Every Bit, Everywhere, All at Once: A Binomial Multibit LLM Watermark

Anonymous Authors[1]

Abstract

With LLM watermarking already being deployed commercially, practical applications increasingly require *multibit* watermarks that encode more complex payloads, such as user IDs or timestamps, into the generated text. In this work, we propose a fundamentally new approach for multibit watermarking: introducing binomial encoding to directly encode every bit of the payload at every token position. We complement our approach with a *stateful encoder* that during generation dynamically redirects encoding pressure toward underencoded bits. Our evaluation against 8 baselines on up to 64-bit payloads shows that our scheme achieves superior message accuracy and robustness, with the gap to baseline methods widening in more relevant settings (i.e., large payloads and low-distortion regimes). At the same time, we challenge prior works' evaluation metrics, highlighting their lack of practical insights, and introduce *per-bit confidence scoring* as a practically relevant metric for evaluating multibit LLM watermarks.

1. Introduction

LLM watermarking has emerged as a leading approach for detecting AI-generated content, with major companies and regulators moving toward deployment (R et al., 2024). Initial watermarking research focused on *zero-bit* watermarks, which answer a single binary question: was this text generated using a given watermark? While technically sufficient for detection, emerging practical applications demand richer watermarking signals that, e.g., allow model providers to embed individual user IDs, encode timestamps, or include licensing metadata. These use cases require *multibit* watermarks that encode an m-bit payload into the model output and are able to reliably decode it from watermarked text.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

Multibit Watermarking To enable the inclusion of such multibit information, existing schemes overwhelmingly rely on *position allocation* (Yoo et al., 2024; Qu et al., 2025; Jiang et al., 2026): at each generation step, the watermark selects a single bit position from the payload and encodes only that bit using a chosen zero-bit scheme. The full message is then recovered by aggregating per-token evidence across many tokens. Yet, allocated positions can be uneven (Jiang et al., 2026), and encoding only one bit per token has been found to be sub-optimal (Gilani et al., 2026), limiting the effectiveness of such schemes in practice (see Sec. 5). At the same time, alternative, more powerful, approaches that avoid position allocation, e.g., Cycle-Shift (Fernandez et al., 2023) and ArcMark (Gilani et al., 2026), are computationally limited to short payloads. Hence, the question remains whether we can design a powerful multibit watermarking scheme that encodes longer bitstrings without having to assign individual bits to individual generation steps?

This Work: Multibit Watermarking via Binomial Encoding We answer this question affirmatively with a powerful multibit watermarking scheme that encodes large payloads without position allocation (illustrated in Figure 1). Our key insight is to replace *fully encoding individual bits in specific locations* with *partly encoding every bit across every location* via *binomial encoding*. Given an m-bit message, at each generation step, we sample m independent Bernoulli score vectors and flip each according to the corresponding bit value, yielding a *binomial score* that aggregates alignment across all m bits (Figure 1, blue shade). The watermarked distribution then favors tokens whose scores align with the full message (Figure 1, middle). Decoding reduces to a majority vote per bit, with statistical significance provided by two-sided binomial tests (Figure 1, right). We further introduce a *stateful encoder* that, during generation, tracks which bits are already well-encoded and redirects encoding pressure toward bits that are not yet correctly recovered. Our evaluation (Sec. 5) shows that our approach outperforms prior works in most practical scenarios.

Beyond the method itself, we also revisit how multibit watermarks should be evaluated under practical deployment conditions. Prior works (Yoo et al., 2024; Qu et al., 2025; Jiang et al., 2026) typically report bit accuracy and message

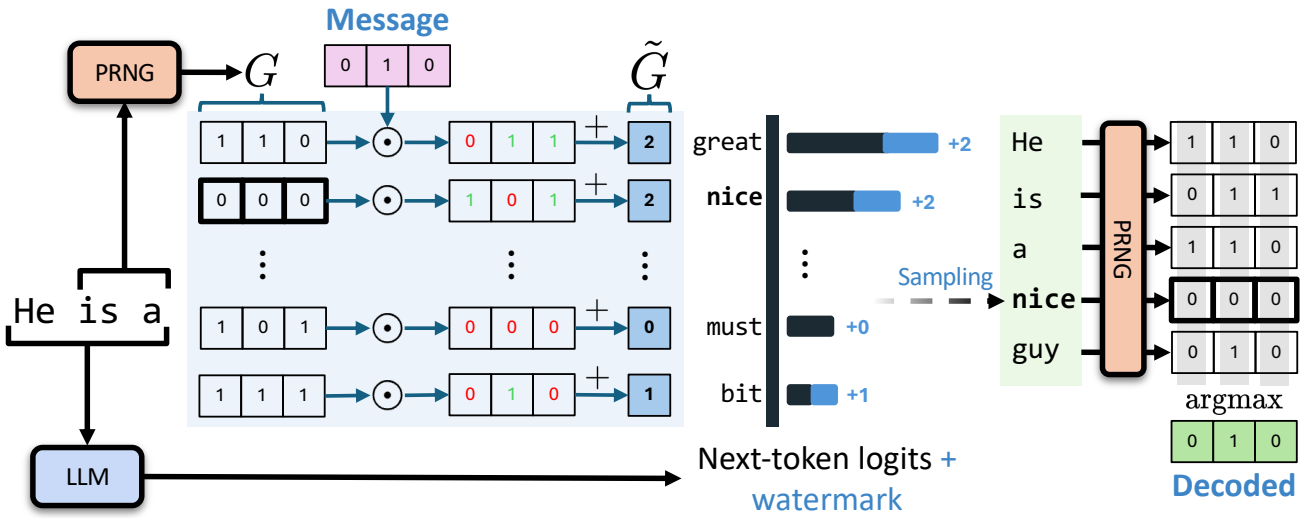

Figure 1. **Overview of Our Multibit Watermark Algorithm Encoder and Decoder:** Given an m-bit bitstring (here `010`), at each step of the generation process, we use the context to seed a pseudorandom number generator (PRNG) that samples m Bernoulli variables (G) for each token in the vocabulary. We encode our bitstring with an *XNOR* operation and sum the encoded Bernoulli variables to obtain binomial token scores ($\tilde{G}$). We then modify the next-token logit distribution with $\tilde{G}$ to obtain our watermarked distribution. For decoding the bitstring from a text, we use the same PRNG to retrieve the m Bernoulli variables for every token in the text. We then decode the bitstring by taking the majority bit at each bit position, using m binomial tests to obtain per-bit confidence.

accuracy assuming the input text is already known to be watermarked; that is, they answer "given a watermarked text, how much of the encoded message can we decode?" They do not address the broader deployment question: given an arbitrary text, can we first detect whether it is watermarked and, if so, reliably decode the embedded message? Since most texts encountered in practice are not watermarked, ignoring false positives in message recovery falls short of real-world deployment constraints. Notably, we find in Sec. 4 that while existing approaches can be post-hoc adapted with *error-detection codes*, this comes at the cost of reduced capacity. In contrast, binomial watermarks directly support *per-bit confidence scoring*, enabling even statistically sound recovery of sub-messages.

Key Contributions. Our main contributions are:

- We introduce the first multibit LLM watermark that encodes the full message bitstring at every token position via *binomial encoding*, avoiding existing shortcomings (Sec. 3.1).
- We propose a *stateful encoder* that dynamically prioritizes underperforming bits during generation, significantly improving bit accuracy without affecting decoding (Sec. 3.2).
- We challenge prior multibit evaluation metrics, highlighting their limited interpretability, and propose per-bit confidence scoring as an alternative metric (Sec. 4).
- We provide a comprehensive evaluation against 8 baselines across 3 bitstring lengths (16, 32, 64 bits), showing that our scheme achieves superior bit accuracy, message accuracy, and robustness, with the gap widening at longer payloads and in low-distortion regimes (Sec. 5).

2. Background and Related Work

In this section, we introduce relevant related work in (multibit) LLM watermarking.

LLM Watermarks The goal of (zero-bit) LLM watermarks is to insert a detectable signal into text generated by an LLM. At each step of the generation process, the previously generated context and a *private key* seed a pseudorandom number generator, which assigns a *score* to each token in the vocabulary. Then, the next-token distribution produced by the LLM is biased to favor tokens with higher scores. For detection, given a sequence of tokens (text), we aggregate the individual token scores. We call this aggregation the detection *statistic*. For instance, with Red-Green watermarks (Kirchenbauer et al., 2023), this statistic is the number of green tokens. If the text was generated without knowledge of the scores, the statistic is random. However, when using the watermark, the statistic is biased toward higher values. Importantly, for deploying detection at internet-scale, the statistic should be model-free, i.e., it should not require access to the LLM.

Popular approaches commonly build on Kirchenbauer et al. (2023), which biases the logits of the next-token distribution according to binary scores. Dathathri et al. (2024) adapts

this by proposing a tournament sampling strategy, where tokens compete against each other and the highest-scoring tokens are selected. Aaronson (2023) leverages the Gumbel-max trick to deterministically sample the next token. Lastly, Wu et al. (2023); Chen et al. (2025) reweight the next-token distribution according to the ranking of the token scores. Each watermark approach strikes a trade-off between the watermark impact on quality and its detectability. Notably, most (Dathathri et al., 2024; Aaronson, 2023; Kuditipudi et al., 2024; Wu et al., 2023; Chen et al., 2025) are distortion-free: in expectation over the private key they do not modify the LLM next-token distribution.

Multibit LLM Watermarks Multibit LLM watermarks inject a bitstring (e.g., user ID, timestamp) into the generated text that can later be decoded. Fernandez et al. (2023) (*Cycle-Shift*) proposes treating each bitstring as a private key and then using any zero-bit watermarking algorithm. For decoding, they compute the statistic with each possible key and decode to the key with the highest statistic. While effective for small messages, this method does not scale to longer ones as decoding time increases exponentially with the message length. Yoo et al. (2024) uses position allocation (*MPAC*) to encode the bitstring with any zero-bit watermarking algorithm: at each generation step, they use the previously generated context to pseudorandomly select a position within the bitstring. Given the bit value at this position, they modify the score by taking its complement (defined as $F^{-1}(1 - F(\text{score}))$, where F is the score CDF and F^{-1} its quantile function) if the bit value is 1. For decoding, they accumulate the scores and complementary scores respectively for each position and compute a (position-wise) statistic with either. The decoded bit value for a position is set to 0 if its statistic computed with the scores is higher than with the complementary scores, and 1 otherwise. Yoo et al. (2024) also suggests extending MPAC with Cycle-Shift: for each position, a sub-bitstring can be encoded using Cycle-Shift. This method also does not scale to longer bitstrings because decoding time increases exponentially with the message length.

Various prior works build on MPAC and Cycle-Shift. Qu et al. (2025) (*RSBH*) combines a balanced position allocation algorithm with Red-Green watermarks (Kirchenbauer et al., 2023) and error-correcting codes to increase robustness. Jiang et al. (2025) (*StealthInk*) combines MPAC with DiPMark (Wu et al., 2023), and Feng et al. (2025) (*BiMark*) combines MPAC with an unbiased reweighting strategy to minimize the impact of the watermark on quality. More recently, Jiang et al. (2026) (*MirrorMark*) proposes CABS, improving MPAC to better balance the selected position, and combines it with SynthID-text (Dathathri et al., 2024). Cui et al. (2026) (*MC2Mark*) expands McMark (Chen et al., 2025) to the multibit setting. Lastly, Gilani et al. (2026) (*ArcMark*) utilizes random linear channel codes and optimal transport on a unit circle to embed the watermark, ensuring that each generated token contains some information about the entire message. However, for ArcMark, similarly to Cycle-Shift, the decoding time also scales exponentially with the message length (Gilani et al., 2026).

Importantly, all approaches either only encode a sub-bitstring per token and must rely on position allocation to encode longer bitstrings or are intractable for longer messages (Fernandez et al., 2023; Gilani et al., 2026). In this work, we introduce the first multibit watermark that can effectively encode any bitstring length without position allocation, and, crucially, is more powerful than prior works, as we show in Sec. 5.

3. Our Method

Next, we introduce our multibit LLM watermarking algorithm based on *binomial encoding*. First, we explain how our approach encodes and decodes payloads into generated token sequences (Sec. 3.1). Then, we introduce an optional stateful encoder that significantly improves the bit accuracy (Sec. 3.2). Lastly, we combine all our elements into an end-to-end watermarking algorithm (Sec. 3.3).

3.1. Encoding and Decoding the Bitstring

Notation We adopt the notation from Gloaguen et al. (2026b). At each token position t, given the current context $\omega_{<t} \in \Sigma^{t-1}$, let $p_t \in \Delta(\Sigma)$ be the next-token probability distribution given by the LLM. We generate, using the context $\omega_{<t}$ and the private key, (pseudorandom) watermark scores $G_t \in \mathbb{R}^{|\Sigma|}$ (i.e., one score per candidate token), and we denote by $q(G_t, p_t) \in \Delta(\Sigma)$ the corresponding watermarked next-token probability distribution used for sampling at position t.

