

Revisiting Segmentation in Multi-bit LLM Watermarking

Beining Huang^{1,2*} Youdu Liu^{2*} Fei Sun¹
Du Su^{1†} Shuangjie Fu^{1,2} Huawei Shen^{1,2} Xueqi Cheng^{1,2}

¹State Key Laboratory of AI Safety,

Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China

²University of Chinese Academy of Sciences, Beijing, China

bnhuang2022@gmail.com liuyoudu24@mails.ucas.ac.cn

{sunfei, sudu, shenhuawei, cxq}@ict.ac.cn

Abstract

Text watermarking enables provenance tracing of LLM-generated text by embedding detectable signals, while multi-bit watermarking further encodes payloads such as user identities or model versions. Scaling multi-bit watermarking to large payloads is challenging as exhaustive decoding grows exponentially with payload length. Existing methods commonly address this by segmenting the payload and decoding each segment independently, but the impact of this design on decoding accuracy remains insufficiently understood. In this paper, we revisit segmentation in multi-bit watermarking and reveal a drawback of the commonly adopted paradigm: segmentation splits the available token evidence among segments, leaving fewer observations for each local decoding decision and thereby reducing decoding reliability. We theoretically characterize this accuracy–efficiency trade-off and empirically validate the claim across diverse settings. Additionally, we introduce a distribution-balanced detector to mitigate message-dependent bias. Experiments across multiple datasets and models show that our method achieves improved accuracy and robustness under editing attacks compared to existing methods. Our code is available at <https://github.com/YouduLiu/SDBMW>

1 Introduction

Large language models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks (Bogadi, 2025; Wang and Zhang, 2024). Despite these advances, the risk of malicious use has also increased. For example, LLMs can be used to generate fake news (Hu et al., 2025), while Microsoft has documented threat actors using LLMs to draft content for spear-phishing campaigns.¹

^{*}Equal contribution.

[†]Corresponding author.

¹Microsoft Threat Intelligence, “Staying ahead of threat actors in the age of AI,”

To address these issues, a variety of techniques have been proposed to identify LLM-generated text (Mitchell et al., 2023; Hans et al., 2024; Chen et al., 2025; Kirchenbauer et al., 2023; Dathathri et al., 2024). Among them, LLM watermarking (Kirchenbauer et al., 2023; Dathathri et al., 2024) is a promising approach due to its low false-positive rate and theoretical guarantees. The mainstream watermarking paradigm perturbs the model output distribution during token sampling, thereby injecting a statistical signal into token-frequency patterns, enabling identification of LLM-generated text afterwards.

However, conventional watermarking only supports binary attribution, i.e., determining whether a text is model-generated. Recent studies (Fernandez et al., 2023; Wang et al., 2024; Yoo et al., 2024; Qu et al., 2025; Jiang et al., 2025; Xu et al., 2026a,b) extend watermarking to embed multi-bit payloads, such as user IDs, model versions, and timestamps, enabling finer-grained provenance analysis. Following prior work, we refer to each payload as a message, with each candidate message corresponding to a distinct watermark signal. Multi-bit watermarking embeds the signal associated with a target message into generated text and recovers the message by evaluating candidate signals and selecting the one with the strongest statistical evidence.

As the message length increases, the message space grows exponentially, making exhaustive detection computationally unaffordable. To improve scalability, a segmentation framework is proposed: the message is partitioned into multiple segments, to which disjoint subsets of tokens are assigned for watermark embedding and detection. As segmentation substantially reduces detection complexity, it is important for the practical deployment of multi-bit watermarking in real-world scenarios with large message spaces.

Although segmentation has been adopted in recent works (Yoo et al., 2024; Qu et al., 2025; Jiang

et al., 2025; Xu et al., 2026b), its impact on detection accuracy has not been fully investigated. Yoo et al. (2024) argue that segmentation improves accuracy by reducing the search space. However, their method, MPAC, achieves relatively low detection accuracy. Qu et al. (2025) attribute this issue to uneven segmentation and propose an alternative segmentation strategy, but the resulting performance remains suboptimal. Xu et al. (2026a) construct green lists based on the majority bit of the message and employ message segmentation to increase the green-list ratio, thereby improving generation quality. However, they do not systematically investigate how segmentation design affects decoding accuracy under a fixed green-list ratio.

In this work, we conduct a theoretical and empirical study on the impact of segmentation. Our analysis reveals a trade-off: while increasing the number of segments improves detection efficiency, it simultaneously weakens the statistical significance of the watermark signal, thus degrading detection accuracy and robustness. We validate this trend through experiments across various message lengths, watermark strengths, and text lengths. These findings suggest that the number of segments should be chosen carefully in practical applications to balance efficiency and detection performance.

Moreover, segmentation-based methods assign only a subset of tokens to each segment, making watermark signals less prominent. Thus, the target message can be obscured by spurious candidates. We observe that different messages exhibit different inherent signal strengths in model-generated text due to discrepancies induced by the underlying language model distribution. To mitigate this, we propose a debiased detector that leverages a proxy model distribution to compensate for message-dependent bias, thereby improving the distinguishability of the target message.

Based on these insights, we propose **SDBMW** (Segmented Distribution-Balanced Multi-bit Watermarking) with an implementation amenable to parallel computation. Experiments across multiple datasets and models demonstrate that SDBMW achieves better detection accuracy and robustness than existing multi-bit watermarking methods.

Our main contributions are as follows:

- We analyze and validate the impact of message segmentation on the detection accuracy of multi-bit watermarking.
- We propose a debiased detector to compensate

for message-dependent statistical bias.

- We conduct extensive experiments under various settings, demonstrating the superior accuracy and robustness of our method.

2 Related Work

Payload-free watermark detection Payload-free, or zero-bit, LLM watermarking embeds no recoverable message. Its detector only decides whether a text carries a watermark.

General LLM watermark methods follow a token-level *red-green list* paradigm: at each generation step, a pseudo-random function utilizes prefix tokens as a seed to partition the vocabulary into a “green list” of favored tokens and a “red list” of discouraged tokens. Kirchenbauer et al. (2023) introduced the red-green list framework. Subsequent work broadens it in several directions: Kuditipudi et al. (2024) introduce a distortion-free sampling rule, Kirchenbauer et al. (2024); Chang et al. (2024) analyze reliability under text edits or black-box constraints, Wang et al. (2025) optimize the quality–detectability trade-off, and Dathathri et al. (2024) show deployment-scale integration through SynthID. This line establishes the standard binary-attribution interface, while remaining unable to support further regulatory tasks like tracing malicious users.

Multi-bit watermarking Multi-bit watermarking extends this detection interface by recovering a payload, such as a user identity, model version, or timestamp, rather than only deciding whether a watermark exists. Fernandez et al. (2023) instantiate this idea through cyclic-shift signals. Wang et al. (2024) formulate Codable Text Watermarking (CTWL) and use a proxy language model to construct high probability green lists during generation. Zamir (2024) studies distortion-free formulations of multi-bit watermarking.

For a B -bit payload, exact decoding must score 2^B message-conditioned signals, so the search cost grows exponentially with the payload length. This cost motivates scalable multi-bit methods that decompose the message space into smaller decoding problems. MPAC (Yoo et al., 2024) allocates tokens to message positions, and decodes each message position separately, enabling lower latency. RSBH (Qu et al., 2025) further develops this segment-based direction by packing bits into segments with balanced token-to-segment assign-

ment and Reed–Solomon error correction. In a many-user setting, [Cohen et al. \(2025\)](#) decompose attribution signals across text blocks to trace outputs to adaptive users.

These methods share the same scalability principle: replacing one full-message search with several local decoding decisions. This decomposition reduces search cost, yet it changes the statistical regime: with s segments, each local decision is supported by roughly $|x|/s$ tokens, so allocation imbalance, sampling noise, or text edits can more easily overturn a segment-level score. The accuracy–efficiency effect of such evidence splitting remains under-characterized.

CTWL and SDBMW both use proxy models for multi-bit watermarking. CTWL applies it to construct probability-balanced vocabulary partitions during generation, and reconstructs the corresponding proxy-aided message function for payload recovery ([Wang et al., 2024](#)). This design targets quality-preserving multi-bit encoding, but the proxy information is tied to the construction of the token partition and does not calibrate candidate-wise score bias after the text has been edited or truncated. Instead of applying the proxy model to reshape the encoding-side partition, SDBMW uses it at detection time to estimate each candidate message’s null mean and variance, then compares normalized scores. This stage shift makes the detector more robust to candidate-wise distributional bias, especially when segmentation or text edit attacks weaken the true payload evidence. In this sense, SDBMW is complementary to CTWL: it turns proxy information from token-partition balancing into score calibration for more reliable and robust payload extraction.

Recent work has explored segmentation from the perspective of preserving generation quality. Majority Bit-Aware Watermarking ([Xu et al., 2026a](#)) introduces a majority-bit construction to improve the quality of multi-bit watermarked text, while WaterMod ([Park et al., 2026](#)) proposes modular token-rank partitioning particularly targeting low-entropy generation. StealthInk ([Jiang et al., 2025](#)) uses position-allocated symbols and probability reweighting to improve textual quality and spoofing resistance ([Jovanović et al., 2024](#)), while XMark ([Xu et al., 2026b](#)) combines multi-shard vocabulary partitioning with leave-one-shard-out encoding to balance generation quality and recovery reliability. These approaches provide recent complementary directions in multi-bit text water-

marking for LLMs.

3 Preliminaries

This section introduces the framework of multi-bit watermarking and the related notation.

