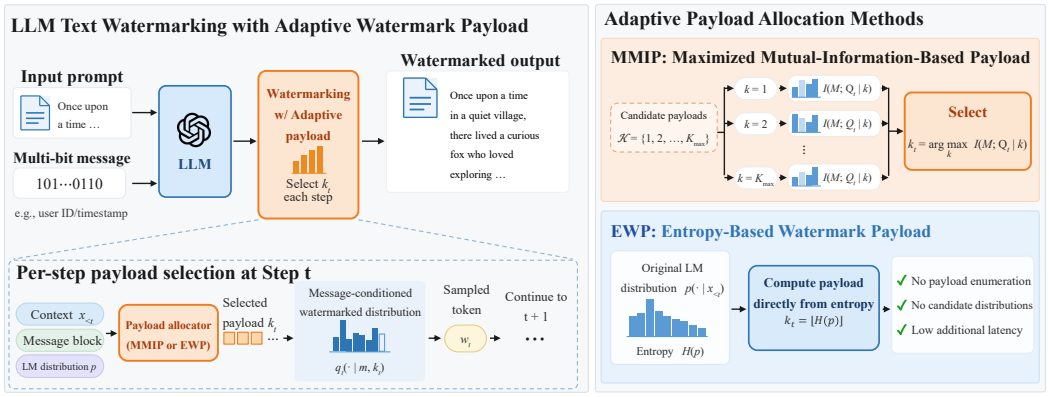


Figure 1: Framework of Our method.

tion following Zhou et al. (2023), and long-context understanding Bai et al. (2024). Despite these advances, the widespread deployment of LLMs has also raised growing concerns regarding content provenance, malicious use, and intellectual property protection Sun et al. (2023). LLM watermarking has therefore emerged as a promising approach for identifying and attributing machine-generated content.

Existing LLM watermarking methods can generally be categorized into zero-bit, also commonly referred to as single-bit, watermarking Lu et al. (2024); Kirchenbauer et al. (2024); Lee et al. (2024) and multi-bit watermarking Li et al. (2024); Fernandez et al. (2023); Yoo et al. (2024); Qu et al. (2025); Jiang et al. (2025); Xu et al. (2026), according to the amount of information embedded in the generated text. Zero-bit watermarking encodes only the presence or absence of a watermark, allowing a detector to determine whether a text contains a designated watermark. In contrast, multi-bit watermarking embeds an explicit message into the generated text, enabling richer metadata, such as a user identifier, model identifier, or generation-session identifier, to be associated with each output Jiang et al. (2025). Consequently, multi-bit watermarking supports more fine-grained attribution and tracing than zero-bit watermarking.

3 Preliminaries

3.1 LLM Text Generation

Let $\mathcal{V}$ denote the vocabulary of a LLM. Given a prompt $x_{<t} = (x_1, \ldots, x_{t-1})$, an autoregressive LLM defines a next-token distribution

$$p_t(v) = \Pr(x_t = v \mid x_{<t}), \quad v \in \mathcal{V}. \tag{1}$$

The next token is generated by sampling from this distribution:

$$x_t \sim p_t(\cdot). \tag{2}$$

After x_t is sampled, it is appended to the prompt, and the model proceeds to the next generation step. Repeating this process produces a sequence

$$x_{1:T} = (x_1, \ldots, x_T). \tag{3}$$

3.2 Multi-bit LLM Watermarking: Message Embedding

Multi-bit watermarking embeds a message

$$\mathbf{M} = (m_1, \ldots, m_L) \in \{0, 1\}^L \tag{4}$$

into the LLM generated text.

A typical fixed-payload multi-bit watermarking method embeds a message block

$$M_t^k \in \{0,1\}^k \tag{5}$$

at generation step t, where k is the fixed payload size. Each block M_t^k represents one of 2^k possible payload values. The block is selected from the message $\mathbf{M}$ according to a deterministic message scheduling rule. We denote by

$$\mathcal{A}_t = (a_{t,1}, \ldots, a_{t,k}) \tag{6}$$

which is associated with the k bits embedded at step t, where $a_{t,j} \in \{1, \ldots, L\}$.

Watermark embedding can be viewed as a transformation of the next-token distribution. Formally, we write the embedding process as

$$q_t(\cdot \mid M_t^k) = \mathcal{W}_{\mathsf{sk}}\left(p_t, x_{<t}, M_t^k\right), \tag{7}$$

where $\mathcal{W}_{\mathsf{sk}}$ is the watermarking algorithm with secret key, and $q_t(\cdot \mid M_t^k)$ is the message block-conditioned watermarked next-token distribution. The generated token is then sampled from the watermarked distribution:

$$x_t \sim q_t(\cdot \mid M_t^k). \tag{8}$$

Different watermarking algorithms instantiate $\mathcal{W}_{\mathsf{sk}}$ in different ways. For example, logit-based methods construct q_t by increasing the logits of tokens associated with the target message block Yoo et al. (2024); Xu et al. (2026); Qu et al. (2025); Fernandez et al. (2023), while distortion-free methods may use secret-keyed sampling rules to encode the payload without changing the marginal next-token distribution Jiang et al. (2025).