We note that as in (Gloaguen et al., 2026b), $q(G_t, p_t) \in \Delta(\Sigma)$ generalizes a range of watermarking schemes, such as SynthID or Red-Green watermarks. In particular, this means that our multibit en-/decoding approach can be easily adapted to use most zero-bit watermarks. For ease of illustration, both this section and Figure 1 present our method using Red-Green watermarks, whereas we provide details on the exact $q(G_t, p_t)$ in Sec. 3.3. With Red-Green watermarks, we have $G_t \in \{0, 1\}^{|\Sigma|}$ (where 0 are red tokens and 1 green tokens) and, more precisely, each entry is an i.i.d. Bernoulli (pseudo)-random variable with probability 0.5. The watermarked next-token probability distribution is

$$q(G_t, p_t) \propto p_t \exp(\delta G_t), \tag{1}$$

where $\delta > 0$ is a hyperparameter that balances the watermark detectability-quality trade-off.

Binomial Encoding Let $M \in \{0,1\}^m$ be the m-bit bitstring that we want to encode in the generated text. Unlike most prior works (Yoo et al., 2024), we propose encoding the full bitstring M at every position t. Our key insight is to encode the m bits directly into the scores G_t. To do so, at each position t we sample m independent Bernoulli(0.5) score vectors $G_t^1, \dots, G_t^m \in \{0,1\}^{|\Sigma|}$ (i.e., each vector defines a split of the entire vocabulary). The score vector G_t^i is used to encode the i-th bit of our bitstring. For each $i \in \{1, \dots, m\}$, we encode the i-th bit M_i using complementary scores,

$$\tilde{G}^i = \begin{cases} G^i & \text{if } M_i = 1, \\ 1 - G^i & \text{otherwise.} \end{cases} \tag{2}$$

At position t, we apply Equation (2) to each G_t^i and obtain complementary score vectors $\tilde{G}_t^1, \dots, \tilde{G}_t^m \in \{0,1\}^{|\Sigma|}$, allowing us to construct the final score vector $\tilde{G}_t := \sum_{i=1}^m \tilde{G}_t^i \in \{0, \dots, m\}^{|\Sigma|}$. Given a token $u \in \Sigma$, $\tilde{G}_t(u)$ indicates how many components $\tilde{G}_t^i(u)$ align with the original bitstring M (i.e., it gives the bit accuracy when decoding the bitstring from token u at position t). Sampling tokens with larger $(\tilde{G}_t)_u$, i.e., with higher scores, therefore increases the expected bit accuracy. Hence, we can compute $q(\tilde{G}_t, p_t)$, i.e., the watermarked next-token probability distribution that favors high-scoring tokens, which yields the embedding step of our watermark at position t. We illustrate this encoding process in Figure 1 (blue shade).

With the Red-Green watermark, the multibit watermarked distribution is then given by

$$q(\tilde{G}_t, p_t) \propto p_t \exp(\delta \tilde{G}_t). \tag{3}$$

This probability distribution biases sampling towards tokens with larger $\tilde{G}_t$. The only difference to Equation (1) is that, instead of red/green Bernoulli scores, we now use message-bitstring-aligned binomial scores. We illustrate this adjusted Red-Green watermark in Figure 1 (middle).

Decoding from a Text Let $\omega \in \Sigma^*$ be a sequence of tokens. Decoding aims to extract a bitstring $\hat{M}$ from ω (assuming our encoding), without relying on the original bitstring M or the LLM next-token probability distributions. For each token position t (where $\omega_t \in \Sigma$ denotes the token in the sequence), we recompute the same pseudorandom Bernoulli scores $G_t^1(\omega_t), \dots, G_t^m(\omega_t) \in \{0,1\}$ as during encoding (using $\omega_{<t}$ and the private key). By Equation (2), we know that for all i, $G_t^i(\omega_t)$ tends to align with the value of M_i, i.e., if $G_t^i(\omega_t) = 0$ then M_i is likely 0, and vice versa. Hence, the decoded message at position t is $\hat{M}^t = [G_t^1(\omega_t), \dots, G_t^m(\omega_t)] \in \{0,1\}^m$. For the final message, we treat each token in the sequence independently and aggregate the per-token evidence using majority vote,

$$\hat{M} = \text{round}\left(\frac{1}{|\omega|} \sum_{t=1}^{|\omega|} \hat{M}^t \right). \tag{4}$$

Importantly, our decoding method also provides statistical significance for each decoded bit. Under the null hypothesis (i.e., assuming the text is generated without the watermark), we can assume that for each $i \in \{1, \dots, m\}$ the sequence $(\hat{M}_i^t)_{t=1}^{|\omega|}$ behaves as i.i.d. Bernoulli(0.5) random variables. We can therefore use a two-sided Binomial test to assess whether the decoded bit $\hat{M}_i$ is significant. We explain how to leverage this property in Sec. 4, and illustrate the decoding in Figure 1 (right).

Zero-Bit Detection Our multibit watermark can also be used as a zero-bit watermark to decide whether the text is watermarked using our binomial approach. While we could directly aggregate the statistical tests from the multibit decoding algorithm, we propose a more powerful test in App. C.

This concludes our baseline multibit encoding and decoding algorithm. We provide further practical details in App. A and a full algorithmic description in App. B.

3.2. Improving the Bit Accuracy with a Stateful Encoder

Next, we increase the bit accuracy of our multibit watermark by improving the encoding scheme from Sec. 3.1, while keeping the decoding strategy fixed. Our idea is that, when sampling a token at position t, we can use our knowledge of the partially generated sequence to assess which bits are already well decoded and which are not. Then, instead of favoring all bits independently via $\tilde{G}_t$, we weight the importance of each bit, favoring bits that are not yet correctly decoded. Specifically, to quantify the importance of each bit, we answer for each bit the following question: what is the expected accuracy of the watermarking algorithm if at this step this bit is correctly encoded?

Expected (Worst-Case) Bit Accuracy Let $\omega_{\leq t} \in \Sigma^t$ be the partially generated sequence. We introduce the current unnormalized decoded message at position t,

$$\forall i \in \{1, \dots, m\}, \quad d_t^i = 2 \sum_{k=1}^{t} \tilde{G}_k^i(\omega_k) - t, \tag{5}$$

where $\tilde{G}_k^i(\omega_k) \in \{0,1\}$ is the complementary score (for bit i) of the realized token ω_k at position k. By design, $\tilde{G}_k^i(\omega_k) = 1$ if the k-th token encodes the i-th bit correctly. Because we use majority voting, the i-th bit is currently correctly decoded if and only if $d_t^i \geq 0$ (without ties).

Let $T \in \mathbb{N}$ denote the total number of generated tokens. Assuming that, after step t, the remaining tokens are generated independently of the watermark (i.e., a worst-case scenario), the expected accuracy of the i-th bit is given by $P[d_T^i \geq 0 \mid d_t^i]$. The total expected bit accuracy is therefore the sum across all bits of $P[d_T^i \geq 0 \mid d_t^i]$. Because the goal of our watermark encoding algorithm is to maximize the expected bit accuracy, instead of using $\tilde{G}$ as scores, we should directly use the expected bit accuracy with the closed-form expression from Lemma 3.1. We defer its proof to App. H.

Lemma 3.1. *Let $t \in \mathbb{N}$. Let $\omega_{\leq t} \in \Sigma^t$ be a partially generated sequence, and let d_t^i denote the signed decoding statistic for bit i, as defined in Equation (5). Assuming that future tokens are generated independently of the watermark, we have for all $i \in \{1, \ldots, m\}$ and $T > t$*

$$P[d_T^i \geq 0 \mid d_t^i] \approx \Phi\left(\frac{d_t^i}{\sqrt{T-t}}\right), \tag{6}$$

where Φ is the standard normal CDF.

In practice, T is not known in advance. We ablate this parameter in App. D.5 and find that averaging T over $\mathcal{T} := \{200, 300, 500, 1000, 2000\}$ yields a significantly higher bit accuracy compared to using $\tilde{G}$. Hence, at time t for each candidate token $u \in \Sigma$, given the current pseudorandom scores $G_t^1(u), \ldots, G_t^m(u)$ (and the corresponding complementary scores $\tilde{G}_t^i(u)$), we can use the following watermarking scores instead of $(\tilde{G}_t)_u$,

$$\tilde{G}'_t(u) := \sum_{i=1}^{m} \frac{1}{|\mathcal{T}|} \sum_{T \in \mathcal{T}} \Phi\left(\frac{1}{\sqrt{T}}(d_{t-1}^i + (2\tilde{G}_t^i(u) - 1))\right). \tag{7}$$

3.3. Our Watermark Algorithm

We now present our end-to-end watermarking algorithm using the multibit encoding and decoding strategy described in Sec. 3.1 and Sec. 3.2, and the Soft PPL scheme from Gloaguen et al. (2026b). We select the Soft PPL scheme as it provides, in its zero-bit variant, a high detectability-quality trade-off (Gloaguen et al., 2026b). In App. E, we alternatively propose using the SynthID scheme from Dathathri et al. (2024) to build a distortion-free multibit watermark based on our encoding and decoding strategy from Sec. 3.1.

At each token position t, let $\omega_{<t} \in \Sigma^{t-1}$ be the currently generated sequence. Given $\omega_{<t}$, we use our LLM to generate a next-token probability distribution $p_t \in \Delta(\Sigma)$. At the same time, using the k previously generated tokens as context (i.e., $\omega_{t-k:t-1}$), the watermark private key, and our fixed m-bit message $M \in \{0,1\}^m$, we pseudorandomly sample binary score vectors $G_t^1, \ldots, G_t^m \in \{0,1\}^{|\Sigma|}$, from which we derive the stateful score $\tilde{G}'_t(u)$ using Equation (7). We then compute the watermarked next-token distribution using the Soft PPL watermarking scheme from Gloaguen et al. (2026b): for all $u \in \Sigma$,

$$q(\tilde{G}'_t, p_t)_u = \mathbf{1}\left\{u = \arg\max_{v \in \Sigma}\left(\tilde{G}'_t(v) + \lambda \log(p_t)_v\right)\right\}, \tag{8}$$

where $\lambda > 0$ is chosen such that $\mathbb{E}_G\left[q(\tilde{G}'_t, p_t) \cdot \log p_t\right] = p_t \cdot \log p_t - \varepsilon$, and $\varepsilon \geq 0$ is a scheme parameter. Alternatively, as we show in App. F, we can directly use $\lambda > 0$ as the scheme parameter, yielding similar performance. For detection, we use the algorithm described in Sec. 3.1.

4. Considerations for Multibit Watermarks

In this section, we take a step back from watermark design and instead consider how practical multibit watermarks should be evaluated. We find that, inadvertently, most prior works evaluate the performance of multibit watermarks *assuming that the text is already known to be watermarked*.

Evaluation of Zero-Bit Watermarks The goal of zero-bit watermarks is to decide whether a given input text was generated by our (watermarked) LLM or not. Most prior works (Kirchenbauer et al., 2023; Kuditipudi et al., 2024) rely on statistical testing and report the True Positive Rate at a given False Positive Rate (TPR@α%FPR) as a metric for watermark performance. With TPR@α%FPR $:= x \in [0, 100]$, one knows that when running the watermark detection on *any text*, (i) if the text was generated by the watermarked model, the detector detects the watermark x% of the time and (ii) if the text was not generated by the watermarked model, it wrongly detects the watermark α% of the time. Yet, reported metrics in prior multibit watermark evaluations usually only measure the first aspect: they implicitly assume that the multibit decoder is always used on *watermarked text* (i.e., they always assume a message is embedded in the text).

Evaluation of Multibit Watermarks in Prior Works In particular, most prior works (Yoo et al., 2024; Qu et al., 2025; Jiang et al., 2026) report the bit accuracy (or message accuracy) of their watermarks as measured *on watermarked samples* (i.e., they sample watermarked text with an encoded message M, and on these samples compute the decoded message $\hat{M}$). These metrics answer the following question: *given a text generated by the watermarked LLM (i.e., assuming there is an embedded message), what is the probability that a given bit (respectively, the full bitstring) is correctly decoded?* Yet, given a text, prior methods would also decode a (random) message from text, even if none had been encoded. Crucially, this means that when decoding the watermark from *any text*, these metrics provide no information about whether the decoded message actually exists. Hence, we argue that, unlike prior metrics, evaluations of

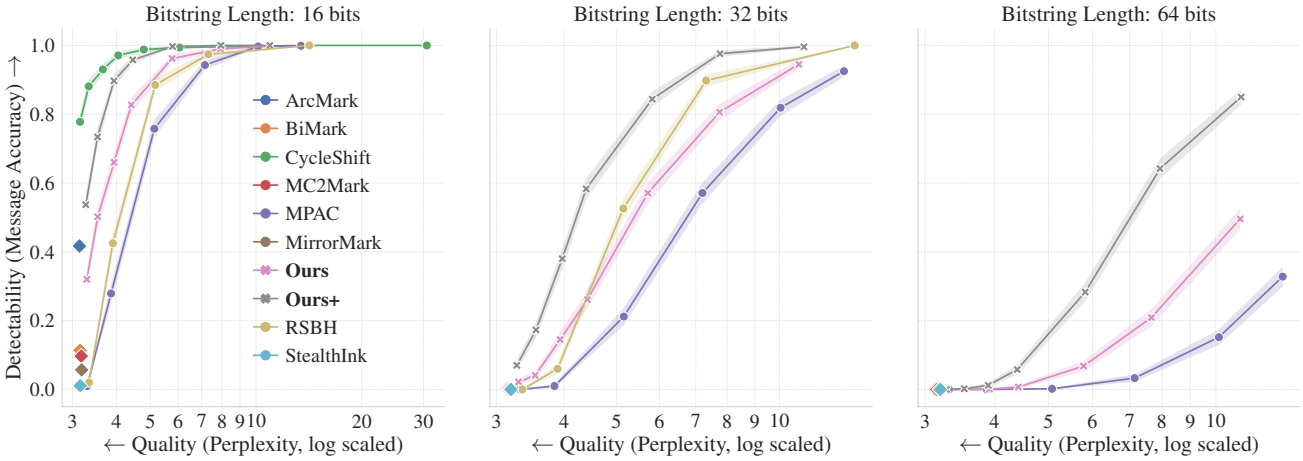

Figure 2. **Comparison of the Message Accuracy-Quality Trade-Off:** We compare the trade-off between message accuracy and text quality (log PPL) for various multibit watermarks and three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5.

multibit watermarks should measure their capability to answer two questions: (i) is there a message embedded in a given text, and (ii) if so, can it be correctly decoded?