3.1 Conventional Multi-bit Watermarking

Let x denote the generated text, a the prompt, m the watermark message, M the language model, V the vocabulary and v a candidate token. We use $P_M(\cdot | a, x_{:t})$ to denote the next-token probability distribution induced by M given the prompt a and the generated prefix $x_{:t}$.

At step t , when generating token x_t , we use a mask vector $r(m, x_{:t}) \in \{0, 1\}^{|V|}$ to represent the green list associated with message m and prefix $x_{:t}$. The entries with value 1 correspond to green-list tokens, whose proportion in the vocabulary is controlled by a hyperparameter γ .

3.1.1 Watermark Injection

Multi-bit watermarking methods typically instantiate a message-dependent green list and red list at each step. Watermark injection is then implemented by adding a positive bias $\delta > 0$ to the logits of green-list tokens:

$$\log P'_M(v | a, x_{:t}) = \log P_M(v | a, x_{:t}) + \delta \cdot r_v(m, x_{:t})$$

Intuitively, the watermark increases the likelihood of generating message-specific green-list tokens while keeping the perturbation within a reasonable range to preserve text quality.

3.1.2 Watermark Detection

Suppose the message length is B bits. Then the candidate message space is $\Gamma = \{0, 1, \dots, 2^B - 1\}$. For each candidate message $m' \in \Gamma$, the watermark detector reconstructs the corresponding masks along the observed text x and computes a message-specific statistic. The most common detector is based on green-token counting:

$$\hat{m}(x) = \arg \max_{m' \in \Gamma} \sum_{t=0}^{|x|-1} r_{x_t}(m', x_{:t}). \quad (1)$$

Under this framework, watermarking reduces message attribution to a competition among candidate messages: the true message should induce the strongest statistical alignment between the observed tokens and its green lists.

Several variants enrich this pipeline without changing its overall structure. [Fernandez et al. \(2023\)](#) construct message-specific masks through cyclic shifts, while [Wang et al. \(2024\)](#) leverage a proxy language model to optimize green-list token assignments. Despite these differences, the common paradigm remains unchanged.

3.2 Segmented Multi-bit Watermarking

A central limitation of the conventional method is its exponential dependence on message length. Since detection evaluates all 2^B candidates over the whole sequence, the cost quickly becomes prohibitive when B grows. A standard remedy is *segmented multi-bit watermarking*. Let the binary representation of the message be $m = (m^0, m^1, \dots, m^{B-1})$. The message is divided into s segments, each containing B/s bits, where $s \mid B$. The i -th ($i \in \{0, 1, \dots, s-1\}$) segment is then defined as:

$$\text{seg}(m, i) = (m^{i \cdot \frac{B}{s}}, m^{i \cdot \frac{B}{s} + 1}, \dots, m^{(i+1) \cdot \frac{B}{s} - 1}).$$

When generating the t -th token, a hash function h_{seg} is applied to the preceding h tokens $x_{t-h:t}$ to determine which segment should be injected. Specifically, the message segment selected at this position is $\text{seg}(m, h_{\text{seg}}(x_{t-h:t}))$, and the watermark is injected using the conventional multi-bit watermarking scheme.

During detection, each segment is decoded independently. Let $\Gamma' = \{0, 1, \dots, 2^{B/s} - 1\}$ denote the candidate space for each segment. The i -th message segment is decoded as follows (the summation means counting the green tokens assigned to the i -th segment):

$$\text{seg}(\hat{m}, i) = \arg \max_{m' \in \Gamma'} \sum_{t, h_{\text{seg}}(x_{t-h:t})=i} r_{x_t}(m', x_{:t}). \quad (2)$$

The full message is then obtained by concatenating all decoded segments. This reduces the search space from 2^B to approximately $s \cdot 2^{B/s}$ candidates, which substantially improves efficiency.

4 Method

To systematically study the impact of segmentation strategies on the decoding accuracy of multi-bit watermarking, we first conduct theoretical analysis.

Let m denote the watermark message injected. Based on the randomness of the watermark mask $r(\cdot, x_{:t})$, the sum of the mask values of other messages in the watermarked text x can be approximately modeled as a binomial distribution. Here,

for simplicity of derivation, we assume that the mean mask value of each token is γ . A more precise treatment will be introduced in the debiased detector component of our method.

Let s be the number of segments and η be the proportion of tokens in x that lie in the green lists for message m (larger η indicates stronger watermarking). Assuming independence between the segments and different messages, the theoretical accuracy of the green-token counting method can be derived.

According to (2), the result is as follows:

$$\prod_{i=0}^{s-1} \prod_{m' \neq \text{seg}(m, i)} P\left(\sum_{t, h_{\text{seg}}(x_{t-h:t})=i} r_{x_t}(m', x_{:t})\right) < \sum_{t, h_{\text{seg}}(x_{t-h:t})=i} r_{x_t}(\text{seg}(m, i), x_{:t}). \quad (3)$$

In the above equation, the outer product represents the product of the probabilities of correct decoding for each segment, and the inner product represents the product of the probabilities of m being greater than the other messages in each segment.

Under uniform segmentation, a common setting in prior work, each segment contains roughly $|x|/s$ tokens. By the definition of η , the per-segment green token count $\sum_{t, h_{\text{seg}}(x_{t-h:t})=i} r_{x_t}(\text{seg}(m, i), x_{:t})$ for the true message m is approximately $\eta|x|/s$. Substituting this into the binomial model introduced above yields the following simplification of (3):

$$F_{\text{binom}}\left(\eta \frac{|x|}{s}; \frac{|x|}{s}, \gamma\right)^{s \cdot (2^{\frac{B}{s}} - 1)} \quad (4)$$

where $F_{\text{binom}}(\cdot; n, p)$ denotes the cumulative distribution function of Binomial(n, p).

We set $B = 20$, $|x| = 200$, and $\gamma = 0.5$, and plot the theoretical decoding accuracy in (4) as a function of η for $s \in \{1, 2, 4, 5, 10, 20\}$ in Figure 1. As the number of segments increases, the theoretical decoding accuracy rises more slowly with η . This is because a larger number of segments implies fewer tokens per segment, and the resulting attenuation of the watermark signal outweighs the noise-reduction effect induced by the smaller message space.

Similar trends hold across typical ranges of s , $|x|$, and B , and this trend can be further explained through an asymptotic analysis of (4). Specifically, we have the following result:

Proposition 1. Denote (4) as $A_{|x|}(s)$. Given $B > 0$, $1 \leq s < B$, and $0 < \gamma < \eta < 1$, there exists

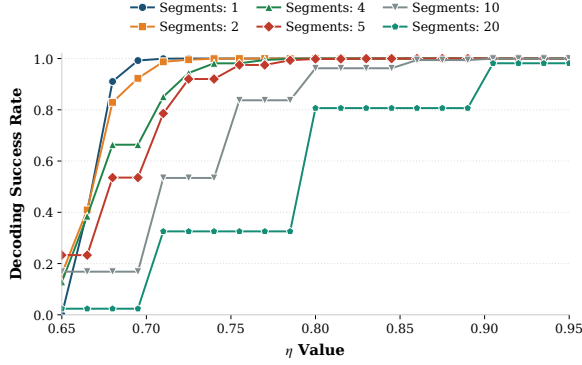


Figure 1: Theoretical decoding accuracy of the green-token counting method as a function of watermark strength η under different segment numbers s . ($B = 20, |x| = 200, \gamma = 0.5$)

a constant $X_0 > 0$, such that for any $|x| \geq X_0$, $A_{|x|}(s)$ is strictly decreasing with respect to s .

The proof is provided in Appendix A and Section 5.2 empirically validates this theoretical trend.

This result suggests that practical multi-bit watermarking systems should minimize the number of message segments, subject to computational constraints, in order to achieve a favorable trade-off between accuracy and efficiency. Since the detection complexity decreases exponentially with the number of segments, such a strategy remains computationally practical.

Beyond segmentation, the per-segment decoding scheme also limits detection accuracy. Existing methods exhibit message-dependent generation biases, allowing non-injected messages to occasionally achieve higher green-token counts than the true message m . To mitigate this effect, we normalize green-token statistics across candidate messages.

Let $G(x_{:t}, m) = \sum_{v \in V} P_M(v \mid a, x_{:t}) \cdot r_v(m, x_{:t})$ denote the probability that generated token x_t falls into the green list of message m given the unwatermarked model M . Since each mask value $r_{x_t}(m, x_{:t})$ is a Bernoulli random variable with expectation $G(x_{:t}, m)$, the total green-token count $\sum_t r_{x_t}(m, x_{:t})$ follows, under the assumption of positional independence and in the absence of watermarking, a distribution with mean $\sum_t G(x_{:t}, m)$ and variance $\sum_t G(x_{:t}, m)(1 - G(x_{:t}, m))$. We normalize the green-token count using its z-score to mitigate the imbalance of green-list distributions across different messages. Since the z-score measures the deviation of each message’s green-token statistic from its corresponding unwatermarked distribution, it enables a fairer comparison of watermark signals across messages.

In practice, we use a lightweight proxy model M' to estimate P_M , conditioned only on the already-generated tokens $x_{:t}$ (since the prompt a is unavailable at detection time). Thus, the estimated mean of $r_{x_t}(m, x_{:t})$ is:

$$\hat{G}(x_{:t}, m) = \sum_{v \in V} P_{M'}(v \mid x_{:t}) \cdot r_v(m, x_{:t}). \quad (5)$$

Therefore, the normalized score $\mathcal{Z}(x, m')$ is defined as:

$$\mathcal{Z}(x, m') = \frac{\sum_{t=0}^{|x|-1} (r_{x_t}(m', x_{:t}) - \hat{G}(x_{:t}, m'))}{\sqrt{\sum_{t=0}^{|x|-1} \hat{G}(x_{:t}, m')(1 - \hat{G}(x_{:t}, m'))}}. \quad (6)$$

Then the decoded message is $\hat{m}(x) = \arg \max_{m' \in \Gamma} \mathcal{Z}(x, m')$. Here we omit the notation of segmentation for simplicity, but applying this formulation to each segment is straightforward.