3.3 Multi-bit LLM Watermarking: Message Decoding

Given a generated sequence $x_{1:T}$ and the secret key sk, the detector aims to recover the embedded message from the observed tokens. At each message decoding step t, the detector reconstructs the original next-token distribution p_t from the observed prefix $x_{<t}$ and evaluates candidate message blocks

$$M_t^k \in \{0,1\}^k. \tag{9}$$

In general, the detector uses a method-specific score function

$$S_t(x_t, M_t^k; \mathsf{sk}) \tag{10}$$

to measure how strongly the observed token x_t supports the candidate message block M_t^k. The decoded message block is obtained by

$$\hat{M}_t^k = \arg\max_{M \in \{0,1\}^k} S_t(x_t, M_t^k; \mathsf{sk}). \tag{11}$$

The detectability of a multi-bit watermark is commonly evaluated by bit accuracy, which measures the fraction of message bits correctly recovered:

$$\mathrm{BitAcc} = \frac{1}{L}\sum_{\ell=1}^{L} \mathbf{1}\left[\hat{m}_\ell = m_\ell\right]. \tag{12}$$

where $\hat{m}_\ell$ is obtained by major voting on m_ℓ. A stronger multi-bit watermark should achieve higher bit accuracy while preserving the quality and fluency of the generated text.

4 Method

4.1 Mutual Information for Multi-bit Watermarking

We quantify the detectability of a multi-bit watermark by how much information an observed token sequence provides about the embedded message. Specifically, at message decoding step t, let M_t^k denote the unknown embedded k-bit message block, X_t^k the random token to be sampled during

generation, and x_t a particular sampled token observed by the detector. The information it provides about M_t^k is

$$I\big(M_t^k; X_t^k = x_t\big) = D_{\mathrm{KL}}\left(P(M_t^k \mid X_t^k = x_t) \,\|\, P(M_t^k)\right). \tag{13}$$

This quantity measures how much observing x_t changes our knowledge of the embedded message: a larger value indicates that x_t provides more information for recovering M_t^k.

Therefore, we propose that maximizing detectability equals to maximize $I\big(M_t^k; X_t^k = x_t\big)$ at every message decoding step. However, during watermark embedding, the actual sampled token x_t is unknown. Therefore, we cannot directly optimize $I(M_t^k; X_t^k = x_t)$. A natural alternative is to maximize its expectation over all possible sampled tokens:

$$I_t(k) = \mathbb{E}_{x_t \sim P(X_t^k)}\left[I\big(M_t^k; X_t^k = x_t\big)\right]. \tag{14}$$

This expected information is exactly the mutual information between the embedded message and the sampled token:

$$I_t(k) = I\big(M_t^k; X_t^k\big). \tag{15}$$

We therefore use $I_t(k)$ as the objective for optimizing multi-bit watermark detectability.

4.2 MMIP: Maximized Mutual-Information-Based Payload

For uniformly distributed k-bit messages, the mutual information $I_t(k)$ can be written as

$$I_t(k) = \frac{1}{2^k} \sum_{m \in \{0,1\}^k} D_{\mathrm{KL}}\left(q_t^k(\cdot \mid m) \| \bar{q}_t^k\right), \tag{16}$$

where

$$\bar{q}_t^k = \frac{1}{2^k} \sum_m q_t^k(\cdot \mid m). \tag{17}$$

The whole derivation is in Appendix A.1.

We thus propose **Maximized Mutual-Information-Based Payload (MMIP)**, which dynamically selects

$$k_t^* = \arg\max_{k \le k_{\max}} I_t(k). \tag{18}$$

where $k_{\max}$ is a hyper-parameter and is set to L as default. The complete MMIP procedure is discussed in Appendix A.2.

4.3 Inference Cost of MMIP

MMIP requires constructing 2^k message-conditioned distributions to calculate mutual information for each candidate payload k. Enumerating all payloads up to $k_{\max}$ therefore requires $2^{k_{\max}+1} - 2$ distribution evaluations at each generation step.

Moreover, MMIP is watermark-dependent because $q_t^k(\cdot \mid m)$ must be constructed using the underlying watermarking algorithm. Such repeated per-step evaluation introduces substantial additional inference overhead, which can limit the practical deployment of MMIP in online text generation.

4.4 EWP: Entropy-Based Watermark Payload

Although MMIP provides a principled way to select the payload, its exhaustive evaluation of candidate payloads introduces substantial inference overhead. We therefore investigate whether the payload selected by MMIP can be predicted directly without calculating mutual information.

As shown in Figure 2, we observe a clear relationship between the entropy of the original distribution $H(p_t)$ and the payload selected by MMIP. Low-entropy distributions are predominantly assigned small payloads, whereas higher-entropy distributions tend to receive larger payloads. These observations suggest that the original entropy $H(p_t)$ provides a lightweight proxy for payload selected by MMIP.