Requirements for Multibit Watermarks More formally (and inspired by the zero-bit case), we therefore should report metrics that answer: *given any text*, what is the probability that a given bit (respectively, the full bitstring) is correctly decoded *with* $(1-\alpha)$ *confidence that the message exists?* We introduce a new metric to answer this question: bit accuracy at α% FPR (BA@α%FPR) (where the false positive corresponds, similar to the zero-bit case, to wrongly detecting a non-existing message). In Sec. 5.2, we also adapt several prior works to be evaluated as baselines on this metric.

Alternatively, using error-detection codes with the encoded message M, we can use the message accuracy metric from prior work to answer the question: *given any text, what is the probability that the decoded message exists?* If this metric is $x \in (0, 100)$, one correctly detects the existence of the message at least x% of the time (and most error-detection codes have a low probability of incorrectly detecting the existence), and correctly decodes the message x% of the time. Yet, this approach requires additional bits to encode a message of length m, it provides no information on how often a valid message is wrong, and does not necessarily provide information on which bits are incorrect. We compare the message accuracy of our approach against all baselines in Sec. 5.1.

With our scheme, we find that we can trade off both metrics. Using the baseline scores $\tilde{G}$ from Sec. 3.1, we obtain higher BA@α%FPR but lower message accuracy, while with the stateful encoder from Sec. 3.2, we have higher message accuracy but lower BA@α%FPR.

5. Evaluation

In this section, we evaluate the performance of our proposed multibit watermark, first with message accuracy (Sec. 5.1) for a fair evaluation against all baselines, and then as suggested in Sec. 4 against relevant baselines using the bit accuracy at 1% FPR (Sec. 5.2). We also measure the robustness of our watermark against text modifications (Sec. 5.3). In App. D.1 we evaluate the zero-bit detection performance of our scheme, and in App. D.2, following prior works, we also compare our scheme against all baselines using bit accuracy. Lastly, in App. D.6 we show that combining position allocation with our approach does not improve performance.

Experimental Setup Unless specified otherwise, we use LLAMA3.1-8B with temperature 0.7 and top-k set to 50. For each watermark configuration we generate replies of between 250 and 350 tokens to 1000 prompts from the ELI5 dataset (Fan et al., 2019). For the baselines, we use the recommended default parameters from prior work as detailed in App. A.2. For our schemes, we denote by *Ours* the version using $\tilde{G}$ and *Ours+* the version using $\tilde{G}'$, and discuss additional parameters in App. A.3. We sample the message bitstring uniformly at random for each completion. We measure the perplexity of the generated samples with QWEN3-30B. In App. A, we provide further hyperparameter details and in App. D.4, we evaluate our watermark on another model (MINISTRAL-3-14B).

5.1. Message Accuracy

Message Accuracy Message accuracy measures, given a watermarked text, the probability that the decoded message is correct (i.e., that no single bit is incorrectly decoded). As explained in Sec. 4, in practice, the message accuracy metric must be interpreted with error-detection codes in mind.

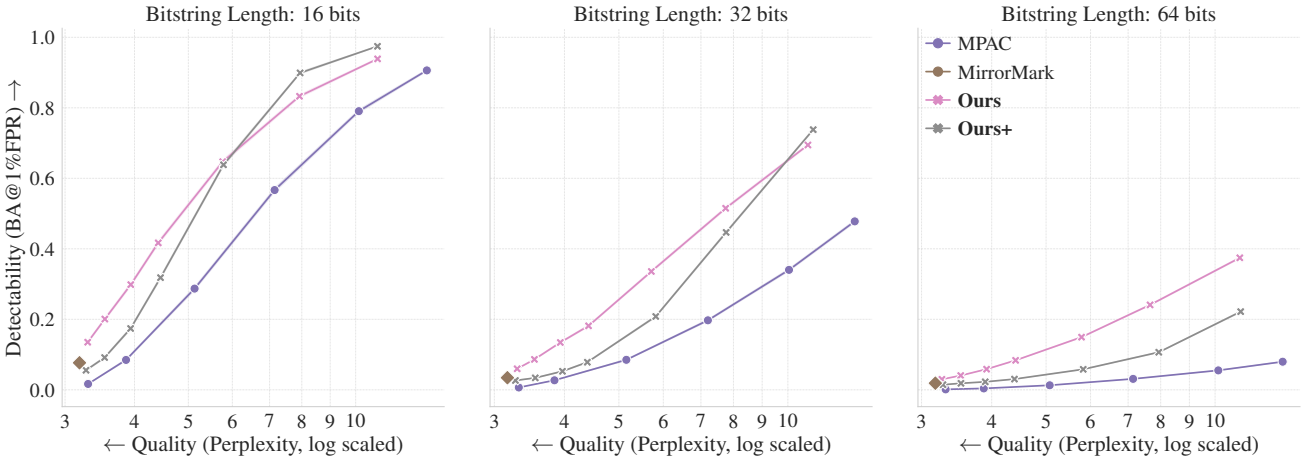

Figure 3. **Comparison of the Confidence Bit Accuracy-Quality Trade-Off:** We compare the trade-off between confidence bit accuracy (BA@1%FPR) and text quality (log PPL) across several multibit watermarks and three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5.

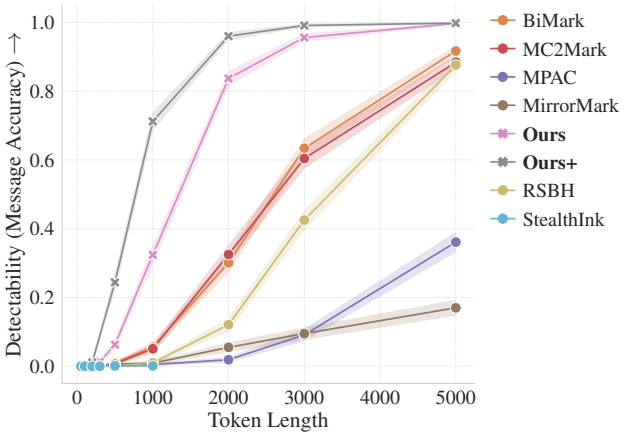

Figure 4. **Message Accuracy with Respect to the Number of Tokens:** We compare how the message accuracy scales with the number of generated tokens. The hyperparameters of each watermarking scheme have been selected to have a similar impact on quality in the low-distortion regime, and all schemes use a bit string of 32 bits.

If a given scheme reaches 100% message accuracy for a bitstring of length m, the size of the bitstring effectively encoded is m minus the number of bits used for the error-detection code. For the most widely used error-detection family, cyclic redundancy checks (Peterson & Brown, 1961), k bits are used to achieve a probability of incorrectly detecting a message of $\approx 2^{-k}$.

Improved Message Accuracy In Figure 2, we show the message accuracy–quality trade-off of different watermarking algorithms for varying bitstring lengths, using perplexity as a proxy for quality. For larger, more practical payloads (32 and 64 bits), our watermark (Ours+) outperforms prior work. In particular, we find that the stateful encoder (Sec. 3.2) significantly improves the message accuracy, as intended, compared to the stateless version (Sec. 3.1). The RSBH baseline uses an error-correction code to improve message accuracy, which explains its improved performance compared to MPAC. As mentioned in Qu et al. (2025), this approach is independent of the multibit algorithm and could also be used with our schemes (and baselines) to improve their message accuracy ad hoc. In Figure 4, we show how message accuracy scales with the number of tokens in the low-distortion regime with a 32-bit payload. We find that our scheme scales significantly more favorably with the number of tokens compared to all tested baselines. In App. D.3, we show that our watermark also outperforms prior work when using average benchmark accuracy as a proxy for quality.

5.2. Bit Accuracy at 1% FPR

Per-Bit Confidence As explained in Sec. 4, we introduce a new, more realistic metric to measure multibit watermark performance: bit accuracy at 1% FPR (BA@1%). For a given bit, the BA@1% FPR is equal to 1 if the decoded bit is correct and its p-value is below 0.01. Hence, if BA@1% FPR is $x \in (0, 100)\%$, it means that (i) if the text was generated by the watermarked model, then the detector detects the watermark $x\%$ of the time, and (ii) if the text was not generated by the watermarked model, then it wrongly decodes the bit (instead of detecting that there is no encoded bit) at most 1% of the time. To compute this metric, we need multibit watermarking schemes that provide per-bit p-values. As stated in Sec. 3.1, our method provides per-bit p-values. Additionally, we find that two baselines (MPAC and MirrorMark) can also be used to provide per-bit p-values. For both MPAC and MirrorMark, by using 1 bit per position, this is equivalent to using m separate zero-bit watermarks on each position. To get a per-bit p-value, we can therefore use the corresponding watermark two-sided statistical tests (e.g., for Red-Green watermarks a two-sided binomial test).

Table 1. **Robustness Evaluation:** Bit accuracy for different schemes with various text modifications (e.g., word deletion, synonym substitution, and paraphrasing) averaged over 1000 samples. The percentage corresponds to the percentage of words deleted/substituted. Each sample is generated with LLAMA3.1-8B using prompts from ELI5 and using a 32-bit bitstring. The highest accuracy in each row is **bolded** for readability.

	Word Deletion					Synonym Substitution					
Watermark	10%	20%	30%	40%	50%	10%	20%	30%	40%	50%	Paraphrasing
BiMark	0.664	0.603	0.564	0.539	0.520	0.666	0.616	0.580	0.556	0.536	0.504
MC2Mark	0.664	0.608	0.571	0.540	0.519	0.671	0.619	0.578	0.551	0.536	0.509
MPAC	0.609	0.572	0.544	0.529	0.524	0.612	0.575	0.555	0.532	0.524	0.505
MirrorMark	0.539	0.521	0.505	0.506	0.500	0.544	0.523	0.509	0.511	0.505	0.498
RSBH	0.511	0.508	0.501	0.500	0.502	0.509	0.506	0.503	0.499	0.501	0.501
StealthInk	0.606	0.569	0.546	0.525	0.515	0.609	0.572	0.549	0.537	0.526	0.507
Ours+	**0.707**	**0.632**	**0.579**	**0.541**	**0.525**	**0.707**	**0.642**	**0.591**	**0.562**	**0.543**	**0.509**

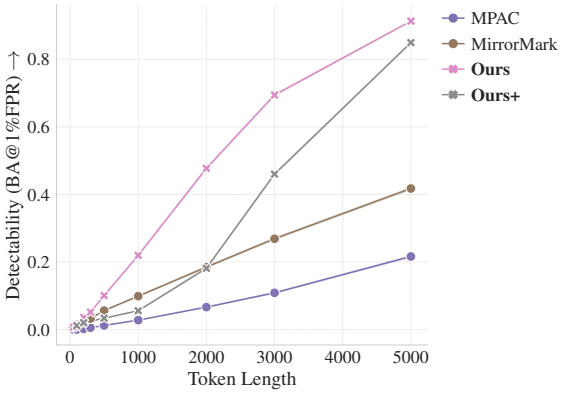

Figure 5. **Bit Accuracy at 1% FPR with Respect to the Number of Tokens:** We compare how BA@1%FPR scales with the number of generated tokens. The hyperparameters of each watermarking scheme are selected to have a similar impact on quality in the low-distortion regime, and all schemes use a bit string of 32 bits.

Results In Figure 3, we show the BA@1% FPR-quality trade-off for our watermark and the two baselines, using perplexity as a proxy for quality. We find that our stateless approach (Sec. 3.1) yields a significantly higher BA@1% FPR than the baselines. At the same time, the stateful version of our watermark (Ours+ from Sec. 3.2) underperforms the stateless version (Ours). As mentioned in Sec. 4, this outcome is expected: the objective of the stateful encoder is to maximize the expected bit accuracy independently of the confidence. Additionally, Figure 5 shows how BA@1% scales with token length, using 32-bit bitstrings in the low-distortion regime. We find that, similarly to Sec. 5.1, BA@1% for our scheme, using the stateless encoder, scales significantly faster than all baselines.

