Advancing Beyond Identification: Multi-bit Watermark for Large Language Models

Anonymous ACL submission

Abstract

We show the viability of tackling misuses of large language models beyond the identification of machine-generated text. While existing methods focus on detection only, some malicious misuses demand tracing the adversary user for counteracting them. To address this, we propose Multi-bit Watermark via Position Allocation, embedding traceable multi-bit information during language model generation. Leveraging the benefits of zero-bit watermarking (Kirchenbauer et al., 2023a), our method enables robust extraction of the watermark without any model access, embedding and extraction of long messages ($\geq$ 32-bit) without finetuning, and maintaining text quality, while allowing zero-bit detection all at the same time. Moreover, our watermark is relatively robust under strong attacks like interleaving human texts and paraphrasing. We compare with existing works to show the effectiveness of our scheme in terms of robustness and latency.

1 Introduction

How can we take a step further from merely identifying machine-generated text to proactively tackling misuses of large language models? The emergence of human-like language models and their easily accessible nature via web interface and APIs have garnered unprecedented attention from the public and academia (Hu, 2023). The ability to follow complex instructions has boosted the productivity of various tasks such as programming, creative writing, and more. However, there have been increasing concerns about exploiting such language models to automate malicious activities such as spreading disinformation. This has necessitated the development of various methods to detect machine-generated texts through techniques such as zero-shot detection, supervised training, watermarking, and more (Mitchell et al., 2023; Wang et al., 2023b; Kirchenbauer et al., 2023a; Krishna et al., 2023). These endeavors focus on the crucial task of *identifying* machine-generated content, which serves as a pivotal step in mitigating the potential harm caused by such text.

However, when it comes to more pernicious misuses of large language models, such as the dissemination of misinformation and propaganda on social media platforms, the stakes are considerably higher, potentially leading to the erosion of social trust (Valenzuela et al., 2022). Notable instances that exploited automated bots in the past include manipulating an election campaign (Badawy et al., 2018), spreading disinformation about the Russian invasion of Ukraine (Pierri et al., 2023), and promoting products through fake reviews (Annie, 2023). With the rapid pace at which large language models are currently developed, similar threats will be automated in a much more rapid and delicate manner in the future.

In such circumstances, merely identifying the machine-generated text may not suffice for the language model providers. Instead, the ability to trace back to the adversary user responsible for generating the content becomes pivotal in counteracting such misuses. By doing so, the API providers can take a precursory measure to ban these users from their systems. More importantly, this allows media and social platforms, along with API providers, to collaborate with law enforcement authorities and take more decisive actions. All in all, watermarking the user information (or part thereof) can hold the adversary user accountable for potential harms facilitated through language model APIs without having to store user queries (Krishna et al., 2023), which would be prohibitively expensive and concern ordinary users who value privacy. Additionally, watermarking can enable language model providers to bind meta-data (e.g. model versions) for tracing the provenance of the language model output.

All this can be achieved by embedding multi-bit information. Our proposed method **M**ulti-bit watermark via **P**osition **A**llo**c**ation (MPAC) first allocates each token pseudo-randomly onto a single position of the message to be embedded. Then the message content at the allocated position decides which subset of tokens to favor following a zero-bit watermarking scheme (Kirchenbauer et al., 2023a) that pseudo-randomly biases certain token subsets for generation. To increase load capacity, we can further partition the vocabulary into multiple "colored" lists instead of a single green list, effectively encoding multiple states for every token. We demonstrate the effectiveness of our method compared to concurrent works in terms of robustness and generation latency, espeically in high bit-width and high corruption settings. Finally, we discuss and analyze the limitations of multi-bit watermarking in Section 5 – namely, the trade-off between watermark detection and bit-width.