We now describe the implementation details of our method SDBMW, which combines the distribution-balanced detector above with the segmentation strategy proposed in this section.

Recall that the B -bit message m is partitioned into s sub-messages $\text{seg}(m, 0), \dots, \text{seg}(m, s-1)$, each containing B/s bits. A hash function h_{seg} assigns each token x_t to a segment index $i_t = h_{\text{seg}}(x_{t-h:t}) \in 0, \dots, s-1$. The mask value of x_t depends only on the sub-message $\text{seg}(m, i_t)$ associated with its assigned segment.

Concretely, the mask $r_{x_t}(m, x_{:t})$ is constructed as follows:

1. Hash the context $x_{:t}$ to generate a random seed, which is used to sample a permutation Perm_1 over V .
2. Hash the sub-message $\text{seg}(m, i_t)$ to generate another seed, which defines a second permutation Perm_2 over V .
3. Compose the two permutations as $\text{Perm} = \text{Perm}_2 \circ \text{Perm}_1$. The first $\gamma|V|$ tokens in Perm are assigned mask value 1, while the remaining tokens receive mask value 0.

During detection, each segment is decoded independently. Since each segment contains only $2^{B/s}$ candidate sub-messages, we precompute Perm_2 for all sub-messages and store the resulting Boolean mask vectors as rows of a shared matrix $\text{Mask}_{\text{msg}} \in \{0, 1\}^{2^{B/s} \times |V|}$, which is reused across

all segments. This matrix layout enables fully parallel GPU computation over all sub-message candidates, while the message-hash technique of (Wang et al., 2024) can be used to further reduce memory consumption.

For normalized detection, we compute the z-score in (6) for all candidate sub-messages within each segment and select the sub-message with the maximum z-score. The decoded sub-messages from all segments are then concatenated to form the final message. This procedure can also be efficiently parallelized using Mask_{msg} .

5 Experiments

We evaluate the proposed method around two questions. First, we evaluate whether SDBMW improves detection accuracy and robustness. Section 5.1 answers this question through comparison against representative multi-bit watermarking baselines and a controlled decoder ablation. Second, we ask whether the segmentation trade-off analyzed in Section 4 appears in real generations. Section 5.2 validates this trend across a range of message lengths, text lengths, and error-correction code settings.

5.1 Evaluating SDBMW

The main comparison evaluates exact 20-bit message recovery from both original and edited generations, while also assessing the impact of watermarking on generation quality. The decoder ablation isolates the contribution of distribution-balanced normalization by applying different decoding strategies to the same generated texts. To further demonstrate the transferability of our detector, we apply it to a recent baseline and observe substantial improvements. We use exact message accuracy as the primary metric because the target applications require recovery of the entire payload. Additional results for longer messages, varying text lengths, proxy-model sensitivity and LLM-based paraphrasing attacks are provided in the appendix.

Experimental setup. We evaluate 20-bit random messages on the C4 news subset (Raffel et al., 2020; Dodge et al., 2021) and the Essays (Schuhmann, 2023) dataset, targeting 500 samples from each dataset. The models used are Llama-3.1-8B (Llama Team, 2024) and Qwen-3-8B (Qwen Team, 2025). Methods requiring proxy language models use Llama-3.2-1B and Qwen-3-1.7B, re-

spectively. Appendix B provides the full configuration.

Baselines. We compare SDBMW with CS (Fernandez et al., 2023), MPAC (Yoo et al., 2024), CTWL (Wang et al., 2024), RSBH (Qu et al., 2025), StealthInk (Jiang et al., 2025), and XMark (Xu et al., 2026b). These baselines cover full-message enumeration, position allocation, proxy-balanced green lists, Reed–Solomon-based error-correction, distribution-preserving sampling and multi-shard vocabulary partitioning.

SDBMW consistently outperforms baselines in decoding accuracy. Table 1 shows that SDBMW remains competitive with the strongest baseline on clean generations and gives clearer gains after editing attacks. In the no-attack setting, SDBMW is close to CTWL. Considering the baseline already reaches high accuracy in this setting, there is limited room for improvement. The accuracy advantage becomes more pronounced under synonym replacement, where SDBMW improves over the strongest baseline by 6.8 and 14.4 percentage points on Llama-3.1-8B, and by 11.4 and 34.2 points on Qwen-3-8B. Under truncation attack, SDBMW also shows consistent gains except for a near tie on Qwen-3-8B and Essays, where the strongest baseline already achieves 0.996 accuracy.

The comparison also reveals the cost of aggressive segmentation. MPAC and RSBH obtain substantially lower exact message accuracy than CS, CTWL, and SDBMW in this evaluation, especially under synonym replacement. This behavior is consistent with the theoretical analysis in Section 4: when a message is divided into many segments, each local decoder receives fewer tokens, and its segment-level statistic becomes unstable under sampling noise or text edits. This instability outweighs the benefit of fewer candidate messages in each segment. SDBMW therefore uses segmentation conservatively and relies on decoding-stage distribution normalization to improve candidate comparison without excessively weakening the per-segment signal. Appendix B reports the method-specific parameters to make this comparison transparent. Specifically, MPAC obtains extremely low accuracy in this experiment because the models used here have relatively low entropy, which is often observed in stronger instruction-tuned models.

Generation Quality. Following the LLM-as-a-judge paradigm (Zheng et al., 2023),

Table 1: Exact message accuracy for 20-bit watermark messages on C4 and Essays. Higher values are better. Synonym replacement changes 25% of words, and truncation removes the first 100 words. Best or tied-best results in each row are bolded.

Model	Dataset	Attack	CS	MPAC	CTWL	RSBH	StealthInk	XMark	SDBMW
Llama-3.1-8B	C4	None	0.880	0.002	0.986	0.782	0.042	0.874	0.988
		Synonym	0.692	0.000	0.870	0.388	0.000	0.184	0.938
		Truncation	0.716	0.002	0.948	0.592	0.000	0.682	0.962
	Essays	None	0.772	0.000	0.960	0.530	0.018	0.844	0.982
		Synonym	0.455	0.000	0.694	0.008	0.000	0.142	0.838
		Truncation	0.570	0.000	0.868	0.294	0.000	0.594	0.942
Qwen-3-8B	C4	None	0.906	0.002	0.996	0.842	0.034	0.742	1.000
		Synonym	0.734	0.002	0.842	0.424	0.002	0.110	0.956
		Truncation	0.789	0.002	0.960	0.608	0.006	0.436	0.992
	Essays	None	0.874	0.002	1.000	0.618	0.002	0.506	1.000
		Synonym	0.524	0.002	0.612	0.136	0.000	0.036	0.954
		Truncation	0.770	0.004	0.996	0.383	0.002	0.174	1.000

Table 2: Generation quality for seven methods on C4. DeepSeek V4 Flash reports the preference for watermarked over paired unwatermarked text (a tie counts as 0.5); BERTScore reports F1 against the paired unwatermarked continuation. Higher values indicate better generation quality. The judge produced 99 valid judgments for SDBMW with Llama after one provider rejection; all other Llama and Qwen cells contain 100 and 200 judgments, respectively.

Method	DeepSeek preference		BERTScore F1	
	Llama	Qwen	Llama	Qwen
CS	0.1800	0.2050	0.8248	0.8198
MPAC	0.2300	0.2050	0.8307	0.8239
CTWL	0.2750	0.2650	0.8386	0.8315
RSBH	0.2250	0.2225	0.8259	0.8240
SDBMW	0.2879	0.2625	0.8340	0.8295
XMark	0.3950	0.3900	0.8367	0.8346
StealthInk	0.4500	0.4400	0.8438	0.8385

we conduct a pairwise judging experiment on the C4 news subset with DeepSeek V4 Flash (deepseek-v4-flash-0731) (Alibaba Cloud, 2026) and measure semantic similarity to paired unwatermarked continuations using BERTScore (Zhang et al., 2020). Appendix B.6 provides the full judging protocol and experiment settings.

Table 2 shows that XMark and StealthInk, two recent methods that explicitly prioritize generation quality, deliver the strongest overall quality results. Apart from these two quality-oriented methods, SDBMW obtains the highest DeepSeek

preference on Llama-3.1-8B and nearly matches CTWL on Qwen-3-8B (0.2625 versus 0.2650). Its BERTScore F1 is also close to CTWL, differing by only 0.0046 and 0.0020 on Llama and Qwen, respectively. Thus, SDBMW provides stronger payload recovery without substantially compromising generation quality.

Varying message bit-widths and text lengths.

To accommodate practical applications requiring larger payloads, we additionally evaluate 32- and 48-bit messages, using the message-hashing technique of Wang et al. (2024) within each SDBMW segment to reduce memory overhead. SDBMW achieves exact decoding accuracies of 0.97 and 0.93, respectively, exceeding the strongest non-SDBMW baseline, Segmented CS, by 0.37 at 48 bits (Appendix B.2). Moreover, it consistently achieves 0.98 accuracy across sequence lengths of 200–400 tokens, demonstrating limited sensitivity to text length within the evaluated range (Appendix B.2).

Robustness to LLM paraphrasing. We additionally evaluate the seven methods on the same 200 C4 samples after paraphrasing with Llama-3.1-8B, with every attacked text truncated to exactly 200 tokens to ensure fair comparison and enhance attack intensity. SDBMW achieves 0.935 exact payload accuracy and 0.9755 bit accuracy, compared with 0.870 and 0.9310 for the strongest baseline, CTWL; the complete results are reported in Appendix Table 6.