5.3. Robustness to Modifications

Experimental Setup We evaluate the robustness of our scheme (Ours+) and of the baselines in the low-distortion regime with a 32-bit bitstring. As text modifications, we consider word deletion and synonym substitution as in Pan et al. (2024), and paraphrasing using GPT-5-MINI as the paraphraser. We report the bit accuracy instead of message accuracy because for all schemes (including ours), the message accuracy after mild text modification is almost null.

Robustness Evaluation Table 1 shows the bit accuracy for different watermarks and text modifications. We find that the robustness of all tested watermarks is limited, with bit accuracy sharply dropping beyond 10% of the words edited, and becoming almost random (i.e., equal to 0.5) under paraphrasing. For message accuracy, it is already null at 10% of the words edited. These results suggest that robust multibit watermarking algorithms (in the low-distortion regime, and at 300 token length) are still an open problem. Nonetheless, for all tested text modifications, our watermark outperforms all prior baselines.

6. Conclusion and Limitations

Our work introduces the first multibit watermarking algorithms that can be applied to large payloads without relying on position allocation, opening a new direction for multibit watermarking research. Our scheme outperforms all tested baselines in most realistic scenarios. We also highlight that prior metrics used to evaluate multibit LLM watermarks fall short of practical considerations, and introduce a new metric, BA@1%FPR, for evaluating multibit watermarks.

Limitations We show that the robustness of multibit LLM watermarks is still an ongoing challenge. While our approach outperforms all evaluated baselines, it remains limited in practical scenarios. Although we empirically show that our approach, which is not based on position allocation, outperforms position allocation-based baselines, we provide no theoretical results showing that non-position allocation-based methods are strictly better.

References

Aaronson, S. Watermarking of large language models. In *Workshop on Large Language Models and Transformers, Simons Institute, UC Berkeley*, 2023.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code. 2021.

Chen, R., Wu, Y., Guo, J., and Huang, H. Improved unbiased watermark for large language models. *CoRR*, 2025.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Cui, X., Chen, R., Wu, Y., and Huang, H. Mc^{2}mark: Distortion-free multi-bit watermarking for long messages, 2026. URL https://arxiv.org/abs/2602.14030.

Dathathri, S., See, A., Ghaisas, S., Huang, P.-S., McAdam, R., Welbl, J., Bachani, V., Kaskasoli, A., Stanforth, R., Matejovicova, T., et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035): 818–823, 2024.

Fan, A., Jernite, Y., Perez, E., Grangier, D., Weston, J., and Auli, M. Eli5: Long form question answering, 2019. URL https://arxiv.org/abs/1907.09190.

Feng, X., Zhang, H., Zhang, Y., Zhang, L. Y., and Pan, S. Bimark: Unbiased multilayer watermarking for large language models, 2025. URL https://arxiv.org/abs/2506.21602.

Fernandez, P., Chaffin, A., Tit, K., Chappelier, V., and Furon, T. Three bricks to consolidate watermarks for large language models. *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, 2023.

Gao, L., Tow, J., Abbasi, B., Biderman, S., Black, S., DiPofi, A., Foster, C., Golding, L., Hsu, J., Le Noac'h, A., Li, H., McDonell, K., Muennighoff, N., Ociepa, C., Phang, J., Reynolds, L., Schoelkopf, H., Skowron, A., Sutawika, L., Tang, E., Thite, A., Wang, B., Wang, K., and Zou, A. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Gilani, A., Long, C. X., Vithana, S., Kosut, O., Sankar, L., and Calmon, F. P. Arcmark: Multi-bit llm watermark via optimal transport, 2026. URL https://arxiv.org/abs/2602.07235.

Gloaguen, T., Staab, R., Jovanović, N., and Vechev, M. Watermarking diffusion language models. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=3aBWTYGcaT.

Gloaguen, T., Staab, R., Jovanović, N., and Vechev, M. A unified framework for llm watermarks, 2026b. URL https://arxiv.org/abs/2602.06754.

Jiang, Y., Wu, C., Boroujeny, M. K., Mark, B., and Zeng, K. Stealthink: A multi-bit and stealthy watermark for large language models, 2025. URL https://arxiv.org/abs/2506.05502.

Jiang, Y., Boroujeny, M. K., Kumar, S. S., and Zeng, K. Mirrormark: A distortion-free multi-bit watermark for large language models, 2026. URL https://arxiv.org/abs/2601.22246.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. A watermark for large language models. In *International Conference on Machine Learning*, pp. 17061–17084. PMLR, 2023.

Kuditipudi, R., Thickstun, J., Hashimoto, T., and Liang, P. Robust distortion-free watermarks for language models. *TMLR*, 2024.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Pan, L., Liu, A., He, Z., Gao, Z., Zhao, X., Lu, Y., Zhou, B., Liu, S., Hu, X., Wen, L., King, I., and Yu, P. S. MarkLLM: An open-source toolkit for LLM watermarking. In Hernandez Farias, D. I., Hope, T., and Li, M. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 61–71, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-demo.7.

Peterson, W. W. and Brown, D. T. Cyclic codes for error detection. *Proceedings of the IRE*, 49(1):228–235, 1961. doi: 10.1109/JRPROC.1961.287814.

Qu, W., Zheng, W., Tao, T., Yin, D., Jiang, Y., Tian, Z., Zou, W., Jia, J., and Zhang, J. Provably robust multi-bit watermarking for ai-generated text, 2025. URL https://arxiv.org/abs/2401.16820.

R, H., I, S., D, F. L., and E, G. Generative ai transparency: Identification of machine-generated content. Scientific analysis or review, Ispra (Italy), 2024.

Wu, Y., Hu, Z., Zhang, H., and Huang, H. Dipmark: A stealthy, efficient and resilient watermark for large language models. 2023.

Yoo, K., Ahn, W., and Kwak, N. Advancing beyond identification: Multi-bit watermark for large language models, 2024. URL https://arxiv.org/abs/2308.00221.

A. Experimental Details

In this section, we describe precisely the experimental setup used in our evaluation (Sec. 5), with the exact inference parameters used (App. A.1), and the watermark configurations (App. A.2 and A.3).

A.1. Inference Setting

Models We use the instruction-tuned versions of LLAMA3.1-8B in Sec. 5 and MINISTRAL-3-14B in App. D.4, with temperature 0.7 and top-k set to 50. We use both models in bf16 precision. For inference, we use vLLM (Kwon et al., 2023) on a single NVIDIA RTX PRO 6000 Blackwell GPU.

Dataset As a prompt dataset, we use questions from ELI5 (Fan et al., 2019). For each watermark configuration, we use the same subset of prompts.

A.2. Baseline Watermark Configuration

Here, we describe the configuration used for our baseline watermarks. The notation used in this part is taken from each watermark's original paper.

ArcMark For ArcMark (Gilani et al., 2026), we use the same default configuration as in Gilani et al. (2026). An m-bit message $M \in \{0,1\}^m$ is encoded as a codeword C_M over $\mathbb{F} = \{0, \dots, p-1\}$, and we set $p = 2^m$ so that the symbol alphabet matches the total number of possible messages. At each generation step t, the shared secret is $S_t = (V_t, \Pi_t)$, where $V_t \sim \mathrm{Unif}(\{0, \dots, r-1\})$, and the channel input is formed as $z_t = \left(\frac{2\pi C_M(t)}{p} + \frac{2\pi V_t}{r}\right) \bmod 2\pi$. We set $r = 256$ for the payload sizes we consider, implement Π_t as an affine permutation over token IDs, and solve the optimal transport problem using the Sinkhorn algorithm with regularization coefficient 0.1. Because there is no official implementation for Gilani et al. (2026), we implement the ArcMark watermark algorithm from scratch.

BiMark For BiMark (Feng et al., 2025), we use the same default configuration as in Feng et al. (2025). In particular, we use the base scaling factor $\tilde{\delta} = 1.0$ and the default number of layers $d = 10$. For the implementation, we fork the code from Feng et al. (2025) and adapt it to work with vLLM.

Cycle-Shift For Cycle-Shift (Fernandez et al., 2023), we use the AAR watermark (Aaronson, 2023) as the underlying watermarking scheme, mirroring the default configuration from Fernandez et al. (2023). Additionally, to obtain a distortionary version, we follow Gloaguen et al. (2026b) and vary δ in $\{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.75, 1.0\}$. For the implementation, we fork the code from Fernandez et al. (2023) and adapt it to work with vLLM.

MC2Mark For MC2Mark (Cui et al., 2026), we use the default number of layers $m = 10$. For computational efficiency, we scale the number of segments g with the number of bits. In particular, we use $g = 2$ segments for 16 bits, $g = 4$ for 32 bits, and $g = 8$ for 64 bits. Because there is no official implementation for Cui et al. (2026), we implement the MC2Mark watermark algorithm from scratch.

MPAC For MPAC (Yoo et al., 2024), we use Red-Green watermarks (Kirchenbauer et al., 2023) as the underlying watermarking scheme, mirroring the default configuration from Yoo et al. (2024). For the green list size, we use $\gamma = 0.5$, meaning that the green list size is $\gamma|\Sigma|$. To enable faster inference, we slightly deviate from the original implementation of Kirchenbauer et al. (2023) and use independent Bernoulli variables with probability γ for the green list, where 1 means that a given token is green. In contrast, Kirchenbauer et al. (2023) enforce the green list size to be exactly $\gamma|\Sigma|$, which is equivalent to using correlated Bernoulli variables with probability γ for the green list. In terms of watermarking performance, the impact of this change is negligible (Gloaguen et al., 2026a). For the watermark strength, we vary δ in $\{1, 2, 3, 4, 5, 6\}$. For the implementation, we mostly reuse the code from Yoo et al. (2024), but adjust it to work with vLLM.

MirrorMark For MirrorMark (Jiang et al., 2026), we use the Gumbel-max variant. For CABS, we use the recommended configuration from Jiang et al. (2026): we use frame size $f = 3$, context window $W = 4$, and max factor `max_factor` $= 1.5$. Because there is no official implementation for Jiang et al. (2026), we implement the MirrorMark watermark algorithm from scratch.

RSBH For RSBH, we follow the recommended configuration from Qu et al. (2025) for the Reed–Solomon codes. In particular, for a 16-bit bitstring, we divide the original message into 4 segments and then encode each segment using Reed–Solomon codes with 6 segments and 4 bits per segment. For a 32-bit bitstring, we divide the original message into 4 segments and then encode each segment using Reed–Solomon codes with 6 segments and 8 bits per segment. For the frequency-balanced mapping, we use the realnewslike subset of the C4 dataset to estimate the frequencies. For the watermark strength, we vary δ in $\{1, 2, 3, 4, 5, 6\}$. For the implementation, we mostly reuse the code from Qu et al. (2025), but adjust it to work with vLLM.

StealthInk For StealthInk (Jiang et al., 2025), we use one bit per position. Because there is no official implementation for Jiang et al. (2025), we implement the StealthInk watermark algorithm from scratch.

A.3. Our Watermark Configuration

For our watermark, to find λ (Equation (8)), we use a bisection algorithm on the interval $(0, 100)$ with 60 iterations. To estimate $\mathbb{E}_G\left[q(\tilde{G}'_t, p_t) \cdot \log p_t\right]$, we use a Monte Carlo estimation with 128 samples. For the watermark strength, we vary ε in $\{0, 0.1, 0.2, 0.3, 0.5, 0.7, 0.9\}$.

Detection To detect the watermark, we follow the recommendation of Fernandez et al. (2023). In particular, we deduplicate the (context, token) pairs from a given text to ensure the statistical validity of our detection algorithm (see Figure 6 in App. C).

B. Algorithms

In this section, we present the algorithmic description of our encoding (Algorithm 1) and decoding algorithms (Algorithm 2) from Sec. 3.

Algorithm 1 Encoding Multibit Algorithm

Require: Prompt $\omega \in \Sigma^*$, Next-token probability $p_t \in \Delta(\Sigma)$, Message $M \in \{0,1\}^m$, Watermark private key ξ

$s_t \leftarrow \text{HASHCONTEXT}(\xi, \omega_{<t})$ ▷ Context is usually the 3 last tokens of $\omega_{<t}$
$G_t^1, \ldots, G_t^m \in \{0,1\}^{|\Sigma|} \leftarrow \text{PRNG}(s_t)$
for $u \in \Sigma$ **do**
  for $i \in \{1, \ldots, m\}$ **do**
    $\tilde{G}_t^i(u) \leftarrow M_i G_t^i(u) + (1 - M_i)(1 - G_t^i(u))$ ▷ Binomial encoding
  end for
  $\tilde{G}_t(u) \leftarrow \sum_{i=1}^m \tilde{G}_t^i(u)$
end for
$q_t \leftarrow \text{WATERMARKTRANSFORM}(p_t, \tilde{G}_t)$ ▷ e.g., the PPL watermark (Equation (8))
return q_t

Algorithm 2 Decoding Multibit Algorithm

Require: Text $\omega \in \Sigma^*$, Message length $m \in \mathbb{N}$, Watermark private key ξ

$S_1, \dots, S_m \leftarrow 0$
for $t \in \{1, \dots, |\omega|\}$ **do**
  $s_t \leftarrow \text{HASHCONTEXT}(\xi, \omega_{<t})$ ▷ Context is usually the 3 last tokens of $\omega_{<t}$
  $G_t^1(\omega_t), \dots, G_t^m(\omega_t) \leftarrow \text{PRNG}(s_t, \omega_t)$ ▷ Same PRNG as in the encoding
  for $i \in \{1, \dots, m\}$ **do**
    $S_i \leftarrow S_i + G_t^i(\omega_t)$
  end for
end for
for $i \in \{1, \dots, m\}$ **do**
  $\hat{M}_i \leftarrow \mathbf{1}\{S_i \geq |\omega|/2\}$ ▷ Majority bit decoding. Ties are broken randomly
end for
$p_{\text{value}} \leftarrow \text{BINOMIALPVALUE}(S_1, \dots, S_m; |\omega|)$
return $(\hat{M}, p_{\text{value}})$

C. Zero-Bit Watermark Detection

In this section, we detail the zero-bit watermark detection we use in our work, and for prior works. We find that some prior works have incorrectly calibrated p-values, leading to higher FPR than expected when running the watermark detection on human-generated text.