Motivated by this finding, we propose **Entropy-Based Watermark Payload (EWP)**, which determines the payload directly from $H(p_t)$. As a result, we select the largest integer payload that does not exceed $H(p_t)$:

$$k_t^{\mathrm{EWP}} = \min\left\{k_{\max}, \lfloor H(p_t) \rfloor\right\}. \tag{19}$$

Unlike MMIP, EWP requires neither payload enumeration nor construction of message-conditioned watermarked distributions. The complete EWP procedure is provided in Algorithm 2 in Appendix A.2.

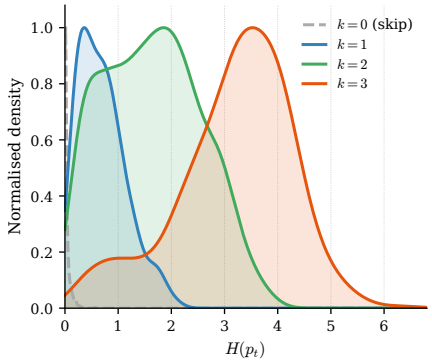


Figure 2: Per-token $H(p_t)$ vs. normalized probability density of k on CNN/DailyMail with Llama-2-7b-hf (δ=3.0). Colour indicates payload k calculated by MMIP. Higher-entropy tokens receive larger payloads, confirming entropy as an effective payload proxy.

5 Experiments

5.1 Experimental Setup

Datasets and models. We evaluate on six settings spanning text, code, and mathematical reasoning: (i) C4 (Raffel et al., 2023) with Llama-2-7B (Touvron et al., 2023), (ii) C4 with Qwen2.5-7B (Qwen et al., 2025), (iii) CNN/DailyMail (Hermann et al., 2015) with Llama-2-7B, (iv) CNN/DailyMail with Qwen2.5-7B, (v) HumanEval (Austin et al., 2021) with Qwen2.5-Coder-7B (Hui et al., 2024), and (vi) GSM8K (Cobbe et al., 2021b) with Qwen2.5-Math-7B (Yang et al., 2024).

Users and evaluation protocol. Multi-bit watermarking embeds a distinct binary message $\mathbf{m} \in \{0,1\}^L$ for each *user* (i.e. identity to be traced), so that the watermark detector can later recover $\mathbf{m}$ and identify which user produced a given text. Following prior work (Yoo et al., 2024; Xu et al., 2026), we simulate N users by pre-generating N distinct 8-bit $\mathbf{m}$ and measuring the average bit accuracy (BA) of message recovery across all users. For text settings we generate one 200-token response per prompt with $N = 100$ users. For code (HumanEval) and math (GSM8K) we set $N = 164$ (one user per benchmark problem) and additionally evaluate functional correctness (pass@1) and answer accuracy to measure watermark impact on task performance.

Methods. We compare three proposed EWP variants against seven baselines. **EWP-L** couples EWP with the logit-boosting backbone of MPAC (Yoo et al., 2024). **EWP-DF** couples EWP with the distortion-free backbone of StealthInk (Jiang et al., 2025). **MMIP-L** replaces EWP's entropy rule with a mutual-information (MI) criterion for k-selection, combined with the same logit-boosting backbone as EWP-L. **MMIP-DF** pairs the same MI-based k-selection with the distortion-free StealthInk backbone. Baselines include **XMark** (Xu et al., 2026), **MPAC** (Yoo et al., 2024), **LOSO** (Xu et al., 2026), **RSBH** (Qu et al., 2025), **CycleShift** (Fernandez et al., 2023), **StealthInk** (Jiang et al., 2025), and **DepthW** (Li et al., 2024).

Hyperparameters. All methods embed an 8-bit message. For EWP and MMIP, we set $k_{\max} = 5$. We sweep the distortion strength $\delta \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0\}$, with generation seed 0, to trace the quality-vs-accuracy Pareto frontier. Perplexity (PPL) or task-specific quality metrics (pass@1, accuracy) serve as the quality axes; bit accuracy (BA) serves as the detection axis.

5.2 Main Results: Quality–Accuracy Pareto Frontier

Figure 3 shows the Pareto frontiers across all six settings. On text tasks (C4 and CNN/DailyMail), the y-axis is $-\log(1-\mathrm{BA})$, a monotone transform that expands the high-BA region where methods cluster. EWP-L consistently dominates or matches the best baselines: it achieves higher $-\log(1-\mathrm{BA})$ than MPAC at equal quality because the adaptive payload avoids wasting distortion budget on low-entropy tokens. EWP-DF inherits StealthInk's distortion-free property while embedding substantially more bits, yielding a frontier that extends well beyond the non-adaptive StealthInk