Table 3: Normalization ablation with $\delta = 2$. All detectors decode the same 500 texts attacked by synonym replacement.

Detector	Bit Acc.	Msg Acc.	Correct	Gain
Raw count	0.8936	0.654	327/500	–
Centered	0.9453	0.816	408/500	+81
Z-score	0.9470	0.820	410/500	+83

Decoder ablation setup. We next validate whether distribution-balanced normalization is effective. The ablation fixes the generated texts, attacked texts, and embedded messages, and changes only the candidate scoring statistic. In contrast to the previous two experiments, we set $\delta = 2$, weaker than the main robustness setting, to make candidate-message bias easier to observe. The raw-count detector ranks candidates by observed green-token count; the centered detector subtracts the proxy-estimated expected count; and the z-score detector additionally divides by the proxy-estimated standard deviation.

Distribution-balanced normalization reduces decoding error. Table 3 shows that centering the score removes most of the decoding error caused by message-dependent green-token bias. Compared with raw count, centered scoring raises bit accuracy from 0.8936 to 0.9453 and exact message accuracy from 0.654 to 0.816, corresponding to 81 additional correctly recovered messages. Z-score normalization gives slightly better results, with 0.9470 bit accuracy and 0.820 exact message accuracy. Its gain over centered scoring is smaller, indicating that mean correction is the dominant factor. A secondary check at $\delta = 3$ showing the same trend is provided in Appendix Table 15.

Transferability of our detector. As our decoder is a relatively self-contained component, it can be readily applied to other multi-bit watermarking methods based on token-level perturbations. To validate its transferability, we apply it to the recent XMark baseline (Appendix B.4). Our decoder improves exact decoding accuracy by 0.092 without attacks and by 0.106–0.182 under truncation, synonym-substitution, and LLM-paraphrasing attacks.

Proxy-model sensitivity and computational overhead. Our method requires a proxy model with the same vocabulary and a similar output distribution as the target model. This requirement is prac-

tical for model service providers, who have full model access and can distill a smaller proxy. Experiments in Appendix B.3 show limited sensitivity to proxy-model scale. Moreover, Appendix B.7 shows that SDBMW has modest decoding overhead and is $7.86\text{--}8.26\times$ faster than CTWL, which uses the same proxy model in our setting.

Zero-Bit Detection. In addition to multi-bit payload recovery, it is also important to distinguish watermarked text from human-written text. Thus, a multi-bit watermarking method should also support zero-bit detection. Along with the main comparison experiment, we also conduct zero-bit detection and report the results in Table 11 in the Appendix. For our method SDBMW, we use the sum of maximum normalized scores of each segment as the detection indicator. For baselines, we use the zero-bit detection methods mentioned in their original papers or provided in their official code repositories. We report the true positive rate at a 1% false positive rate. SDBMW achieves the best performance in 8 out of 12 settings and remains close to the strongest baseline in the remaining settings.

5.2 Validating the Segmentation Trade-off

This subsection validates the theoretical claim in Section 4. Segmenting the message reduces the local candidate space, but it also weakens the available token evidence across segments. If the analysis in Section 4 captures the dominant effect, then heavily segmented watermarking should incur accuracy loss. We verify this claim across various message lengths, text lengths, and error-correction code variants.

Controlled segmentation sweep. To isolate segmentation from SDBMW normalization, we run the cyclic-shift CS baseline (Fernandez et al., 2023) in the open-source RSBH codebase². The sweep uses Llama-3.1-8B on C4, no attack, exact message accuracy, 200-token generations, $\gamma = 0.5$, and $\delta \in \{1.0, 1.5, \dots, 4.5\}$. We evaluate $s \in \{1, 2, 4, 5, 10, 20\}$ for 20-bit payloads, $s \in \{2, 4, 8, 16, 32\}$ for 32-bit payloads, and $s \in \{2, 4, 5, 8, 10, 20, 40\}$ for 40-bit payloads. The x-axis in Figure 2 reports the empirical green-token ratio η measured from the generated texts; the full sweep configuration appears in Appendix C.

²<https://github.com/randomizedtree/segment-watermark>

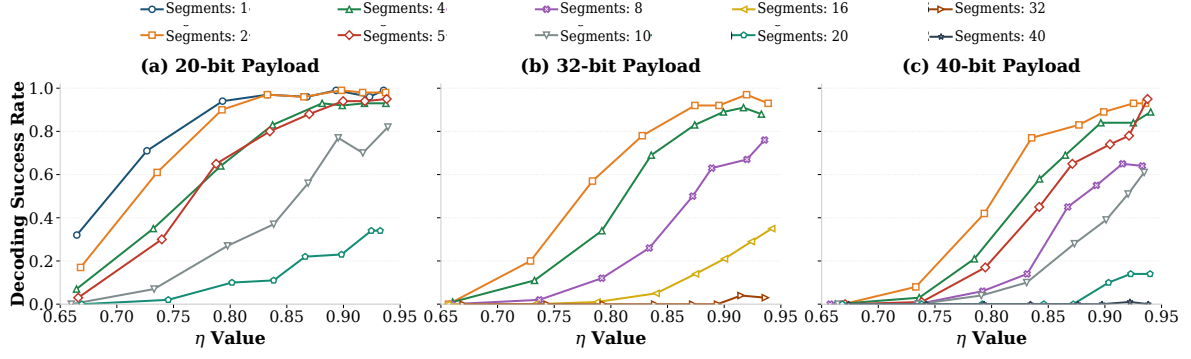


Figure 2: Empirical segmentation ablation under the no-attack setting for three payload lengths. All panels use 200-token generations, sweep δ from 1.0 to 4.5, and report exact decoding success against the resulting empirical watermark strength η .

Real generations reproduce the predicted ordering. Figure 2 shows the same ordering predicted by the theoretical curve in Figure 1. With a fixed text length, smaller segment numbers reach high exact decoding success at lower empirical watermark strength, whereas larger segment numbers require substantially stronger evidence. In the 20-bit setting, the $s = 1$ and $s = 2$ curves rise much earlier than the $s = 10$ and $s = 20$ curves. The same pattern persists for 32-bit and 40-bit payloads. This repetition across payload sizes indicates that the observed degradation is a finite-sample consequence of distributing the available evidence across many local decisions.

Longer text alleviates but does not eliminate over-segmentation. To examine whether 200 tokens provide insufficient evidence for segmented decoding, we vary the generation length from 200 to 400 tokens at $\delta = 2.0$ using the balanced RSBH mapping. As shown in Appendix Figure 3, longer texts improve exact message accuracy across all segment counts, but configurations with more segments remain less accurate and require more tokens to achieve comparable performance. Thus, additional text alleviates rather than eliminates the evidence loss caused by aggressive segmentation.

Error correction is constrained by the same evidence budget. We also evaluate whether Reed–Solomon error correction code can compensate for the loss of per-segment evidence. Appendix Figure 4 compares the 20-bit segmentation sweep with payload-preserving ECC variants, including RS(4, 2), RS(6, 4), and RS(7, 5). ECC provides tolerance for decoding error through redundancy, but also increases the number of encoded segments from k to n , resulting in weaker segment-level wa-

termark evidence. Consequently, ECC is not a free remedy: under weaker watermark strengths, the signal loss from additional encoded segments can offset the redundancy benefit.

Implication for SDBMW. These results support conservative segmentation. While segmentation enables scalable decoding, excessive segmentation weakens the evidence available to each local decoder. SDBMW therefore uses the fewest segments permitted by the computational budget.

6 Conclusion

In this paper, we revisit segmentation in multi-bit watermarking for LLM-generated text and show that segmentation, while essential for scalable decoding, introduces a trade-off between efficiency and accuracy. Through theoretical analysis and empirical validation, we reveal this under-characterized drawback and advocate conservative segmentation under computational constraints. We further introduce a proxy-model-based normalized detector to mitigate message-dependent bias during decoding. Experiments on multiple datasets and models show that SDBMW improves decoding accuracy and robustness over representative multi-bit watermarking baselines.

Limitations

Multi-bit watermarking methods typically require a preceding zero-bit verification step, which determines whether a given text carries a watermark before attempting payload recovery. Although this paper provides theoretical analysis and empirical validation of the relationship between segmentation strategies and multi-bit decoding accuracy, we do not further analyze how segmentation affects

zero-bit detection accuracy. This choice is mainly motivated by the fact that exact multi-bit decoding is a more challenging and critical requirement in payload-recovery scenarios. We leave a systematic study of the impact of segmentation on zero-bit detection to our future work.

In addition, the datasets and models used in this paper can be further broadened. The current experiments are conducted on English datasets, and the evaluated models are no larger than 8B parameters. We adopt this setting to maintain consistency with prior works and to enable controlled comparisons. In future work, we expect to further validate the effectiveness of our method on datasets in more languages and on larger-scale language models.

Acknowledgements

This work was supported by the Beijing Natural Science Foundation (4252023), and the National Natural Science Foundation of China under Grant (No. 62472409).