Our Zero-Bit Detector As mentioned in Sec. 3.1, our multibit watermark can also be used as a zero-bit watermark to decide whether a text ω is generated using our watermark or not. We recall that $\hat{M}_i^t$ is the i-th decoded bit at token position t. Let $S_i \coloneqq \sum_{t=1}^{|\omega|} \hat{M}_i^t$ for each $i \in \{1, \dots, m\}$. We propose computing the following statistic,

$$\Lambda(\omega) \coloneqq \Lambda(S_1, \dots, S_m) \coloneqq \sum_{i=1}^{m} \left[S_i \log\left(\frac{S_i}{|\omega|/2}\right) + (|\omega| - S_i) \log\left(\frac{|\omega| - S_i}{|\omega|/2}\right) \right]. \tag{9}$$

with the convention $0 \log 0 \coloneqq 0$. Equation (9) represents the difference in log-likelihood between the null hypothesis (all S_i are Binomial $(|\omega|, 0.5)$) and the alternative that they are not Binomial. In particular, Λ is minimized when all $S_i = |\omega|/2$, which is the most likely case under the null. Thus, to get a p-value, we use a Monte-Carlo estimation of

$$\text{p-value} \coloneqq \mathbb{E}_{S_1, \dots, S_m}[\mathbf{1}\{\Lambda(\omega) > \Lambda(S_1, \dots, S_m)\}], \tag{10}$$

where $S_1, \dots, S_m$ are sampled i.i.d. from Binomial$(|\omega|, 0.5)$.

Zero-Bit Detection Calibration To verify whether the zero-bit detection is well calibrated, we run the detection algorithm on 1000 sequences of 200 tokens extracted from human-generated text, in our case extracted from the C4 realnewslike dataset. Then, we measure the number of texts flagged as watermarked (i.e., empirical FPR) given an expected (theoretical) FPR. If the p-values returned by the detector are sound, the empirical FPR should be below the theoretical FPR (i.e., we should see a plot below the identity line). In Figure 6 (left), we show the empirical FPR given a theoretical FPR for two baselines, MPAC and StealthInk. We find that the suggested p-values are ill-calibrated.

Calibrate Zero-Bit Detectors For calibrated zero-bit p-values, we propose adjusting our Monte-Carlo approach from Sec. 3.1 to prior works' statistics. Specifically, for Yoo et al. (2024); Jiang et al. (2025; 2026) we use the same statistic as the one suggested in their respective papers, but use Monte-Carlo estimation to estimate the null distribution. For Cycle-Shift, we keep the original paper implementation as we find it is already calibrated and statistically sound. In Figure 6 (right), we show the empirical FPR given a theoretical FPR using the calibrated detector. We indeed find that, under the null, the empirical FPR is upper bounded by the theoretical FPR. This means that the zero-bit p-values are now well-calibrated.

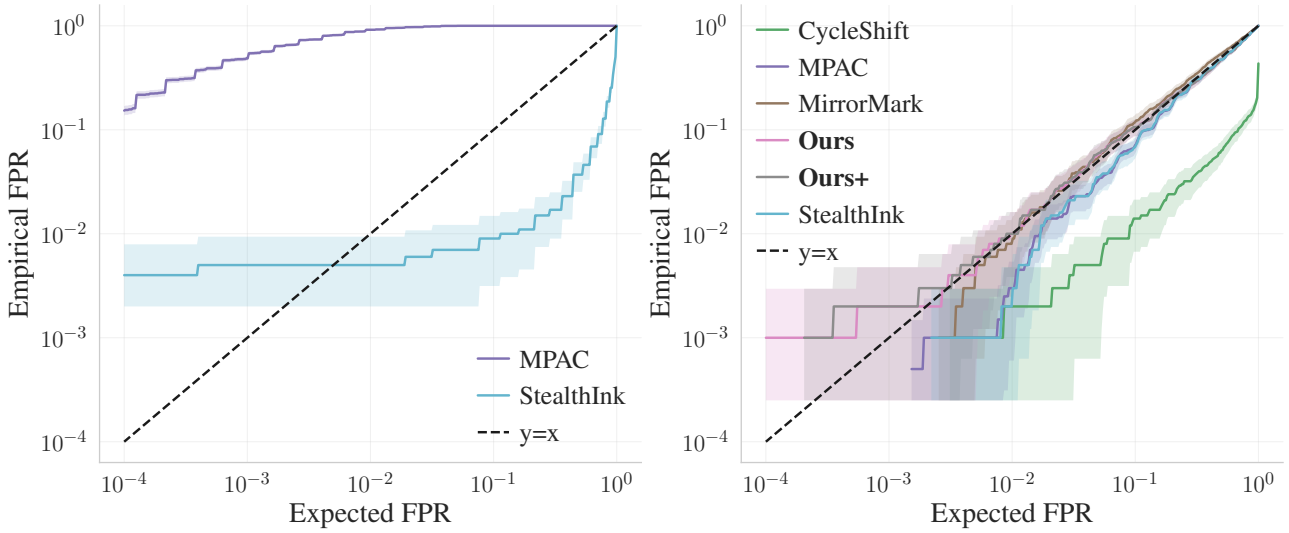

Figure 6. **Zero-Bit Watermark Detection Under the Null:** We show the empirical FPR versus the theoretical FPR for various zero-bit watermark detectors. The left side uses the original p-value implementation, whereas the right side uses Monte Carlo-based p-values (except for Cycle-Shift). To compute the p-values, we use 1000 200-token-long texts extracted from the realnewslike subset of C4. For all watermarks, we use a bitstring length of 16.

D. Additional Experiments

D.1. Zero-Bit Watermark Detection

In this part, we evaluate the zero-bit performance of our multibit watermark against relevant baselines.

Zero-Bit Detection While zero-bit detection could be encoded in the embedded bitstring (e.g., by using integrity bits), here we measure the zero-bit detection ability using the p-value provided by the watermark detector (e.g., for our watermark, using Equation (10)). For ArcMark, BiMark, MC2Mark, and RSBH, the watermark decoder does not provide p-values and we therefore exclude them from this analysis.

In particular, using the same experimental setup as in Sec. 5, in Figure 7, we measure on each generated sample the true positive rate at 1% false positive rate (TPR@1%FPR), i.e., the average number of p-values below 0.01. We find that, for all watermarking schemes, the TPR@1%FPR decreases with the payload size. This means that, for a fixed impact on quality and number of tokens, it is harder to determine (using statistical tools) whether the decoded bitstring differs from random noise. Regarding our watermark, the stateless version (Ours) (i.e., using the scores $\tilde{G}$ from Sec. 3.1) outperforms all prior schemes, and importantly significantly outperforms the stateful version (Ours+) from Sec. 3.2. This result is expected: in the stateful version, we assign higher importance to bits that are currently not correctly decoded. This means that, while all bits are more likely to be correctly decoded, none of them particularly stands out compared to random noise.

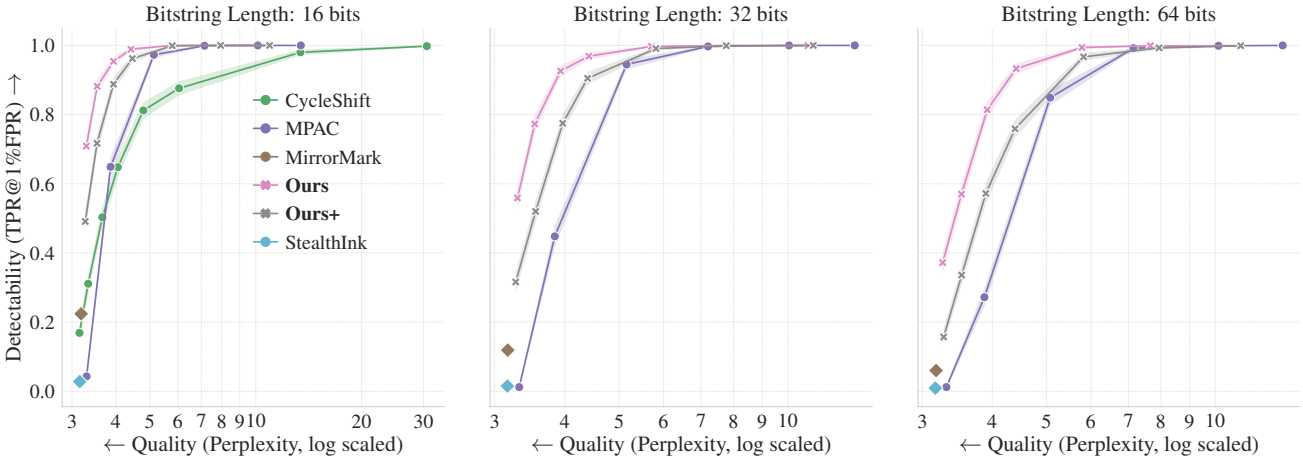

Figure 7. **Comparison of the Detectability-Quality Trade-Off:** We compare the trade-off between zero-bit detectability (TPR@1%FPR) and text quality (log PPL) across various multibit watermarks and three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5.

D.2. Bit Accuracy

In this part, following prior works, we compare our scheme's bit accuracy with all baselines. We find that, in most cases, our scheme outperforms prior works.

Improved Bit Accuracy For each completion, we record the bit accuracy, i.e., the percentage of bits that we decode correctly. A bit accuracy of 0.5 means that the decoded message is random. In Figure 8, we show the bit accuracy of the different watermarking algorithms for different bitstring lengths. For schemes with a strength parameter, we vary the strength to show the bit accuracy-quality trade-off, where we measure quality via perplexity. Our approach achieves a better bit accuracy-quality trade-off than prior work, especially in low-distortion settings (i.e., low-perplexity settings) and at larger bitstring lengths.

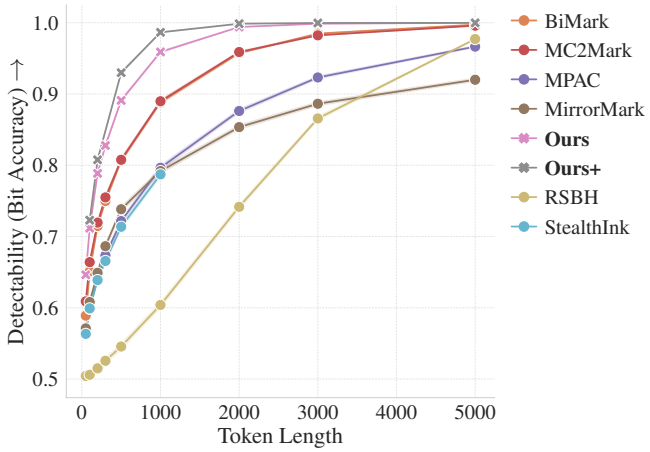

Figure 9. **Bit Accuracy with Respect to the Number of Tokens:** We compare how the bit accuracy scales with the number of generated tokens. The hyperparameters of each watermarking scheme have been selected to have a similar impact on quality, and all schemes use a bit string of 32 bits.

Scaling with the Number of Tokens In Figure 9, we show how the bit accuracy scales with the number of tokens. For this experiment, we select all watermark hyperparameters to operate in a low-distortion regime (i.e., we use $\varepsilon = 0$ for our watermark), which ensures a similar quality impact across the compared methods, and we use a 32-bit bitstring. Across all tested baselines, our scheme scales more favorably with the number of tokens. At 500 tokens, the bit accuracy of our scheme exceeds 90%, whereas the best baseline achieves around 80%.