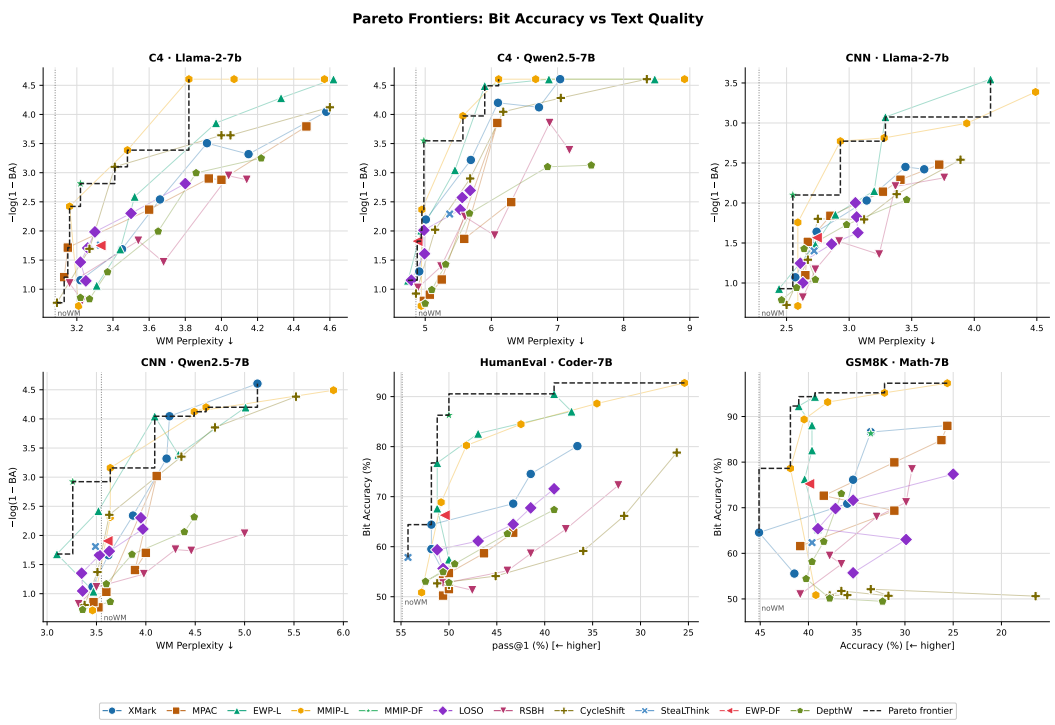


Figure 3: Quality–accuracy trade-offs across six settings. Each point corresponds to one value of δ; connected points trace each method's δ sweep. Higher BA at equal quality—or equal BA at lower PPL / higher task score—indicates a strictly superior trade-off.

baseline. MMIP-L pushes the BA ceiling further by selecting k via mutual information, achieving the highest $-\log(1-\mathrm{BA})$ among all methods in most settings. MMIP-DF, achieves high BA with near-zero PPL overhead. On code (HumanEval) and math (GSM8K), raw BA is used directly. Token entropy is low ($\sim$88% and $\sim$78% skip rates for EWP-L respectively), so all methods achieve modest BA; nevertheless EWP-L and MMIP-L improve over the fixed-k MPAC baseline by avoiding forced embedding on near-deterministic tokens, as evidenced by the smaller quality penalty at equal BA.

5.3 Robustness Evaluation

We evaluate robustness to three post-hoc text attacks on the C4/Llama-2-7B setting: random *word deletion*, random *word insertion*, and *synonym substitution*, each at perturbation rates $p \in \{0.1, 0.2, 0.3\}$. Table 1 reports the average BA (%) over 100 users.

MMIP-L and MMIP-DF together dominate most conditions, each winning roughly half.

Both MMIP methods substantially outperform EWP-L ($+5-9$ pp), confirming that the MI-based adaptive k-selection is the primary reason of robustness gains.

5.4 Speed Comparison

Table 2 reports average generation and detection times per user at $\delta = 2.0$ on the C4/Llama-2-7B setting (100 users, single A100 GPU); results for all six settings are in Appendix A.3. Hash-based detection (XMark, MPAC, LOSO, RSBH, CycleShift) scores tokens against a pseudorandom partition and does not require a GPU forward pass. Model-based detection (StealthInk, EWP-L, EWP-DF, DepthW, MMIP-L, MMIP-DF) must re-run the model on the observed prefix to recover per-step distributions.

Table 1: Robustness of bit accuracy (%) under three attacks at perturbation rates $p \in \{0.1, 0.2, 0.3\}$ (C4 / Llama-2-7B, 100 users; logit-based methods at $\delta = 1.0$). Best result per column is **bold**; our methods are marked with †.

	Word Delete			*Word Insert*			*Synonym*		
Method	p=.1	p=.2	p=.3	p=.1	p=.2	p=.3	p=.1	p=.2	p=.3
XMark (Xu et al., 2026)	72.5	72.0	65.6	75.2	73.4	67.8	75.1	70.4	68.1
MPAC (Yoo et al., 2024)	75.9	74.6	**73.5**	75.6	75.2	72.4	75.8	72.5	72.2
LOSO (Huang et al., 2025)	73.8	70.0	66.1	74.9	72.2	66.9	72.4	72.1	69.4
RSBH (Qu et al., 2025)	48.4	51.2	49.8	50.2	48.8	49.5	48.6	50.4	50.2
CycleShift (Fernandez et al., 2023)	70.8	67.4	64.9	74.5	68.4	66.0	69.1	66.5	62.5
StealthInk (Jiang et al., 2025)	72.5	68.6	62.2	71.1	67.6	62.1	73.4	66.6	66.1
EWP-L†	77.5	72.0	68.2	73.6	70.9	68.0	76.1	72.2	70.0
EWP-DF†	71.9	67.1	62.2	72.1	67.9	56.9	74.4	67.9	59.9
MMIP-L†	**82.9**	**77.6**	71.9	80.9	**79.6**	74.5	**83.9**	**80.0**	74.6
MMIP-DF†	81.4	77.2	72.5	**81.8**	78.2	**75.0**	82.5	78.8	**75.8**

Table 2: Average generation and detection time per user (seconds) at $\delta = 2.0$, C4 / Llama-2-7B, 100 users, single A100 GPU. $\star$: detection requires a GPU model forward pass. See Appendix A.3 for all six settings.