References

- Alibaba Cloud. 2026. [DeepSeek-V4-Flash-0731 model information](#). Alibaba Cloud Model Studio documentation. Released August 1, 2026; accessed August 26, 2026.
- Árpád Baricz. 2008. [Mills’ ratio: Monotonicity patterns and functional inequalities](#). *Journal of Mathematical Analysis and Applications*, 340(2):1362–1370.
- Chaithanya Reddy Bogadi. 2025. [The rise of foundation models in industry: A cross domain survey of LLM applications in healthcare, finance, legal, and education](#). *International Journal of Computer Trends and Technology*, 73(7):1–15.
- Yapei Chang, Kalpesh Krishna, Amir Houmansadr, John Frederick Wieting, and Mohit Iyyer. 2024. [Post-Mark: A robust blackbox watermark for large language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8969–8987, Miami, Florida, USA. Association for Computational Linguistics.
- Jiaqi Chen, Xiaoye Zhu, Tianyang Liu, Ying Chen, Xinhui Chen, Yiwen Yuan, Chak Tou Leong, Zuchao Li, Long Tang, Lei Zhang, Chenyu Yan, Guanghao Mei, Jie Zhang, and Lefei Zhang. 2025. [Imitate before detect: Aligning machine stylistic preference for machine-revised text detection](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(22):23559–23567.
- Aloni Cohen, Alexander Hoover, and Gabe Schoenbach. 2025. [Watermarking language models for many adaptive users](#). In *2025 IEEE Symposium on Security and Privacy (SP)*, pages 2583–2601. IEEE.
- Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Posen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, and 5 others. 2024. [Scalable watermarking for identifying large language model outputs](#). *Nature*, 634(8035):818–823.
- Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. 2021. [Documenting large webtext corpora: A case study on the colossal clean crawled corpus](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1286–1305. Association for Computational Linguistics.
- Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023. [Three bricks to consolidate watermarks for large language models](#). In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6.
- Abhimanyu Hans, Avi Schwarzschild, Valeriia Cherepanova, Hamid Kazemi, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2024. [Spotting LLMs with binoculars: Zero-shot detection of machine-generated text](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 17519–17537. PMLR.
- Beizhe Hu, Qiang Sheng, Juan Cao, Yang Li, and Danding Wang. 2025. [LLM-generated fake news induces truth decay in news ecosystem: A case study on neural news recommendation](#). In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR ’25*, pages 435–445, New York, NY, USA. Association for Computing Machinery.
- Ya Jiang, Chuxiong Wu, Massieh Kordi Boroujeny, Brian Mark, and Kai Zeng. 2025. [StealthInk: A multi-bit and stealthy watermark for large language models](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 27685–27709. PMLR.
- Nikola Jovanović, Robin Staab, and Martin Vechev. 2024. [Watermark stealing in large language models](#). In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 22570–22593. PMLR.
- John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. [A watermark for large language models](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

- John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2024. [On the reliability of watermarks for large language models](#). In *The Twelfth International Conference on Learning Representations*.
- Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. [Robust distortion-free watermarks for language models](#). *Transactions on Machine Learning Research*.
- Llama Team. 2024. [The Llama 3 herd of models](#). Preprint, arXiv:2407.21783.
- Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D. Manning, and Chelsea Finn. 2023. [DetectGPT: Zero-shot machine-generated text detection using probability curvature](#). In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 24950–24962. PMLR.
- Shinwoo Park, Hyejin Park, Hyeseon Ahn, and Yo-Sub Han. 2026. [WaterMod: Modular token-rank partitioning for probability-balanced LLM watermarking](#). *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(39):32683–32691.
- Wenjie Qu, Wengruai Zheng, Tianyang Tao, Dong Yin, Yanze Jiang, Zhihua Tian, Wei Zou, Jinyuan Jia, and Jiaheng Zhang. 2025. [Provably robust multi-bit watermarking for AI-Generated text](#). In *34th USENIX Security Symposium (USENIX Security 25)*, pages 201–220, Seattle, WA. USENIX Association.
- Qwen Team. 2025. [Qwen3 technical report](#). Preprint, arXiv:2505.09388.
- Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. [Exploring the limits of transfer learning with a unified text-to-text transformer](#). *Journal of Machine Learning Research*, 21(140):1–67.
- Christoph Schuhmann. 2023. [Essays with instructions](#). Hugging Face dataset. Accessed: 2026-04-28.
- Chenrui Wang, Junyi Shu, Billy Chiu, Yu Li, Saleh Alharbi, Min Zhang, and Jing Li. 2025. [Learning to watermark: A selective watermarking framework for large language models via multi-objective optimization](#). In *Advances in Neural Information Processing Systems*, volume 38.
- Dandan Wang and Shiqing Zhang. 2024. [Large language models in medical and healthcare fields: Applications, advances, and challenges](#). *Artificial Intelligence Review*, 57:299.
- Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2024. [Towards codable watermarking for injecting multi-bits information to llms](#). In *The Twelfth International Conference on Learning Representations*.
- Jiahao Xu, Rui Hu, Olivera Kotevska, and Zikai Zhang. 2026a. [Majority bit-aware watermarking for large language models](#). Preprint, arXiv:2508.03829.
- Jiahao Xu, Rui Hu, Olivera Kotevska, and Zikai Zhang. 2026b. [XMark: Reliable multi-bit watermarking for LLM-generated texts](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14747–14763, San Diego, California, United States. Association for Computational Linguistics.
- KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. 2024. [Advancing beyond identification: Multi-bit watermark for large language models via position allocation](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4031–4055, Mexico City, Mexico. Association for Computational Linguistics.
- Or Zamir. 2024. [Undetectable steganography for language models](#). *Transactions on Machine Learning Research*.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. [BERTScore: Evaluating text generation with BERT](#). In *International Conference on Learning Representations*.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-judge with MT-bench and chatbot arena](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

A Proof of the Theoretical Analysis

Proof. By the central limit theorem, as $|x| \rightarrow \infty$,

$$F_{\text{binom}}(\eta n; n, \gamma) = \Phi(z_s) + o(1),$$

$$z_s = \frac{(\eta - \gamma)\sqrt{n}}{\sigma} = \frac{\eta - \gamma}{\sqrt{\gamma(1 - \gamma)}} \sqrt{\frac{|x|}{s}},$$

where Φ denotes the cumulative distribution function of the standard normal distribution, and ϕ denotes its probability density function:

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}.$$

Since $\eta > \gamma$, we have $z_s \rightarrow +\infty$ as $|x| \rightarrow \infty$. By Mills' ratio (Baricz, 2008),

$$1 - \Phi(z_s) \sim \frac{\phi(z_s)}{z_s}.$$

Hence, the binomial cumulative distribution function can be written as

$$F_{\text{binom}}(\eta n; n, \gamma) = 1 - u_s,$$

$$u_s \sim \frac{\phi(z_s)}{z_s}.$$

Define

$$K(s) = s(2^{B/s} - 1).$$

Then

$$\log A_{|x|}(s) = K(s) \log(1 - u_s).$$

Next, we expand $\log(1 - u_s)$. By the Taylor expansion,

$$\log(1 - u) = -u + O(u^2), \quad u \rightarrow 0.$$

Since $u_s \rightarrow 0$,

$$\log(1 - u_s) = -u_s + O(u_s^2) \sim -u_s.$$

Substituting this into the previous expression yields

$$\log A_{|x|}(s) = K(s) \log(1 - u_s) \sim -K(s)u_s$$

$$\sim -K(s) \frac{\phi(z_s)}{z_s}.$$

We now derive an explicit asymptotic expression for $\phi(z_s)/z_s$. From

$$z_s = \frac{\eta - \gamma}{\sqrt{\gamma(1 - \gamma)}} \sqrt{\frac{|x|}{s}},$$

we obtain

$$z_s^2 = \frac{(\eta - \gamma)^2}{\gamma(1 - \gamma)} \frac{|x|}{s}.$$

Therefore,

$$\phi(z_s) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{z_s^2}{2}\right)$$

$$= \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(\eta - \gamma)^2}{2\gamma(1 - \gamma)} \frac{|x|}{s}\right),$$

and consequently,

$$\frac{\phi(z_s)}{z_s} = \frac{\sqrt{\gamma(1 - \gamma)}}{(\eta - \gamma)\sqrt{2\pi}} \sqrt{\frac{s}{|x|}} \exp\left(-\frac{(\eta - \gamma)^2}{2\gamma(1 - \gamma)} \frac{|x|}{s}\right).$$

Define

$$C = \frac{\sqrt{\gamma(1 - \gamma)}}{(\eta - \gamma)\sqrt{2\pi}}, \quad \lambda = \frac{(\eta - \gamma)^2}{2\gamma(1 - \gamma)}.$$

Then

$$\log A_{|x|}(s) \sim -C|x|^{-1/2} G_{|x|}(s),$$

where

$$G_{|x|}(s) = K(s) \sqrt{s} \exp\left(-\lambda \frac{|x|}{s}\right).$$

We now analyze the monotonicity of the dominant term $-C|x|^{-1/2} G_{|x|}(s)$ with respect to s . Since $C > 0$ and $|x|^{-1/2} > 0$, the prefactor is independent of s . Therefore, the monotonicity of $\log A_{|x|}(s)$ is determined by $-G_{|x|}(s)$. As shown in the subsequent derivations, the higher-order terms do not alter the strict monotonicity of the leading term. In particular, if $G_{|x|}(s)$ is strictly increasing in s , then $-G_{|x|}(s)$ is strictly decreasing, implying that $\log A_{|x|}(s)$ is also strictly decreasing.