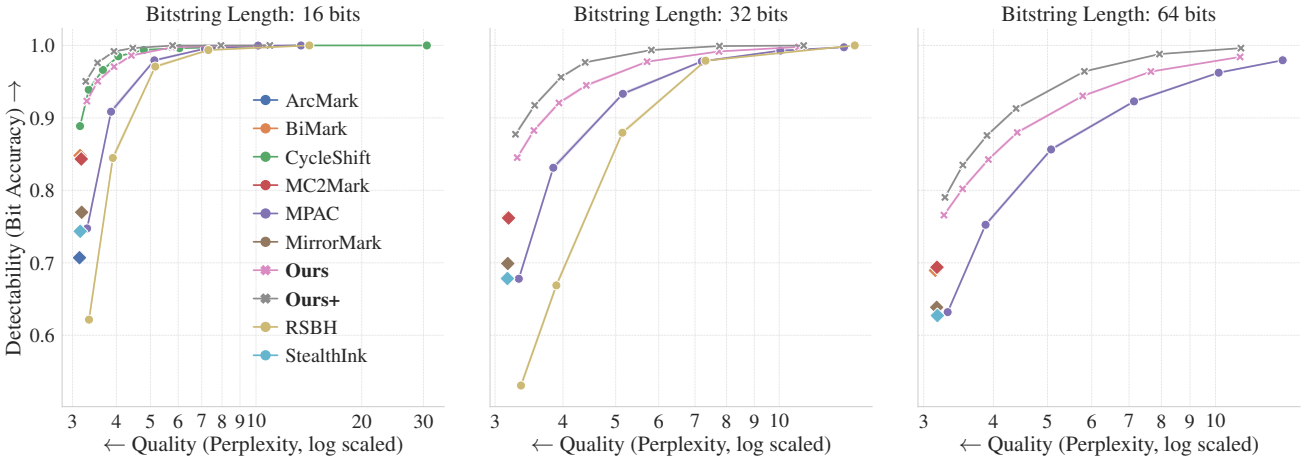

Figure 8. **Comparison of the Bit Accuracy-Quality Trade-Off:** We compare the trade-off between bit accuracy and text quality (log PPL) for various multibit watermarks and three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5. For Cycle-Shift, only payloads below $\log_2 |\Sigma| \approx 20$ bits are supported. For ArcMark, only payloads below 16 bits are supported. For RSBH, Qu et al. (2025) does not provide a configuration for a 64-bit payload.

D.3. Additional Quality Metrics

In this part, we compare the message accuracy-quality trade-off of our watermark against the baselines using benchmark accuracy as a proxy for quality instead of perplexity.

Experimental Setup For all watermark configurations, we measure benchmark accuracy using the EleutherAI evaluation harness (Gao et al., 2024) on GSM8k (Cobbe et al., 2021), HumanEval (Chen et al., 2021), and MBPP (Austin et al., 2021). We then report the average accuracy across the three benchmarks.

Improved Message Accuracy Figure 10 shows that, when using benchmark accuracies as a proxy for quality, our approach still achieves a better message accuracy-quality trade-off than prior work.

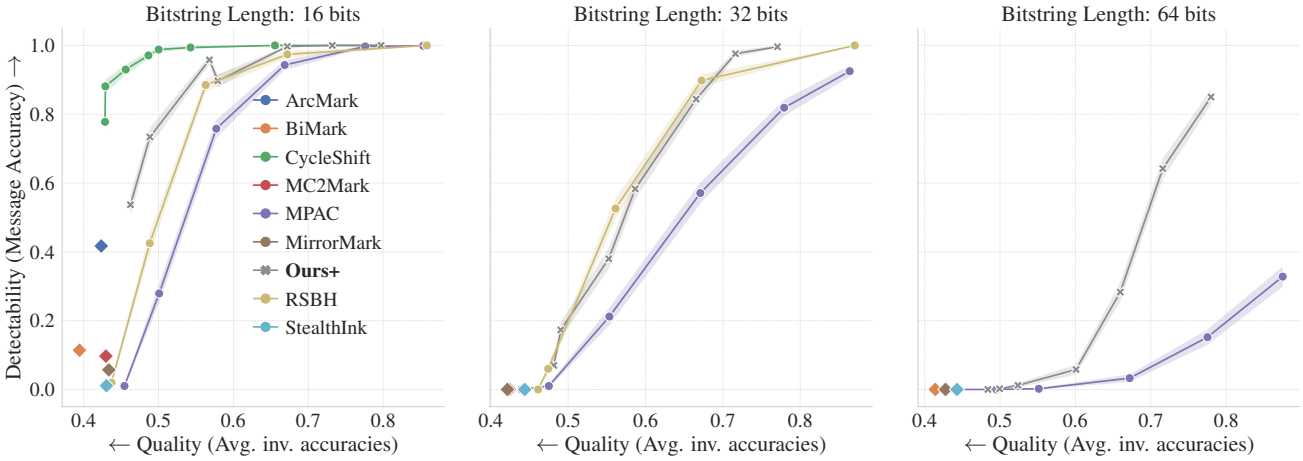

Figure 10. **Comparison of the Message Accuracy-Quality Trade-Off using LLM Benchmarks:** We compare the trade-off between message accuracy and text quality (average benchmark accuracy) for various multibit watermarks and three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5. For Cycle-Shift, only payloads below $\log_2 |\Sigma| \approx 20$ bits are supported. For ArcMark, only payloads below 16 bits are supported. For RSBH, Qu et al. (2025) does not provide a configuration for a 64-bit payload.

D.4. Additional Model

Here, we reproduce our main results using MINISTRAL-3-14B instead of LLAMA3.1-8B.

Experimental Setup We use the same experimental setup as in Sec. 5: for each watermark configuration, we generate completions of between 250 and 350 tokens for 1000 prompts from ELI5, sample bitstrings uniformly at random, and measure quality using QWEN3-30B perplexity. The only difference is that the watermarked text is generated with MINISTRAL-3-14B instead of LLAMA3.1-8B, using the inference parameters from App. A.1.

Results In Figures 11 and 12, we observe the same trends as in Figures 2 and 8. Our watermark achieves a better bit accuracy-quality trade-off than prior work, especially for larger bitstring lengths and in low-distortion regimes. For message accuracy, our watermark also outperforms prior work at larger, more practical payloads (32 and 64 bits). As in the LLAMA3.1-8B evaluation, the ordering of the baselines differs between bit accuracy and message accuracy, since ArcMark, Cycle-Shift, and RSBH are designed to improve whole-message recovery rather than per-bit accuracy alone. Finally, Figure 13 shows that the confidence-based BA@1% FPR metric follows the same pattern as with LLAMA3.1-8B: the stateless version of our watermark provides the best confidence bit accuracy-quality trade-off, whereas Ours+ underperforms on confidence-based metrics because it prioritizes the expected bit accuracy rather than per-bit confidence.

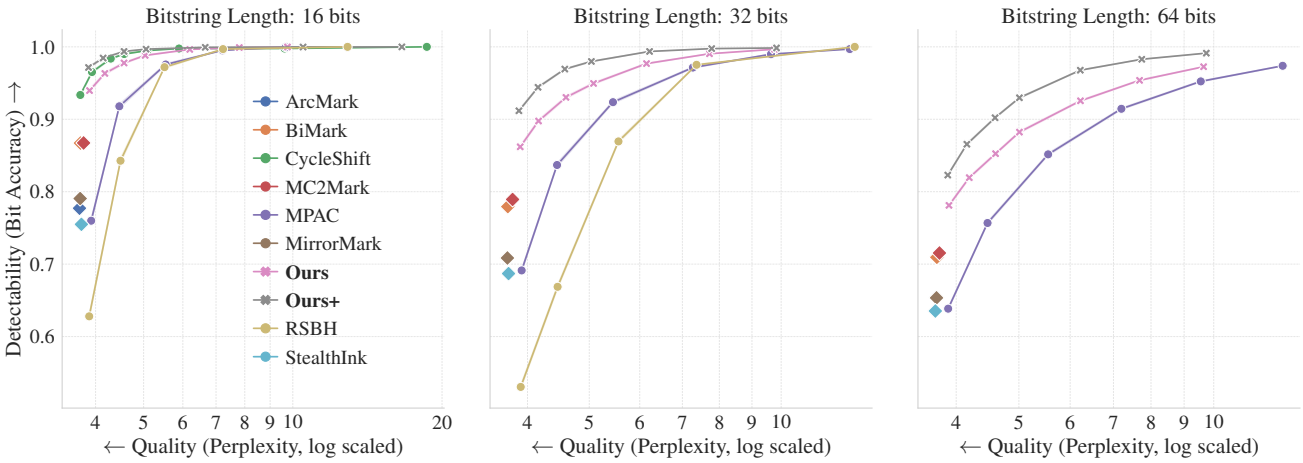

Figure 11. **Comparison of the Bit Accuracy-Quality Trade-Off with MINISTRAL-3-14B:** We compare the trade-off between bit accuracy and text quality (log PPL) for various multibit watermarks and three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by MINISTRAL-3-14B using prompts from ELI5. For Cycle-Shift, only payloads below $\log_2 |\Sigma| \approx 20$ bits are supported. For ArcMark, only payloads below 16 bits are supported. For RSBH, Qu et al. (2025) does not provide a configuration for a 64-bit payload.

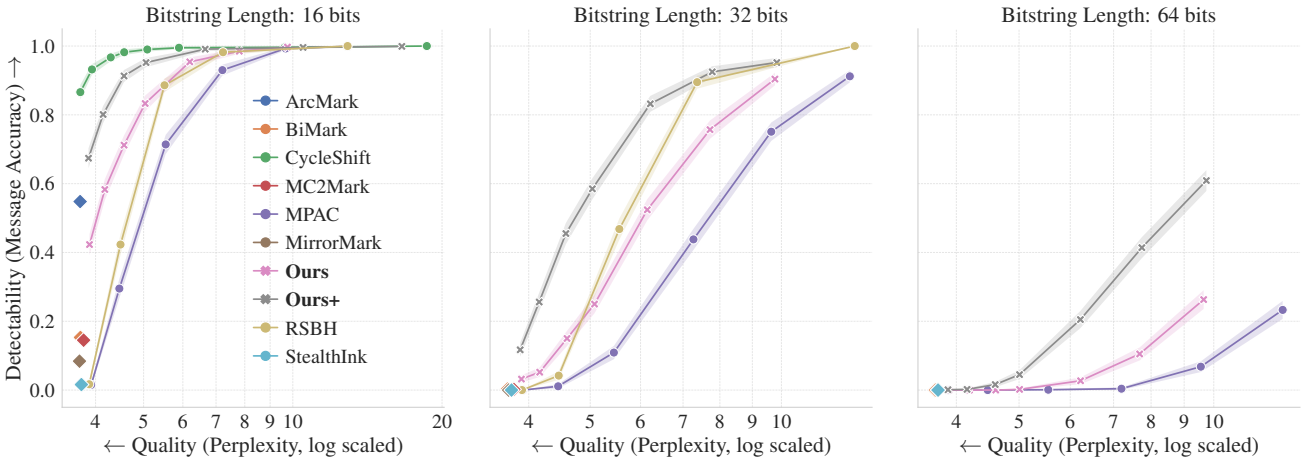

Figure 12. **Comparison of the Message Accuracy-Quality Trade-Off with MINISTRAL-3-14B:** We compare the trade-off between message accuracy and text quality (log PPL) for various multibit watermarks and three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by MINISTRAL-3-14B using prompts from ELI5.

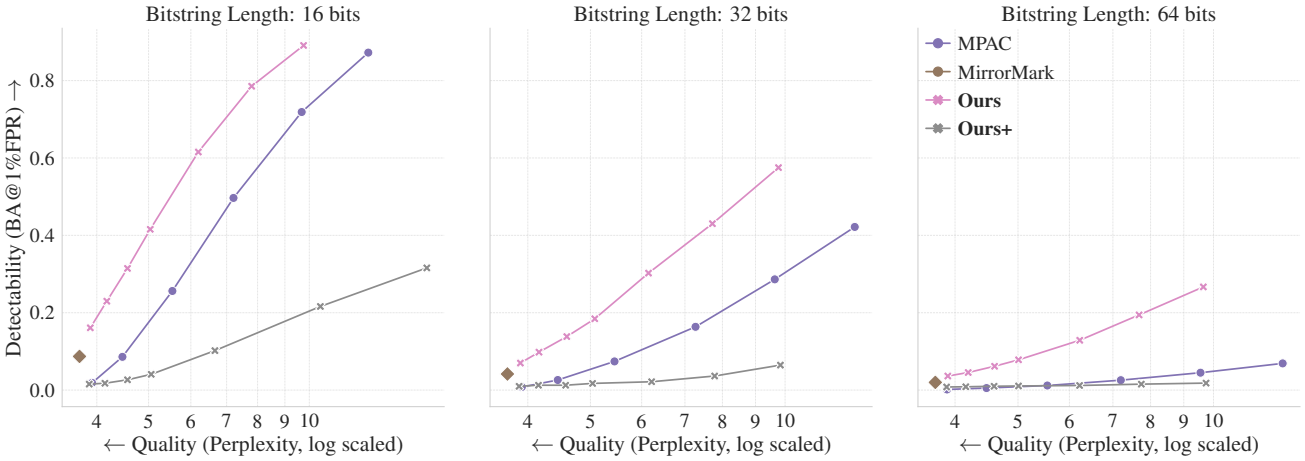

Figure 13. **Comparison of the Confidence Bit Accuracy-Quality Trade-Off with MINISTRAL-3-14B:** We compare the trade-off between confidence bit accuracy (BA@1%FPR) and text quality (log PPL) across several multibit watermarks and three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by MINISTRAL-3-14B using prompts from ELI5.

D.5. Ablation on the Stateful Encoder Remaining Bits

In this part, we ablate the choice of the remaining-token parameter T from the stateful encoder from Sec. 3.2.