Method	Gen (s)	Det (s)	Det type
No Watermark	10.5	—	—
XMark (Xu et al., 2026)	11.7	0.2	hash
MPAC (Yoo et al., 2024)	10.6	0.3	hash
LOSO (Huang et al., 2025)	10.8	0.1	hash
RSBH (Qu et al., 2025)	10.8	0.3	hash
CycleShift (Fernandez et al., 2023)	11.2	0.2	hash
StealthInk (Jiang et al., 2025)	10.9	0.1	model*
DepthW (Li et al., 2024)	10.8	29.0	model*
EWP-L†	10.6	0.2	model*
EWP-DF†	10.7	0.2	model*
MMIP-L†	13.3	3.8	model*
MMIP-DF†	13.8	2.8	model*

Generation overhead is minimal for most methods relative to the no-watermark baseline. XMark adds $\sim$1.2 s from its reinforcement-based logit search. MMIP-L and MMIP-DF both add $\sim$2.5–2.8 s to generation because the MI criterion requires an additional model forward pass at each token to evaluate candidate k values. EWP-L detection (0.2 s) is fast because the detector runs a single batched forward pass over the full generated sequence to recover per-step entropy values, then applies hash-based decoding; no repeated per-token inference is needed. EWP-DF achieves equally fast detection (0.2 s) by re-using the StealthInk forward pass with no additional overhead. MMIP-L detection (3.8 s on C4/Llama) and MMIP-DF detection (2.8 s) are slower because both must re-evaluate the MI criterion over all candidate k values at each token position. See Appendix A.3 for results in all settings.

5.5 Token-Level k-Distribution Analysis

To understand how each method allocates payload across tokens, Table 3 reports the fraction of tokens assigned each k value at $\delta = 2.0$ across all six settings. Appendix A.4 shows the full distribution across all six δ values.

First, EWP-L's distribution is nearly *delta-independent*: because $k_t = \lfloor H_t \rfloor$ depends only on the model's entropy rather than δ, the allocation barely shifts across distortion strengths. Second, MMIP-L concentrates mass at $k = 2$ because its MI criterion rewards tokens where a 2^2-way partition yields the strongest channel signal; at $\delta = 2.0$, $k = 2$ accounts for 58% of embedded tokens on

Table 3: Token-level k-distribution (%) at $\delta = 2.0$, averaged over users per setting. $k = 0$: token skipped. Embed% $= 1 - \Pr[k = 0]$. See Appendix A.4 for results across all δ.

Setting	Method	k=0	k=1	k=2	k=3	k=4	Embed%
C4 / Llama	MMIP-L	17	21	58	3	0	83
	EWP-L	42	17	17	14	10	58
C4 / Qwen	MMIP-L	26	13	59	2	0	74
	EWP-L	33	19	19	16	13	67
CNN / Llama	MMIP-L	46	11	42	1	0	54
	EWP-L	54	14	14	10	8	46
CNN / Qwen	MMIP-L	35	17	46	2	0	65
	EWP-L	51	21	14	8	6	49
HumanEval	MMIP-L	70	13	15	1	0	30
	EWP-L	87	8	3	1	1	13
GSM8K	MMIP-L	64	15	20	1	0	36
	EWP-L	78	13	6	2	0	22

C4/Llama and 59% on C4/Qwen. We believe this is the reason for why most prior studies choose fixed payload as 2 for best detection performance (Xu et al., 2026; Jiang et al., 2025; Qu et al., 2025). Third, MMIP-L achieves a substantially higher effective embedding rate (*Embed*%$= 1 - \Pr[k = 0]$) than EWP-L in every setting: 83% vs. 58% on C4/Llama, and 30% vs. 13% on HumanEval. This explains the Pareto advantage of MMIP-L in Figure 3—it embeds bits into more tokens per sequence at the cost of a larger logit bias per step. Fourth, on code and math tasks, both methods skip most tokens (>60%) due to near-deterministic generation; MMIP-L still maintains a $\sim 2\times$ embedding-rate advantage (36% vs. 22% on GSM8K), consistent with its stronger Pareto curves on those tasks.

6 Conclusion

In this paper, we formulate adaptive payload allocation in multi-bit LLM watermarking as a mutual-information optimzation problem and introduce MMIP to directly optimize the payload at each generation step. To reduce its inference cost, we further propose EWP, which selects the payload directly from the original next-token distribution. Experiments show that EWP achieves comparable performance with MMIP with negligible overhead and both methods improve the message-detectability and generation-quality trade-off over fixed-payload baselines.