To analyze $G_{|x|}(s)$, consider

$$\log G_{|x|}(s) = \log K(s) + \frac{1}{2} \log s - \lambda \frac{|x|}{s}.$$

Differentiating with respect to s gives

$$\frac{d}{ds} \log G_{|x|}(s) = \frac{K'(s)}{K(s)} + \frac{1}{2s} + \lambda \frac{|x|}{s^2}.$$

We next show that $\frac{K'(s)}{K(s)}$ is uniformly bounded on $[1, B)$. First,

$$K'(s) = 2^{B/s} \left(1 - \frac{B \ln 2}{s}\right) - 1.$$

Let

$$y = \frac{B}{s}, \quad h(y) = 2^y (1 - y \ln 2) - 1.$$

Then, for all $s \in [1, B)$, we have $y \geq 1$ and

$$K'(s) = h(y).$$

Differentiating $h(y)$ yields

$$h'(y) = -y(\ln 2)^2 2^y < 0.$$

Thus, $h(y)$ is strictly decreasing on $[1, +\infty)$, and hence

$$K'(s) = h(y) \leq h(1) = 2(1 - \ln 2) - 1 < 0.$$

Therefore, $K(s)$ is strictly decreasing on $[1, B)$.

On the other hand, by continuity and monotonicity,

$$K(s) \geq \lim_{s \rightarrow B^-} K(s) = B > 0.$$

Hence, $K(s)$ admits a positive lower bound on $[1, B)$. Since $K'(s)$ is continuous on this interval, there exists a constant $M_1 > 0$ such that

$$|K'(s)| \leq M_1, \quad 1 \leq s < B.$$

Combining this with $K(s) \geq B$, there exists a constant $M > 0$ such that

$$\left| \frac{K'(s)}{K(s)} \right| \leq M, \quad 1 \leq s < B.$$

Consequently,

$$\frac{d}{ds} \log G_{|x|}(s) \geq -M + \lambda \frac{|x|}{B^2}.$$

Therefore, there exists a constant $X_0 > 0$ such that whenever $|x| \geq X_0$,

$$\frac{d}{ds} \log G_{|x|}(s) > 0.$$

Hence, $G_{|x|}(s)$ is strictly increasing with respect to s , which implies that

$$-C|x|^{-1/2}G_{|x|}(s)$$

is strictly decreasing in s . Since

$$\log A_{|x|}(s) \sim -C|x|^{-1/2}G_{|x|}(s),$$

it follows that $\log A_{|x|}(s)$ is strictly decreasing in s for sufficiently large $|x|$.

Finally, because the logarithm function is strictly increasing, $A_{|x|}(s)$ and $\log A_{|x|}(s)$ share the same monotonicity. Therefore, $A_{|x|}(s)$ is strictly decreasing with respect to s . $\square$

B Main Evaluation Experiment

Table 4 summarizes the shared protocol for the main 20-bit robustness evaluation. Table 5 lists the method-specific watermark and decoding parameters used in the comparison. These details are included for the reproducibility of the results in Table 1.

Table 4: Shared configuration for the main 20-bit robustness evaluation. Exact message accuracy uses successfully decoded payloads (n_{payload}), whereas zero-bit TPR uses watermarked scores that pass the minimum-token threshold (n_{wm}).

Item	Configuration
Datasets	C4 news and Essays; target 500 generations per method, dataset, and model. Actual n_{payload} values are listed below.
Prompts	C4 uses the first two sentences; Essays uses the writing instruction field.
Base LMs	Llama-3.1-8B and Qwen-3-8B.
Proxy LMs	Llama-3.2-1B for Llama-3.1-8B; Qwen-3-1.7B for Qwen-3-8B.
Precision	Base and proxy models are loaded in bfloat16.
Payload	Random 20-bit message; payload space size $2^{20} = 1,048,576$.
Sampling	Temperature 0.7, top- $p = 0.9$, repetition penalty 1.05, batch size 1.
Length	Maximum generation length 400 tokens; the zero-bit analysis uses a 250-token minimum.
Random seed	12 for the main robustness runner.
Attacks	None; substitution_0.25; cropping_100.
Metrics	Exact message accuracy for Table 1; bit accuracy and AUC are logged by the runner when applicable.

Actual exact-recovery denominators. All unlisted cells in Table 1 use $n_{\text{payload}} = 500$. Lower counts arise from decoder-stage skips (including word_count_lt_20).

Method	Model	Dataset	Attack	n_{payload}
CS	Llama-3.1-8B	Essays	None	499
CS	Llama-3.1-8B	Essays	Synonym	499
CS	Llama-3.1-8B	Essays	Truncation	498
CS	Qwen-3-8B	C4	Truncation	498
RSBH	Llama-3.1-8B	C4	Truncation	497
RSBH	Llama-3.1-8B	Essays	None	498
RSBH	Qwen-3-8B	Essays	Truncation	499

B.1 Robustness to LLM Paraphrasing

We evaluate seven methods on aligned C4 source IDs 0–199 after paraphrasing with Llama-3.1-8B. The paraphraser rewrites each full completion with greedy decoding, and every attacked completion contains exactly 200 tokens under the shared Llama tokenizer. Thus, all methods use the same source samples and attacked-text length.

B.2 Scaling Beyond the Main Setting

Larger payloads. We evaluate clean recovery beyond the 20-bit main setting on the same 100 C4

source samples using Llama-3.1-8B, seed 0, and 200 generated tokens. SDBMW and Segmented CS use 16 bits per segment, so the 32- and 48-bit payloads use two and three segments, respectively; XMark uses its native multi-shard layout. Table 7 shows that SDBMW-z achieves 0.970 exact recovery at 32 bits and retains 0.930 at 48 bits, whereas XMark reaches 0.320 and 0.060.

Text-length sweep. We further evaluate SDBMW over 200–400 tokens using the same 200 C4 source IDs and payloads at every length. Each generated text is truncated to the target length with the shared Llama tokenizer. As shown in Table 8, exact payload accuracy remains 0.980 throughout the sweep, while bit accuracy ranges from 0.9912 to 0.9928. This complements the segmentation study by directly confirming the stability of the complete SDBMW method over the tested length range.

B.3 Proxy-Model Sensitivity

We isolate proxy scale by decoding the same 500 Llama-3.1-8B C4 generations and payloads with either the tokenizer-compatible Llama-3.2-1B proxy used in the main experiments or the full Llama-3.1-

Table 5: Method-specific configuration for the 20-bit robustness comparison. All methods share the data, models, sampling, and attacks in Table 4.

Method	Message layout	Key watermark parameters	Decoder statistic
CS	Full-message cyclic shift	$\gamma = 0.5$, $\delta = 3$, prefix length 1, payload space 2^{20}	Green-token count over all 20-bit candidates
MPAC	Position allocation	$\gamma = 0.25$, $\delta = 3$, base 4, message length 20	MPAC score
CTWL	Full-message proxy-balanced green lists	$\gamma = 0.5$, $\delta = 3$, proxy prefix 10, 100 random permutations, top- $k = 1000$, encode ratio 20	Confidence score
RSBH	Balanced segment assignment with RS code	$\gamma = 0.5$, $\delta = 3$, RS(6, 4) over $\text{GF}(2^5)$ for 20-bit payloads	Segment-wise green-token count with RS decoding
StealthInk	Ten 2-bit symbols	Chunk capacity 2 and message length 20; native probability reweighting	Native symbol-wise detector
XMark	Ten 2-bit symbols with multi-shard partitioning	$\delta = 3$, hash key 15485863, message length 20	Native XMark shard score
SDBMW	Two 10-bit segments	$\gamma = 0.5$, $\delta = 3$, proxy initialization length 20, hash key 15485863	Z-score normalized

Table 6: Recovery after fixed-length LLM paraphrasing. All methods use the same 200 C4 samples and exactly 200 attacked tokens. Exact-accuracy brackets give Wilson 95% confidence intervals. Higher values are better.

Method	Exact Acc. (95% CI)	Bit Acc.
CS	0.000 [0.000, 0.019]	0.5033
MPAC	0.010 [0.003, 0.036]	0.5838
CTWL	0.870 [0.816, 0.910]	0.9310
RSBH	0.335 [0.273, 0.403]	0.7828
StealthInk	0.010 [0.003, 0.036]	0.6938
XMark	0.410 [0.344, 0.479]	0.9085
SDBMW	0.935 [0.892, 0.962]	0.9755

8B model as the proxy. Table 9 shows identical clean recovery and only a 0.002 difference in exact payload accuracy under synonym replacement. Thus, within this controlled comparison, increasing the proxy from 1B to 8B does not improve recovery; this result does not imply insensitivity to arbitrary proxy architectures or incompatible tokenizers.

B.4 Transferring Normalization to XMark

We test whether the normalized detector transfers to a different segmentation construction by ap-

Table 7: Clean recovery for 32- and 48-bit payloads on C4. All settings use 100 Llama-3.1-8B generations of 200 tokens. Exact-accuracy brackets give Wilson 95% confidence intervals. Higher values are better.

Method	Bits	s	Exact Acc. (95% CI)	Bit Acc.
Segmented CS	32	2	0.940 [0.875, 0.972]	0.9847
XMark	32	–	0.320 [0.237, 0.417]	0.9366
SDBMW-raw	32	2	0.940 [0.875, 0.972]	0.9756
SDBMW-z	32	2	0.970 [0.915, 0.990]	0.9881
Segmented CS	48	3	0.560 [0.462, 0.653]	0.8979
XMark	48	–	0.060 [0.028, 0.125]	0.8821
SDBMW-raw	48	3	0.640 [0.542, 0.727]	0.9046
SDBMW-z	48	3	0.930 [0.863, 0.966]	0.9848

Table 8: SDBMW clean recovery over the 200–400-token range on the same 200 C4 samples and payloads. Higher values are better.

Tokens	Payload Acc.	Bit Acc.
200	0.980	0.9912
250	0.980	0.9915
300	0.980	0.9928
350	0.980	0.9928
400	0.980	0.9928

Table 9: Proxy-scale sensitivity on fixed Llama-3.1-8B C4 generations. Both proxy conditions decode the same 500 texts and payloads. Higher values are better.