We recall that, in Sec. 3.2, T denotes the total number of generated tokens used in the stateful encoder's expected per-bit recovery estimate. Hence, at generation step t, the quantity $T - t$ is the number of future token positions remaining under the worst-case null assumption. Varying this value changes how strongly the stateful encoder prioritizes bits whose current decoding statistic d_t^i is close to the decision boundary. In the main experiments, we use the averaged score over $\mathcal{T} \coloneqq \{200, 300, 500, 1000, 2000\}$.

Experimental Setup To justify the averaged choice used in the main experiments, we ablate the values of T. Specifically, we compute the message accuracy of our stateful encoder with $T \in \{64, 200, 300, 500, 1000, 2000\}$, as well as when averaging Equation (7) over $\mathcal{T}$ for different token lengths.

Results Figure 14 shows that, in most cases, using the same T as the length of the generated text is optimal (i.e., using $T = 2000$ when generating 2000-token-long text). When using the average, we find that it systematically remains close to the best-performing T. The choice of T has a significant effect on message accuracy: unlike bit accuracy, message accuracy is a whole-string metric, so moderate per-bit changes can translate into large differences in full-message recovery. This result supports using the averaged score, since it avoids needing to estimate the generation length while retaining most of the gain from the best-matched T values.

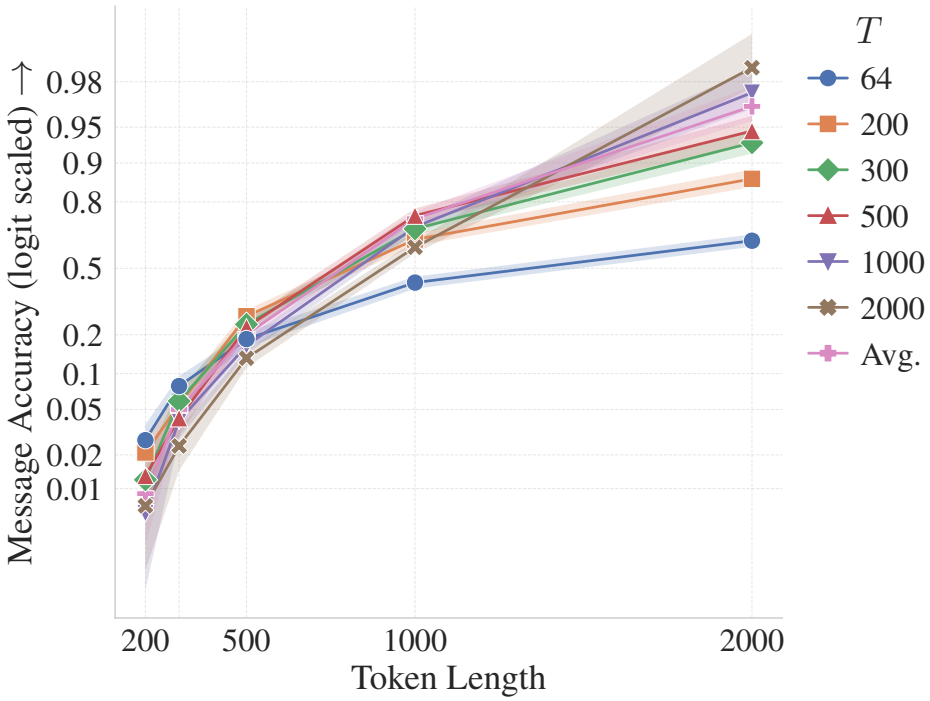

Figure 14. **Ablation on the Remaining Bits T:** We compare the message accuracy of our stateful binomial encoder for different values of T at different token lengths.

D.6. Ablation on Position Allocation

Here, we verify whether our method, which encodes every bit at once using binomial encoding, improves over position allocation.

Experimental Setup We use the same experimental setup as in Sec. 5: for each watermark configuration, we generate completions of between 250 and 350 tokens with LLAMA3.1-8B for 1000 prompts from ELI5, sample bit strings uniformly at random, and measure quality using QWEN3-30B perplexity. For the watermark, we combine binomial encoding with position allocation. We split the m-bit string into k segments. Then, at each generation step, we use the context hash to select which segment to encode and use binomial encoding to encode the m/k bits of this segment. In particular, using $k := 1$ corresponds to using our watermark as described in Sec. 3, and using $k := m$ corresponds to using MPAC with Soft-PPL as the underlying watermarking scheme.

Binomial Encoding Outperforms Position Allocation Figure 15 shows the message accuracy–quality trade-off for various numbers of segments ($k \in \{1, 4, 8, 16, 32\}$), as well as for our method using the stateful encoder (Ours+). We find that, with the stateless encoder, binomial encoding outperforms position allocation: using a smaller number of segments gives a better message accuracy–quality trade-off in relevant settings, but the advantage shrinks as the number of segments decreases. Importantly, when using the stateful encoder (which is not compatible with position allocation), our method significantly outperforms position allocation. This finding confirms our intuition that position allocation can be limiting for the power of multibit watermarking algorithms.

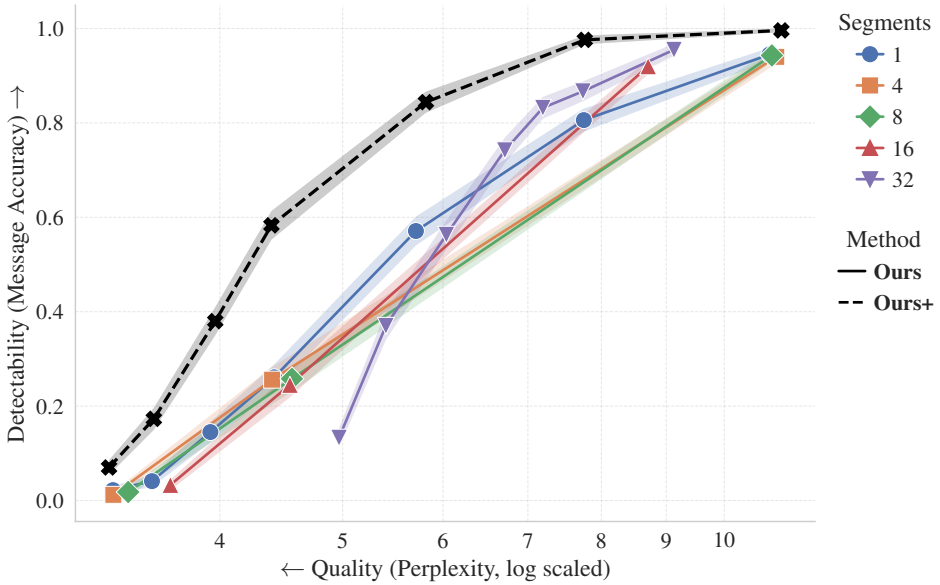

Figure 15. **Message Accuracy for Different Segment Sizes:** We compare how our method combines with position allocation by splitting the 32 bits into $k \in \{1, 4, 8, 16, 32\}$ segments.

E. SynthID-Text Multibit Watermark

In this section, we propose replacing the Soft PPL watermarking scheme (Equation (8)) with the SynthID-Text watermarking scheme from Dathathri et al. (2024).

Zero-Bit SynthID-Text At each token position t, let $\omega_{<t} \in \Sigma^{t-1}$ be the current context and $p_t \in \Delta(\Sigma)$ the next-token probability distribution given by the LLM. With SynthID-text, each token $u \in \Sigma$ is assigned a sequence of n_{layer} Bernoulli scores $[G_t(u,1), \dots, G_t(u, n_{\text{layer}})]$ with probability 0.5 called g-values. Then, to get the watermarked probability distribution we recursively iterate the following equation,

$$\forall u \in \Sigma, \quad p_t^k(u) = p_t^{k-1}(u) \left[1 + G_t(u,k) - \sum_{v \in \Sigma} p_t^k(v) G_t(v,k) \right], \tag{11}$$

with $p_t^0 := p_t$. The final watermarked probability distribution is given by $p_t^{n_{\text{layer}}}$, and each iteration of Equation (11) is called a layer. Importantly, this process is distortion-free, which means that in expectation over the scores, $p_t^{n_{\text{layer}}}$ is equal to p_t.

Multibit SynthID-Text We focus on a single layer first and drop the layer index from the notation. For a single layer, the zero-bit version consists in assigning a Bernoulli score to each token and then applying Equation (11). We simply propose replacing the Bernoulli scores with the complementary final score $\tilde{G}_t$ from Sec. 3.1. To keep the single layer transformation distortion-free, we need to adjust Equation (11), as explained in Gloaguen et al. (2026b),

$$\forall u \in \Sigma, \quad q_t(\tilde{G}, p_t)(u) = p_t(u) \left[1 + \sigma \left(\tilde{G}_t(u) - \sum_{v \in \Sigma} p_t(v) \tilde{G}_t(v) \right) \right], \tag{12}$$

where $\sigma := \left(\max_{v \in \Sigma} \tilde{G}_t(v) - \min_{v \in \Sigma} \tilde{G}_t(v) \right)^{-1}$. For the n_{layer} version, we simply iterate Equation (12) using at each layer a different seed to sample the m Bernoulli variables and encode the same bitstring M with complementary scores (Equation (2)).

For detection, we adjust the weighted mean detector from Dathathri et al. (2024), with

$$\hat{M} := \text{round} \left(\frac{1}{|\omega| \, n_{\text{layer}}} \sum_{t=1}^{|\omega|} \sum_{k=1}^{n_{\text{layer}}} w_k \left[G_t^1(\omega_t, k), \dots, G_t^m(\omega_t, k) \right] \right), \tag{13}$$

where $w_1, \ldots, w_{n_{\text{layer}}}$ are linearly decaying weights (from 10 down to 1) normalized such that $\sum_{k=1}^{n_{\text{layer}}} w_k = n_{\text{layer}}$.

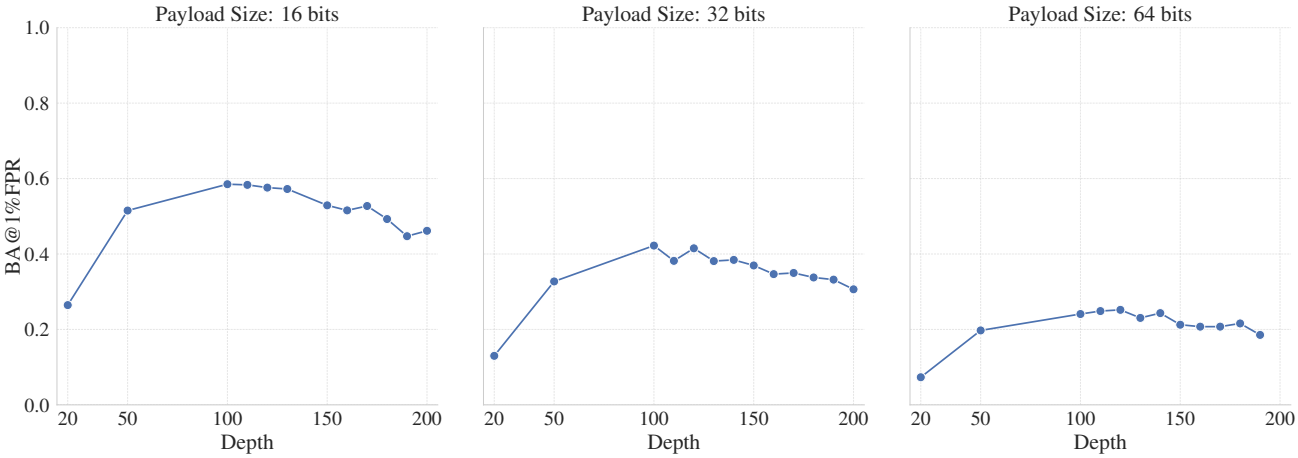

Figure 16. **Comparison of the Detectability of Various SynthID Layers:** We compare the BA@1%FPR for various numbers of SynthID layers n_{layer} across three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5.

Table 2. **Distortion-Free Watermark Evaluation:** Message accuracy, BA@1%FPR, and bit accuracy (BA) for 16-bit, 32-bit, and 64-bit bitstrings. All watermarks evaluated here are distortion-free, including Ours using SynthID. Missing values indicate that either the scheme does not support the metric or the bitstring length. Metrics are averaged over 1000 samples where each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5. The highest value in each metric column is **bolded**.

	16 bits			32 bits			64 bits		
Watermark	Message Acc.	BA@1%FPR	BA	Message Acc.	BA@1%FPR	BA	Message Acc.	BA@1%FPR	BA
Cycle-Shift	**0.778**	–	**0.889**	–	–	–	–	–	–
ArcMark	0.417	–	0.707	–	–	–	–	–	–
BiMark	0.114	–	0.848	0.002	–	0.762	0.000	–	0.690
MC2Mark	0.097	–	0.843	0.000	–	0.762	0.000	–	0.694
MirrorMark	0.057	0.077	0.770	0.000	0.034	0.699	0.000	0.019	0.639
StealthInk	0.011	–	0.744	0.000	–	0.678	0.000	–	0.627
Ours (SynthID)	0.158	**0.589**	0.872	**0.003**	**0.413**	**0.793**	0.000	**0.270**	**0.718**
Ours+ (SynthID)	0.045	0.446	0.816	0.000	0.322	0.749	0.000	0.190	0.687

Calibrating the Number of Layers In Figure 16, we compute the BA@1%FPR for various numbers of tournament layers n_{layer}. We find that using the multibit SynthID variant requires a higher number of layers n_{layer} than the original zero-bit watermark, with detectability reaching a maximum around $n_{\text{layer}} = 100$. We also find that using the stateful encoder from Sec. 3.2 (i.e., replacing $\tilde{G}$ with $\tilde{G}'$) does not improve the bit accuracy. We hypothesize that this occurs because the optimal number of layers depends on the distribution of the random scores. With the stateful encoder, the distribution of scores is different at each step, but for detection we must keep a fixed number of tournament layers. Therefore, the potential gains in accuracy from using the stateful encoder are negated by the loss incurred from using a suboptimal number of tournament layers.