AI use statement

In this work, we used generative AI tools for writing code, running experiments, making figure and revising paper draft. We have reviewed all AI-assisted work. We checked the correctness of LLM-generated code and reviewed the LLM generated paper draft. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Anthropic. How claude's text watermark works. https://www.anthropic.com/news/claude-text-watermark, August 2026. Accessed September 2026.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.07732.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137, Bangkok, Thailand, August

2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.172. URL https://aclanthology.org/2024.acl-long.172/.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021a. URL https://arxiv.org/abs/2110.14168.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021b. URL https://arxiv.org/abs/2110.14168.

Cyberspace Administration of China and others. Measures for labeling artificial intelligence-generated synthetic content. https://www.cac.gov.cn/2025-03/14/c_1743654684782215.htm, March 2025. Effective September 1, 2025.

Sumanth Dathathri et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634:818–823, 2024. doi: 10.1038/s41586-024-08025-4. URL https://www.nature.com/articles/s41586-024-08025-4.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6. IEEE, 2023. doi: 10.1109/WIFS58808.2023.10374576. URL https://doi.org/10.1109/WIFS58808.2023.10374576.

Karl Moritz Hermann, Tomáš Kočiský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

Beining Huang, Du Su, Fei Sun, Qi Cao, Huawei Shen, and Xueqi Cheng. Low-entropy watermark detection via Bayes' rule derived detector. In Wanxiang Che, Joyce Nabende, Ekaterina

Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 14330–14344, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.739. URL https://aclanthology.org/2025.findings-acl.739/.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Ya Jiang, Chuxiong Wu, Massieh Kordi Boroujeny, Brian Mark, and Kai Zeng. StealthInk: A multi-bit and stealthy watermark for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 27685–27709. PMLR, 2025. URL https://proceedings.mlr.press/v267/jiang25j.html.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models, 2024. URL https://arxiv.org/abs/2301.10226.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. Who wrote this code? watermarking for code generation, 2024. URL https://arxiv.org/abs/2305.15060.

Liying Li, Yihan Bai, and Minhao Cheng. Where am I from? identifying origin of LLM-generated content. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 12218–12229, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.681. URL https://aclanthology.org/2024.emnlp-main.681/.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. An entropy-based text watermarking detection method, 2024. URL https://arxiv.org/abs/2403.13485.

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. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 201–220, Seattle, WA, August 2025. USENIX Association. ISBN 978-1-939133-52-6. URL https://www.usenix.org/conference/usenixsecurity25/presentation/qu-watermarking.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023. URL https://arxiv.org/abs/1910.10683.

Zhensu Sun, Xiaoning Du, Fu Song, and Li Li. Codemark: Imperceptible watermarking for code datasets against neural code completion models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE '23, pp. 1561–1572. ACM, November 2023. doi: 10.1145/3611643.3616297. URL http://dx.doi.org/10.1145/3611643.3616297.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur,

Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.

European Union. Regulation (eu) 2024/1689: Artificial intelligence act., June 2024. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32024R1689. Blog post.

Jiahao Xu, Rui Hu, Olivera Kotevska, and Zikai Zhang. XMark: Reliable multi-bit watermarking for LLM-generated texts. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14747–14763, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.672. URL https://aclanthology.org/2026.acl-long.672/.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 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)*, pp. 4031–4055, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.naacl-long.224. URL https://aclanthology.org/2024.naacl-long.224/.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A Appendix

A.1 Derivation Over Mutual Information

For uniformly distributed k-bit messages, $P(M_t^k = m) = 2^{-k}$. Starting from the definition of mutual information,

$$I_t(k) = \sum_{m,x} P(m,x) \log \frac{P(m,x)}{P(m)P(x)} \tag{20}$$

$$= \sum_{m} P(m) \sum_{x} P(x \mid m) \log \frac{P(x \mid m)}{P(x)} \tag{21}$$

$$= \frac{1}{2^k} \sum_{m \in \{0,1\}^k} D_{\mathrm{KL}} \left(q_t^k(\cdot \mid m) \| \bar{q}_t^k \right), \tag{22}$$

where

$$\bar{q}_t^k = \frac{1}{2^k} \sum_{m \in \{0,1\}^k} q_t^k(\cdot \mid m) \tag{23}$$

is the marginal distribution of the sampled token.

A.2 Payload Allocation Algorithms

This section provides the complete procedures of MMIP and EWP.

A.2.1 MMIP

The complete MMIP procedure is summarized in Algorithm 1.

It is worth to note that maximizing over the candidate set does not guarantee that the resulting mutual information is sufficiently large for reliable message embedding. For such cases, we believe it is better to skip watermarking for maintaining text quality. We therefore introduce a predefined minimum mutual-information threshold τ. If

$$I_t(\hat{k}_t) < \tau, \tag{24}$$

even the best candidate payload conveys insufficient information about the embedded message. In this case, MMIP sets

$$k_t^* = 0, \tag{25}$$

indicating that watermark embedding is skipped at the current decoding step. Otherwise,

$$k_t^* = \hat{k}_t. \tag{26}$$

Hence, MMIP jointly determines whether the current position is suitable for watermark embedding and, if so, how many bits should be embedded.