Setting	Proxy	n	Payload Acc.	Bit Acc.
Clean	Llama-3.2-1B	500	0.988	0.9959
Clean	Llama-3.1-8B	500	0.988	0.9959
Synonym	Llama-3.2-1B	500	0.938	0.9799
Synonym	Llama-3.1-8B	500	0.936	0.9784

Table 10: Transfer of z-score normalization to fixed XMark generations on C4 ($n = 500$). Raw and z-score variants decode identical texts and payloads. Higher values are better.

Attack	Payload Acc.		Bit Acc.	
	XMark-raw	XMark-z	XMark-raw	XMark-z
None	0.874	0.966	0.983	0.993
Synonym	0.184	0.290	0.876	0.904
Truncation	0.682	0.864	0.954	0.975
LLM paraphrase	0.064	0.216	0.817	0.872

plying it to XMark. XMark-raw reproduces the original method, whereas XMark-z replaces only its candidate-scoring statistic with proxy-based z-score normalization. Both variants decode the same 500 Llama-3.1-8B C4 texts and 20-bit payloads in each setting, using Llama-3.2-1B as the proxy. Table 10 shows consistent gains without modifying XMark’s generation procedure: exact payload accuracy improves by 0.092 on clean text, 0.106 under synonym replacement, 0.182 under truncation, and 0.152 under LLM paraphrasing.

B.5 Zero-Bit Detection Results

The zero-bit detection evaluation adopts the same settings as the main comparison experiment in Section 5.1. The results are provided in Table 11.

B.6 Quality Evaluation and Decoder-Ablation

Table 2 in the main text reports quality-judging and BERTScore results discussed in Section 5.1. Table 12 gives the evaluation protocol. The prompt for quality evaluation is as follows:

Prompt for quality evaluation

Please act as an impartial judge and evaluate the quality of the text completions provided by two large language models to the prefix displayed below. Assess each response according to the criteria outlined.

After scoring each criterion, provide a summary of your evaluation for each response. Additionally, ensure that the order in which the responses are presented does not affect your decision. Do not allow the length of the responses to influence your evaluation. Be as objective as possible.

Criteria:

1. Relevance to the prefix
2. Coherence
3. Interestingness

Start with a brief statement about which response you think is better overall. Then, for each criterion, state which response is better, or if there is a tie, followed by a concise justification for that judgment.

At the last line of your response, declare your verdict by giving one of the choices below, strictly following the given format: [[A]] if assistant A is better overall, [[B]] if assistant B is better overall, or [[C]] for a tie.

Only output the verdict in the last line, no other text.

[Prefix] prefix

[Response A] candidate_1

[Response B] candidate_2

Tables 13, 14, and 15 provide the controlled decoder-ablation setup, the definition of three scoring variants, and the full raw results for both tested watermark strengths.

B.7 Computational Overhead

We evaluate decoding efficiency under the same hardware and timing protocol used in the main experiments. Detection time excludes text generation. For a controlled comparison, we use CTWL as the baseline because it is evaluated through the same implementation pipeline; CS and RSBH rely on a separate codebase, while MPAC achieves nearly zero exact payload recovery in the main experiments.

Table 16 reports runtime over four matched model–dataset settings, each containing 500 texts. SDBMW is consistently $7.86\text{--}8.26\times$ faster than CTWL, reducing the detection time from 4.97–6.04 to 0.63–0.73 seconds per text.

To identify the main sources of overhead, we further profile raw-count and z-score decoding on the same 200 SDBMW outputs, each containing 400 scored tokens. Each text is measured three times after 10 warm-up iterations. Model loading, one-time mask construction, and diagnostic computations are excluded.

Z-score calibration adds 0.5831 seconds per text relative to raw counting. Candidate-wise accumulation accounts for 96.6% of this increase, whereas the proxy-model forward pass constitutes only 2.2% of the total z-score decoding time. Thus, proxy inference is not the dominant overhead; candidate-wise calibration is the primary target for further optimization.

C Segmentation Validation Details

Tables 18, 19, and 20 separate generation settings, watermark/evaluation settings, and payload-

Table 11: Zero-bit detection experiment. Each value is TPR@1%FPR computed over the token-filtered watermarked subset n_{wm} ; this denominator is distinct from n_{payload} in Table 1.

Dataset	Model	Method	No Attack	Substitution	Truncation
C4	Llama-3.1-8B	CTWL	0.978	0.714	0.944
		MPAC	0.560	0.168	0.420
		SDBMW	0.992	0.958	0.983
		CS	0.996	0.748	0.969
		RSBH	0.996	0.746	0.978
	Qwen-3-8B	CTWL	0.994	0.684	0.944
		MPAC	0.358	0.074	0.222
		SDBMW	1.000	0.964	0.994
		CS	0.986	0.636	0.916
		RSBH	0.982	0.560	0.907
Essays	Llama-3.1-8B	CTWL	0.936	0.520	0.835
		MPAC	0.730	0.406	0.618
		SDBMW	0.984	0.904	0.967
		CS	1.000	0.916	0.991
		RSBH	1.000	0.928	0.992
	Qwen-3-8B	CTWL	0.998	0.434	0.949
		MPAC	0.348	0.076	0.203
		SDBMW	1.000	0.964	1.000
		CS	0.978	0.620	0.835
		RSBH	0.962	0.608	0.831

Actual zero-bit denominators. The truncation counts after the 250-token filter are:

Dataset	Model	CTWL	MPAC	SDBMW	CS	RSBH
C4	Llama-3.1-8B	468	481	475	454	455
C4	Qwen-3-8B	480	474	487	465	462
Essays	Llama-3.1-8B	492	484	488	466	500
Essays	Qwen-3-8B	495	492	493	492	492

For no attack and substitution, all cells use $n_{\text{wm}} = 500$ except CS/Llama/Essays (499 and 498, respectively).

to-segment layouts used by the segmentation, length-scaling, and RS+ECC ablations. All settings use C4 prompts and Llama-3.1-8B, evaluate exact message accuracy under the no-attack setting, and use random payloads. Figures 3 and 4 provide the appendix-only ablations discussed in Section 5.2.

Table 12: Quality-evaluation configuration. Candidate A is the unwatermarked continuation and candidate B is the watermarked continuation; the DeepSeek preference counts a B win as 1, a tie as 0.5, and an A win as 0.

Item	Configuration
Prompt source	C4 prompts, paired by prompt ID between unwatermarked and watermarked outputs.
Compared continuations	One unwatermarked continuation versus one watermarked continuation generated with the same base LM and the same sampling setting as Table 4.
Judged methods	CS, MPAC, CTWL, RSBH, SDBMW, XMark, and StealthInk.
Base LMs and sample counts	BERTScore uses 100 Llama-3.1-8B and 200 Qwen-3-8B pairs per method. The judge has the same valid counts except for SDBMW with Llama, for which one provider-rejected request is excluded ($n = 99$).
Judge model	DeepSeek V4 Flash (deepseek-v4-flash-0731) (Alibaba Cloud, 2026) through the Alibaba Cloud Model Studio OpenAI-compatible endpoint, with thinking enabled, temperature 0, top- $p = 0.95$, and a neutral system prompt.
Judge output budget	The five original methods use an 8,192-token output cap. Among the 600 XMark/StealthInk requests, 64, 130, and 406 use 2,048-, 4,096-, and 8,192-token caps, respectively; every retained response ends with a valid verdict.
Candidate truncation	Each candidate is trimmed to at most 300 words before judging; incomplete trailing sentences are removed when possible.
Judging criteria	Prompt relevance, coherence, and interestingness.
Verdict parser	The final line must be <code>[[A]]</code> , <code>[[B]]</code> , or <code>[[C]]</code> ; malformed verdicts and provider-rejected requests are excluded from the denominator.
Aggregation	Watermarked preference = $(\#B \text{ wins} + 0.5 \cdot \# \text{ties}) / \# \text{valid judgments}$; 95% confidence intervals use 10,000 bootstrap resamples with seed 12.
BERTScore	F1 from RoBERTa-large layer 17 without baseline rescaling, using the same paired continuations and truncation policy; intervals use 10,000 bootstrap resamples with seed 12.

Table 13: SDBMW normalization-ablation configuration. The ablation keeps the generated texts, attacked texts, and embedded messages fixed and recomputes only the decoding statistic.

Component	Configuration	Notes
Generation setting	Llama-3.1-8B on C4; 500 samples; random 20-bit payloads; two 10-bit SDBMW segments; maximum generation length 400 tokens; temperature 0.7; top- $p = 0.9$; repetition penalty 1.05; $\gamma = 0.5$.	This matches the main SDBMW robustness setup except for the weaker watermark strength used in the stress test.
Attack setting	substitution_0.25.	The attack replaces 25% of words with synonyms before decoding.
Main ablation strength	$\delta = 2$.	This weak-watermark setting makes raw-count candidate bias easier to observe; Table 15 repeats the raw result values used in Table 3.
Secondary check	$\delta = 3$.	This main-strength check uses the same fixed-text protocol and tests whether the normalization trend remains under the robustness setting.
Proxy model	Llama-3.2-1B with bfloat16 precision.	The proxy estimates candidate-wise expected green-token counts and variances.
Minimum-token filtering	No additional minimum-token filter is applied to the attribution metric in this ablation.	The reported bit accuracy and exact message accuracy are computed over the same 500 attacked samples for all detector variants.
Score files	Raw, centered, and z-score JSONL score files are exported separately.	The multisegment analysis helper consumes the three detector-specific score files for the same substitution_0.25 attack output.

Table 14: Decoder variants in the normalization ablation. Let c_m be the observed green-token count for candidate message m , and let μ_m and σ_m^2 be the proxy-estimated mean and variance.