Results: Our Scheme with SynthID Outperforms Most Distortion-Free Watermarks Table 2 shows the message accuracy, BA@1%FPR, and bit accuracy for all considered distortion-free schemes, including ours with SynthID (using $n_{\text{layer}} = 100$). We find that in most practical scenarios (e.g., larger bitstrings), our scheme outperforms all prior baselines on all considered metrics. At the same time, as anticipated, the stateful encoder does not improve message accuracy with SynthID compared to the stateless encoder.

F. Unconstrained Soft PPL Scheme

In this section, we propose replacing the Lagrangian constraint from the Soft PPL watermarking scheme (Equation (8)) with an unconstrained version. We find that, while being computationally more efficient, this re-parametrization yields similar performance to the original one.

Experimental Setup We use the same experimental setup as in Sec. 5: for each watermark configuration, we generate completions of between 250 and 350 tokens for 1000 prompts from ELI5, sample bitstrings uniformly at random, and measure quality using QWEN3-30B perplexity. We compare the constrained version used in the main paper, where λ is obtained by solving the Soft PPL constraint equation in Equation (8) (effectively using ε as a scheme parameter), against *Ours (unc.)*, where λ is varied directly as the scheme parameter.

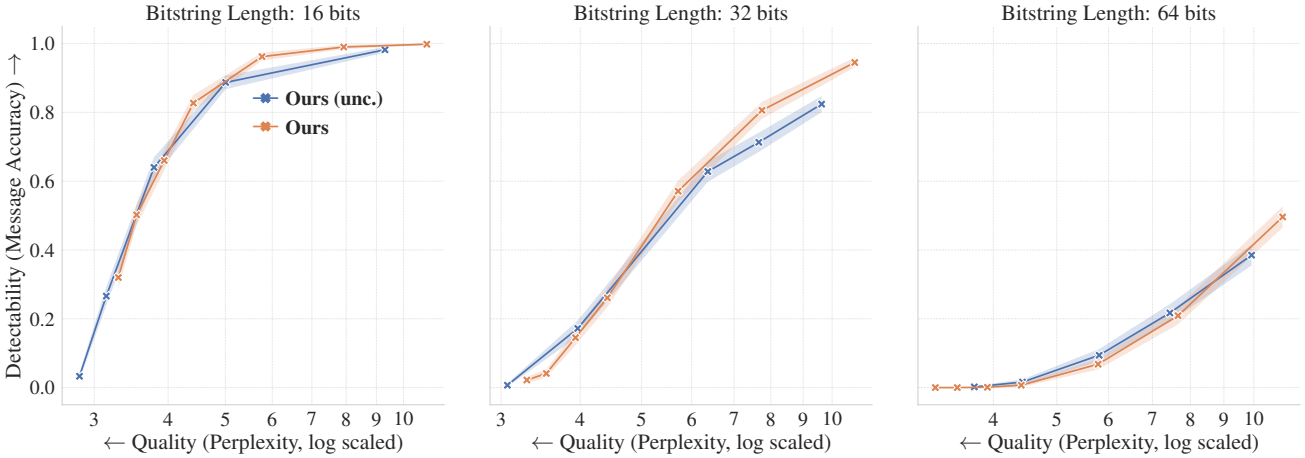

Figure 17. **Comparison of the Message Accuracy-Quality Trade-Off for Unconstrained Soft PPL:** We compare the trade-off between message accuracy and text quality (log PPL) for the constrained Soft PPL parametrization (Ours) and the unconstrained parametrization using λ as a scheme parameter (Ours (unc.)), for three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5.

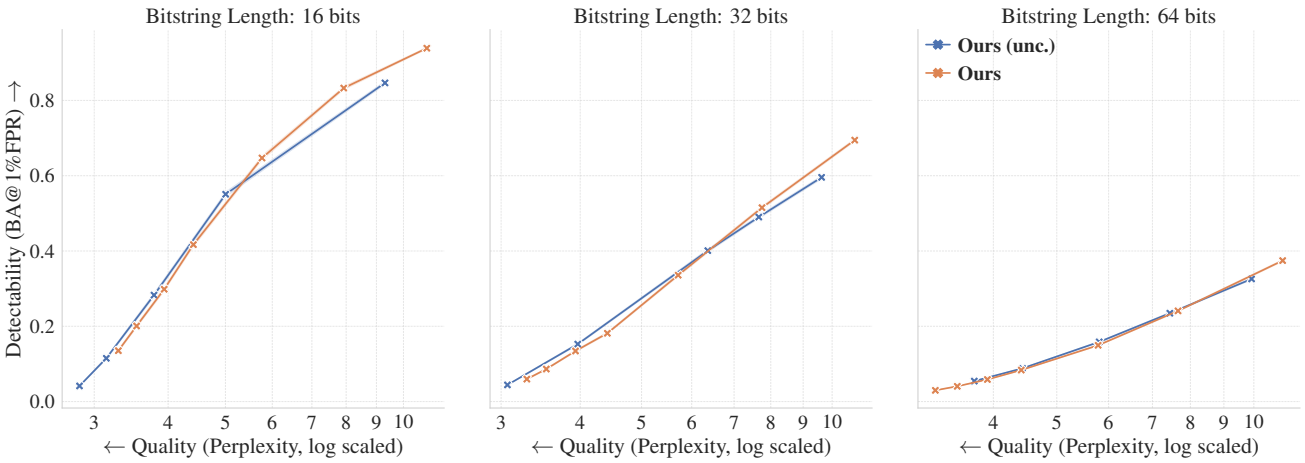

Figure 18. **Comparison of the Confidence Bit Accuracy-Quality Trade-Off for Unconstrained Soft PPL:** We compare the trade-off between confidence bit accuracy (BA@1% FPR) and text quality (log PPL) for the constrained Soft PPL parametrization (Ours) and the unconstrained parametrization using λ as a scheme parameter (Ours (unc.)), for three bitstring lengths (16, 32, and 64 bits). Each sample contains between 250 and 350 tokens and is generated by LLAMA3.1-8B using prompts from ELI5.

Results As shown in Figures 17 and 18, the constrained and unconstrained versions yield similar detectability-quality trade-offs across payload sizes. The unconstrained version is more computationally efficient because it avoids repeatedly estimating the expectation over watermark scores and solving for λ at each generation step. However, it is harder to

parametrize and compare across settings: a fixed ε has the same expected quality impact across bitstring lengths by construction, whereas the quality impact of a fixed λ depends on the bitstring length. This is why we use the constrained version in our main experiments despite the increased computational cost.

G. Broader Impacts and Licenses

G.1. Societal Impacts

Positive Impacts Multibit LLM watermarks can support provenance mechanisms for generated text. Compared to zero-bit watermarks, they can encode richer metadata such as user identifiers, timestamps, or licensing information, which may help platform operators audit generated content, investigate misuse, and provide transparency around model-generated text. Our work also emphasizes confidence-calibrated decoding, which can reduce overconfident interpretation of weak watermark signals.

Risks and Limitations The same ability to encode richer metadata can create privacy risks if identifiers are embedded without an appropriate governance policy, user notice, access control, or retention policy. Watermark detectors may also be misused as evidence of authorship beyond their statistical scope, especially when text is edited, paraphrased, or generated by unwatermarked systems. Our robustness experiments in Sec. 5.3 show that multibit watermark recovery remains fragile under text modifications, so deployment should not rely on these methods as the only provenance or abuse-prevention mechanism. Finally, a provider-controlled watermark can create asymmetric power: users may be unable to inspect what metadata is embedded or who can decode it. Responsible deployment therefore requires clear disclosure, limited and purpose-specific payloads, key-management procedures, and calibration thresholds that reflect the intended decision risk.

G.2. Existing Assets

Datasets

- **ELI5 (Fan et al., 2019).** We use questions from ELI5 as prompts. The official repository for recreating ELI5 is distributed under a BSD license for the accompanying software, but the dataset itself is derived from Reddit and CommonCrawl content.
- **C4 realnewslike.** We use the realnewslike subset of C4 to estimate frequencies for RSBH and to calibrate zero-bit detection on human-generated text. The Hugging Face dataset card lists C4 under ODC-BY and notes that use is also subject to the Common Crawl terms for the underlying content.
- **GSM8K (Cobbe et al., 2021).** We use GSM8K through the EleutherAI evaluation harness as one of the benchmark-quality metrics. The Hugging Face dataset card lists the license as MIT.
- **HumanEval (Chen et al., 2021).** We use HumanEval through the EleutherAI evaluation harness as one of the benchmark-quality metrics. The official OpenAI repository is distributed under the MIT license.
- **MBPP (Austin et al., 2021).** We use MBPP through the EleutherAI evaluation harness as one of the benchmark-quality metrics. The Hugging Face dataset card for the Google Research dataset lists the license as CC-BY-4.0.

Models

- **LLAMA3.1-8B.** We use the instruction-tuned Llama 3.1 8B model for the main generation experiments. The Hugging Face model card lists the model under the Llama 3.1 Community License and requires gated access.
- **MINISTRAL-3-14B.** We use Ministral 3 14B Instruct for the additional-model experiments. The Hugging Face model card lists Apache-2.0 as the license.
- **QWEN3-30B.** We use Qwen3-30B-A3B to measure perplexity of generated samples. The Hugging Face model card lists Apache-2.0 as the license.

H. Proof of Lemma 3.1

Lemma 3.1. *Let $t \in \mathbb{N}$. Let $\omega_{\leq t} \in \Sigma^t$ be a partially generated sequence, and let d_t^i denote the signed decoding statistic for bit i, as defined in Equation (5). Assuming that future tokens are generated independently of the watermark, we have for all $i \in \{1, \ldots, m\}$ and $T > t$*

$$P[d_T^i \geq 0 \mid d_t^i] \approx \Phi\left(\frac{d_t^i}{\sqrt{T-t}}\right), \tag{14}$$

where Φ is the standard normal CDF.

Proof of Lemma 3.1. Fix a bit index $i \in \{1, \ldots, m\}$, where m is the message length in bits. Recall that $\omega_{\leq t} \in \Sigma^t$ is the generated prefix of length t, that ω_k is the realized token at position k, and that $\tilde{G}_k^i(\omega_k) \in \{0, 1\}$ is the complementary score assigned to that realized token for bit i at position k. Also recall that

$$d_t^i = 2\sum_{k=1}^{t} \tilde{G}_k^i(\omega_k) - t \tag{15}$$

is the signed decoding statistic of bit i after the first t generated tokens, and that T is the total number of generated tokens, so $T - t$ is the number of future token positions.

For each future position $k \in \{t+1, \ldots, T\}$, define the increment

$$X_k^i := 2\tilde{G}_k^i(\omega_k) - 1 \in \{-1, +1\}. \tag{16}$$

Because the future tokens are assumed to be generated independently of the watermark, each future complementary score $\tilde{G}_k^i(\omega_k)$ is Bernoulli(0.5). Equivalently, the increments $X_{t+1}^i, \ldots, X_T^i$ are i.i.d. Rademacher random variables with mean 0 and variance 1. Therefore the final signed decoding statistic

$$d_T^i = 2\sum_{k=1}^{T} \tilde{G}_k^i(\omega_k) - T \tag{17}$$

decomposes as

$$d_T^i = d_t^i + \sum_{k=t+1}^{T} X_k^i. \tag{18}$$

Conditioning on the current statistic d_t^i, we get

$$P[d_T^i \geq 0 \mid d_t^i] = P\left[\sum_{k=t+1}^{T} X_k^i \geq -d_t^i \,\middle|\, d_t^i\right]. \tag{19}$$

Now $\sum_{k=t+1}^{T} X_k^i$ is the sum of exactly $T - t$ i.i.d. mean-zero, variance-one random variables, where $T - t$ is the number of future positions. Hence, by the central limit theorem,

$$\frac{1}{\sqrt{T-t}} \sum_{k=t+1}^{T} X_k^i \approx Z, \qquad Z \sim \mathcal{N}(0, 1). \tag{20}$$

Substituting this Gaussian approximation into the previous display yields

$$P[d_T^i \geq 0 \mid d_t^i] \approx P\left[Z \geq -\frac{d_t^i}{\sqrt{T-t}}\right] = \Phi\left(\frac{d_t^i}{\sqrt{T-t}}\right), \tag{21}$$

where Φ is the CDF of the standard normal distribution. This is exactly Equation (6). □