A.2.2 EWP

EWP avoids constructing message-conditioned watermarked distributions. Instead, it directly estimates the payload from the entropy of the original next-token distribution. The complete MMIP procedure is summarized in Algorithm 2.

Algorithm 1 Maximized Mutual-Information-Based Payload (MMIP)

Require: Original next-token distribution p_t; watermarking algorithm $\mathcal{W}$; candidate payload set $\mathcal{K} = \{1, \ldots, k_{\max}\}$; minimum mutual-information threshold τ; message $M = (M_0, \ldots, M_{L-1})$ of length L; secret key sk; hash function $h(\cdot)$
Ensure: Selected payload k_t^* and message block m_t^*

for $k \in \mathcal{K}$ **do**
  for $m \in \{0,1\}^k$ **do**
    $q_t^k(\cdot \mid m) \leftarrow \mathcal{W}(p_t, m, k)$
  end for
  $\bar{q}_t^k \leftarrow \dfrac{1}{2^k} \displaystyle\sum_{m \in \{0,1\}^k} q_t^k(\cdot \mid m)$
  $I_t(k) \leftarrow \dfrac{1}{2^k} \displaystyle\sum_{m \in \{0,1\}^k} D_{\mathrm{KL}}\left(q_t^k(\cdot \mid m) \| \bar{q}_t^k\right)$
end for
$\hat{k}_t \leftarrow \arg\max_{k \in \mathcal{K}} I_t(k)$
if $I_t(\hat{k}_t) < \tau$ **then**
  $k_t^* \leftarrow 0$
  $m_t^* \leftarrow \varnothing$ ▷ skip watermark embedding
else
  $k_t^* \leftarrow \hat{k}_t$
  $s_t \leftarrow h(sk) \bmod L$
  $m_t^* \leftarrow \left(M_{s_t}, M_{(s_t+1) \bmod L}, \ldots, M_{(s_t+k_t^*-1) \bmod L}\right)$
end if
return (k_t^*, m_t^*)

Algorithm 2 Entropy-Based Watermark Payload (EWP)

Require: Original next-token distribution p_t; maximum payload $k_{\max}$; message $M = (M_0, \ldots, M_{L-1})$ of length L; secret key sk; hash function $h(\cdot)$
Ensure: Selected payload k_t^{EWP} and message block m_t

$H(p_t) \leftarrow -\displaystyle\sum_{x \in \mathcal{V}} p_t(x) \log_2 p_t(x)$
$k_t^{\mathrm{EWP}} \leftarrow \min\{k_{\max}, \lfloor H(p_t) \rfloor\}$
if $k_t^{\mathrm{EWP}} = 0$ **then**
  $m_t \leftarrow \varnothing$ ▷ skip watermark embedding
else
  $s_t \leftarrow h(sk) \bmod L$
  $m_t \leftarrow \left(M_{s_t}, M_{(s_t+1) \bmod L}, \ldots, M_{(s_t+k_t^{\mathrm{EWP}}-1) \bmod L}\right)$
end if
return $(k_t^{\mathrm{EWP}}, m_t)$

A.3 Speed Comparison across All Six Settings

Table 4 extends Table 2 to all six evaluation settings. Generation times are measured at $\delta = 2.0$; detection times are independent of δ. Model-based detectors (marked $\star$) are generally faster on smaller models (Llama-2-7B) than on larger Qwen-family models, and faster on short-context tasks (text) than on long-context math reasoning (GSM8K).

A.4 k-Distribution across All δ Values

Tables 5–10 report the full token-level k-distribution for MMIP-L and EWP-L at all six δ values $(0.5, 1.0, 1.5, 2.0, 2.5, 3.0)$ across each evaluation setting. These tables substantiate two claims from Section 5.5: (i) EWP-L's distribution is nearly δ-independent, and (ii) MMIP-L shifts weight toward $k = 2$ and $k = 3$ as δ increases.

Table 4: Average generation / detection time (seconds) per user at $\delta = 2.0$, single A100 GPU. $\star$: detection requires a model forward pass. "—" denotes no detection cost (no watermark baseline).

	C4/Llama		C4/Qwen		CNN/Llama		CNN/Qwen		HumanEval		GSM8K	
Method	Gen	Det	Gen	Det	Gen	Det	Gen	Det	Gen	Det	Gen	Det
No WM	10.5	—	10.5	—	10.5	—	10.5	—	10.3	—	17.9	—
XMark	11.7	0.2	19.8	1.0	12.0	0.2	18.9	1.2	19.3	0.9	34.0	1.3
MPAC	10.6	0.3	11.5	0.6	10.6	0.2	11.4	0.7	10.9	0.5	17.8	0.8
LOSO	10.8	0.1	11.1	0.2	10.7	0.1	10.8	0.2	10.6	0.1	18.8	0.1
RSBH	10.8	0.3	12.5	1.8	10.6	0.3	11.9	2.5	11.6	1.6	19.7	2.4
CycleShift	11.2	0.2	12.4	0.5	10.5	0.1	12.4	0.6	13.0	0.4	20.9	0.8
StealthInk	10.9	0.1*	12.2	0.7*	10.8	0.2*	12.2	0.8*	11.9	0.5*	19.2	0.5*
EWP-L†	10.6	0.2*	11.3	0.6*	10.5	0.2*	11.3	0.5*	11.1	0.2*	18.8	0.3*
EWP-DF†	10.7	0.2*	11.5	0.7*	10.8	0.2*	11.7	0.5*	11.2	0.2*	17.5	0.2*
MMIP-L†	13.3	3.8*	13.7	6.6*	12.5	3.0*	13.7	6.1*	13.5	4.6*	20.9	7.4*
MMIP-DF†	13.8	2.8*	13.8	3.1*	13.0	2.5*	14.1	3.4*	13.8	5.1*	20.9	4.8*