Detector	Candidate score	Uses proxy mean	Uses proxy variance
Raw count	c_m	No	No
Centered	$c_m - \mu_m$	Yes	No
Z-score	$(c_m - \mu_m) / \sqrt{\sigma_m^2}$	Yes	Yes

Table 15: Raw decoder-ablation results for the two tested watermark strengths. The $\delta = 2$ results correspond to Table 3 in the main text; the $\delta = 3$ results provide the main-strength secondary check.

Detector	$\delta = 2$ Bit Acc.	$\delta = 2$ Accuracy	$\delta = 3$ Bit Acc.	$\delta = 3$ Accuracy
Raw count	0.8936	0.654	0.9707	0.898
Centered	0.9453	0.816	0.9807	0.940
Z-score	0.9470	0.820	0.9811	0.940

Table 16: Detection runtime across four matched model–dataset settings. Values report the range across settings.

Method	Samples	Mean tokens	Time/text (s)	s/100 tokens
CTWL	500	399.5–400.0	4.97–6.04	1.24–1.51
SDBMW	500	400.0	0.63–0.73	0.158–0.183

Table 17: Fine-grained runtime profile of SDBMW decoding.

Detector	Time/text (s)	95% CI (s)	Proxy forward (s)	Candidate accumulation (s)
Raw count	0.1298	[0.1295, 0.1302]	–	0.1274
Z-score	0.7129	[0.7123, 0.7135]	0.0159	0.6905

Table 18: Generation setup for the segmentation, length-scaling, and RS+ECC ablations. The 20-bit sweep row describes the result files used for Figure 2(a); rerun wrappers expose the sample count through NSAMPLES.

Ablation	Provenance	Sampling setup
20-bit sweep	20-bit sweep wrappers; Figure 2(a) plotter.	100 plotted samples per valid (s, δ) point; length 200; temp. 0.8; top- p 0.95; rep. penalty 1.0; seed 0.
32-bit sweep	32-bit sweep wrapper; Figure 2(b) plotter.	100 samples per setting; length 200; temp. 0.8; top- p 0.95; rep. penalty 1.0; seed 0.
40-bit sweep	40-bit sweep wrapper; Figure 2(c) plotter.	100 samples per setting; length 200; temp. 0.8; top- p 0.95; rep. penalty 1.0; seed 0.
Length scaling	RSBH length-generation wrapper, CPU evaluator, and Figure 3 plotter.	100 samples per setting; $ x \in \{200, 250, 300, 350, 400\}$; temp. 0.7; top- p 0.9; rep. penalty 1.2; seed 0.
RS+ECC	RS+ECC sweep wrapper; Figure 4 plotter.	200 samples per setting by wrapper default; length 200; temp. 0.8; top- p 0.95; rep. penalty 1.0; seed 0.

Table 19: Watermark and evaluation setup for the segmentation, length-scaling, and RS+ECC ablations.

Ablation	Payload / detector	Watermark and metric
20-bit sweep	Payload space 2^{20} ; segmented decoding.	$\gamma = 0.5$; $\delta \in \{1.0, 1.5, \dots, 4.5\}$; no attack; exact message acc. from <code>results.jsonl</code> and <code>scores.jsonl</code> .
32-bit sweep	Payload space 2^{32} ; segmented decoding.	$\gamma = 0.5$; $\delta \in \{1.0, 1.5, \dots, 4.5\}$; no attack; exact message acc.
40-bit sweep	Payload space 2^{40} ; segmented decoding.	$\gamma = 0.5$; $\delta \in \{1.0, 1.5, \dots, 4.5\}$; no attack; exact message acc.
Length scaling	Payload space 2^{20} ; balanced RSBH mapping.	$\gamma = 0.5$; fixed $\delta = 2.0$; no attack; exact message acc.
RS+ECC	Payload space 2^{20} ; RS-coded segmented decoding.	$\gamma = 0.5$; $\delta \in \{1.0, 1.5, \dots, 4.5\}$; no attack; exact message acc.

Table 20: Payload and segment layouts for the segmentation, length-scaling, and RS+ECC ablations. Here B is the payload length, s is the number of information segments, m is the number of bits per segment, and n is the number of encoded segments after Reed–Solomon coding.

Experiment	Method	Segment layouts	Notes
20-bit sweep	CS for $s = 1$; pseudo-random RS for $s > 1$.	$B = 20$ $s = 1, 2, 4, 5, 10, 20$ $m = 20, 10, 5, 4, 2, 1; n = s$	No ECC redundancy.
32-bit sweep	Pseudo-random RS.	$B = 32$ $s = 2, 4, 8, 16, 32$ $m = 16, 8, 4, 2, 1; n = s$	No $s = 1$ CS baseline: it would enumerate 2^{32} candidates.
40-bit sweep	Pseudo-random RS.	$B = 40$ $s = 2, 4, 5, 8, 10, 20, 40$ $m = 20, 10, 8, 5, 4, 2, 1; n = s$	No $s = 1$ CS baseline: it would enumerate 2^{40} candidates.
Length scaling	Balanced RSBH.	$B = 20$ $s = 1, 2, 4, 5, 10, 20$ $m = 20, 10, 5, 4, 2, 1; n = s$	One map per s ; map-building key 426371835.
RS+ECC	Pseudo-random RS with redundancy.	$B = 20$ RS(4, 2) over $\text{GF}(2^{10})$ RS(6, 4) over $\text{GF}(2^5)$ RS(7, 5) over $\text{GF}(2^4)$	RS(12, 10)/ $\text{GF}(2^2)$ and RS(22, 20)/ $\text{GF}(2^1)$ are invalid and skipped.

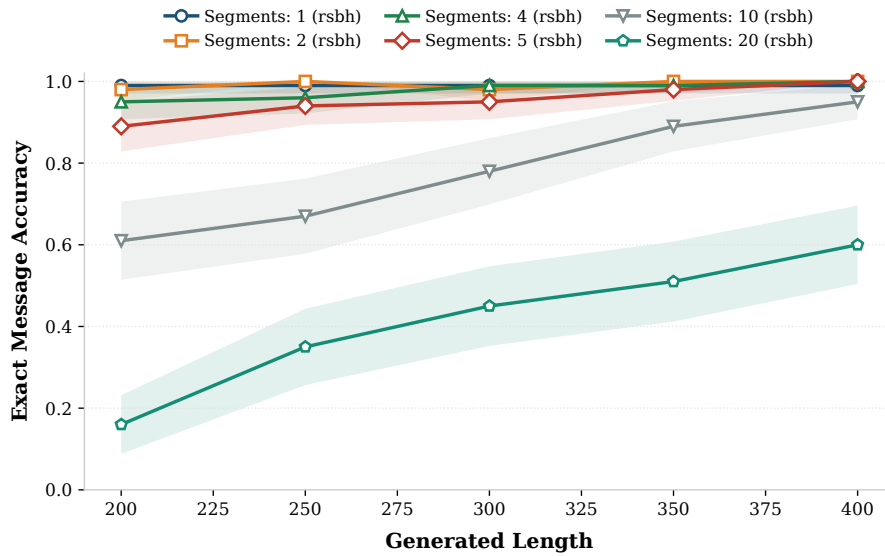


Figure 3: Length-scaling ablation with balanced RSBH mapping at fixed watermark strength $\delta = 2.0$. Longer generations improve exact message accuracy, but the higher-segmentation settings remain below the lower-segmentation settings across the tested lengths.

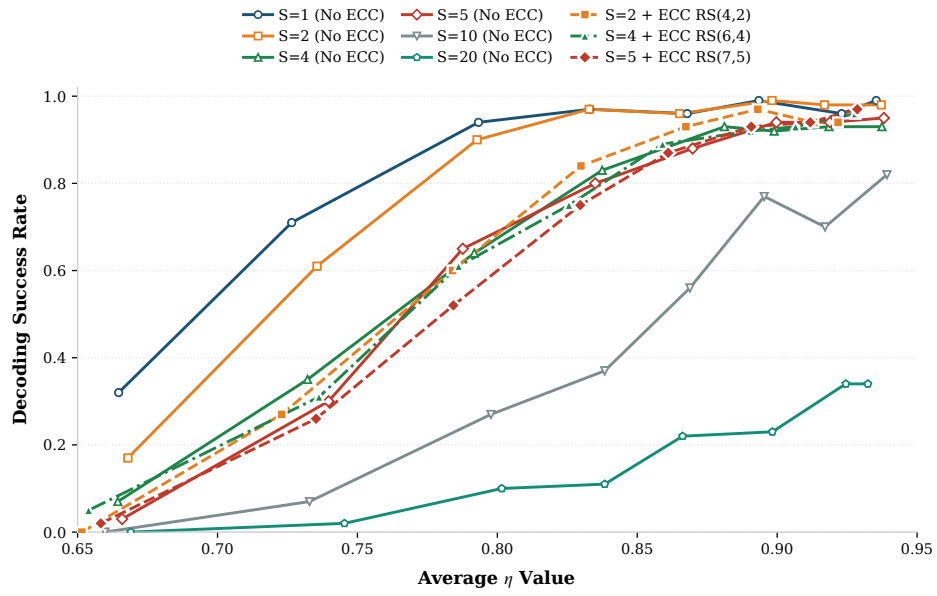


Figure 4: RS+ECC comparison for the 20-bit segmentation ablation. The ECC variants preserve the original payload but encode $S = 2$, $S = 4$, and $S = 5$ information segments as RS(4, 2), RS(6, 4), and RS(7, 5) codewords, respectively. Redundancy can help constrain decoded messages, but additional encoded segments also reduce the evidence available to each segment.