* model-based detection. All values directly measured at $\delta = 2.0$. MMIP-L generation is slower than other methods due to per-token MI forward pass overhead.

Table 5: k-distribution (%) on C4 / Llama-2-7B across δ.

δ	Method	k=0	k=1	k=2	k=3	k=4
0.5	MMIP-L	37	40	18	4	0
	EWP-L	42	17	17	14	10
1.0	MMIP-L	34	36	27	2	0
	EWP-L	42	17	17	14	10
1.5	MMIP-L	20	28	49	2	0
	EWP-L	42	17	17	14	10
2.0	MMIP-L	17	21	58	3	0
	EWP-L	42	17	17	14	10
2.5	MMIP-L	28	11	48	13	0
	EWP-L	41	17	18	14	10
3.0	MMIP-L	29	8	33	30	0
	EWP-L	43	17	18	14	9

Table 6: k-distribution (%) on C4 / Qwen2.5-7B across δ.

δ	Method	k=0	k=1	k=2	k=3	k=4
0.5	MMIP-L	100	0	0	0	0
	EWP-L	32	20	20	16	13
1.0	MMIP-L	63	18	19	0	0
	EWP-L	31	20	20	16	12
1.5	MMIP-L	32	20	48	0	0
	EWP-L	31	19	20	16	14
2.0	MMIP-L	26	13	59	2	0
	EWP-L	33	19	19	16	13
2.5	MMIP-L	23	10	51	16	0
	EWP-L	31	20	20	16	13
3.0	MMIP-L	17	6	37	40	0
	EWP-L	34	18	20	15	14

Table 7: k-distribution (%) on CNN / Llama-2-7B across δ.

δ	Method	k=0	k=1	k=2	k=3	k=4
0.5	MMIP-L	100	0	0	0	0
	EWP-L	55	14	14	10	7
1.0	MMIP-L	76	13	11	0	0
	EWP-L	53	14	14	11	8
1.5	MMIP-L	51	15	34	0	0
	EWP-L	53	15	15	10	7
2.0	MMIP-L	46	11	42	1	0
	EWP-L	54	14	14	10	8
2.5	MMIP-L	39	8	41	12	0
	EWP-L	55	14	14	10	7
3.0	MMIP-L	35	6	32	27	0
	EWP-L	53	14	14	11	9

Table 8: k-distribution (%) on CNN / Qwen2.5-7B across δ.

δ	Method	k=0	k=1	k=2	k=3	k=4
0.5	MMIP-L	100	0	0	0	0
	EWP-L	50	20	14	9	7
1.0	MMIP-L	74	15	11	0	0
	EWP-L	55	20	13	7	5
1.5	MMIP-L	48	20	32	0	0
	EWP-L	52	20	14	8	6
2.0	MMIP-L	35	17	46	2	0
	EWP-L	51	21	14	8	6
2.5	MMIP-L	31	15	44	10	0
	EWP-L	51	20	14	8	5
3.0	MMIP-L	24	11	39	26	0
	EWP-L	47	22	16	9	6

Table 9: k-distribution (%) on HumanEval / Qwen2.5-Coder-7B across δ.

δ	Method	k=0	k=1	k=2	k=3	k=4
0.5	MMIP-L	100	0	0	0	0
	EWP-L	87	8	3	1	1
1.0	MMIP-L	95	4	1	0	0
	EWP-L	88	8	3	1	1
1.5	MMIP-L	80	11	9	0	0
	EWP-L	87	8	3	1	1
2.0	MMIP-L	70	13	15	1	0
	EWP-L	87	8	3	1	1
2.5	MMIP-L	61	13	21	5	0
	EWP-L	84	10	4	2	1
3.0	MMIP-L	51	13	26	10	0
	EWP-L	83	10	4	2	1

Table 10: k-distribution (%) on GSM8K / Qwen2.5-Math-7B across δ.

δ	Method	k=0	k=1	k=2	k=3	k=4
0.5	MMIP-L	100	0	0	0	0
	EWP-L	79	12	6	2	1
1.0	MMIP-L	90	8	2	0	0
	EWP-L	78	13	6	2	1
1.5	MMIP-L	74	14	12	0	0
	EWP-L	79	12	6	2	1
2.0	MMIP-L	64	15	20	1	0
	EWP-L	78	13	6	2	0
2.5	MMIP-L	56	14	25	5	0
	EWP-L	75	14	8	3	1
3.0	MMIP-L	53	12	25	10	0
	EWP-L	74	14	8	3	1