Since our method works on top of zero-bit watermarking, it leverages most of the advantages: (1) Multi-bit message can be extracted without access to the model parameters or the API, allowing other parties to extract the adversary information (e.g. timestamp, ID) if given access to the extraction algorithm. (2) It can be done on the fly without pretraining or finetuning the model and can embed and extract long messages ($\geq$ 32-bit) with negligible overhead. (3) The watermark is not fragile against realistic corruptions such as interleaving with human texts or paraphrasing. This has not been previously demonstrated in other post-processing multi-bit watermarks (Yoo et al., 2023) or stenography methods (Ziegler et al., 2019; de Witt et al., 2023). (4) Finally, our watermarking framework can distinguish between machine and human text and simultaneously embed multi-bit information while maintaining the same text quality as its zero-bit counterpart. Our experiments demonstrate that 8-bit messages can be embedded effectively in short text lengths ($\leq$ 100 words) with over 95% bit accuracy. We hope this opens up new research directions for proactively counteracting malicious use cases of language model APIs.[1]

2 Related Works

Watermarking has been studied in various types of multimedia such as image (Potdar et al., 2005), video (Asikuzzaman and Pickering, 2017), audio (Hua et al., 2016), and natural language (Topkara et al., 2005). Following previous works (Zhu et al., 2018; Luo et al., 2020), we use the term watermarking to denote embedding information into natural language in a manner that is robust against possible attacks given a watermarked text – in our case, this is the output generated by a language model given the prompt. This differs from steganography (Cheddad et al., 2010; Fang et al., 2017; Ziegler et al., 2019; de Witt et al., 2023), which focuses more on the undetectability of a secret message that is embedded in the multimedia rather than robustness. For instance, Ziegler et al. (2019) sequentially encodes information via arithmetic coding every token. Naively applying this deterministic encoding scheme makes the watermark extremely fragile to simple corruptions as shown in Appendix Fig. 5.

Recently, methods relying on neural networks have shown progress in natural language watermarking, outperforming traditional methods that rely on rule-based watermarks (Topkara et al., 2006b,a; Atallah et al., 2001). Abdelnabi and Fritz (2021) proposed an end-to-end framework where a decoder network predicts the encoded message. Yang et al. (2022) improved upon the quality of the watermarked text by using an algorithmic approach. Building upon this, Yoo et al. (2023) focused on robustness and capacity, outperforming previous works on both aspects. However, since the proposed method works at the sentence-level, any addition or removal of a sentence will fail to extract the watermark. Moreover, these works cannot distinguish non-watermarked texts, making them unsuitable for distinguishing between machine text and human text.

Meanwhile, directly watermarking language models in a zero-bit manner during token generation has emerged as a promising approach for distinguishing language model outputs from human text (Kirchenbauer et al., 2023a; Aaronson and Kirchner, 2023) while achieving robustness against realistic attacks (Kirchenbauer et al., 2023b). Several works have improved upon Kirchenbauer et al. (2023a), e.g., in low entropy generation tasks such as code generation (Lee et al., 2023), undetectability of the watermark (Christ et al., 2023), and its robustness (Munyer and Zhong, 2023). We focus on extending the prior work for a more proactive counteraction towards identifying malicious users of language models by embedding *any* information while maintaining the key advantages.

[1] https://github.com/anoymous92874838/multibit-watermark-for-llms

Concurrent to our work, Fernandez et al. (2023a) and Wang et al. (2023a) use the entire message to create a signal unique to each message. Crucially, both works use the entire message content directly during embedding as input to the random seed generator, which leads to key differences in terms of robustness and latency. We further discuss their methodology in comparison with ours in the next section. Aside from this, Wang et al. (2023a) further utilize a proxy language model to enhance text quality.

3 Method

We outline the multi-bit watermark protocol:

1. A user sends a prompt X to the language model provider.
2. Using the message encoding function $\mathcal{E}$, the language model provider generates watermarked text Y embedded with a multi-bit information. The message contains user-specific meta-data that can aid tracing back to the user (e.g. timestamp, location, ID).
3. The user publishes the text $\tilde{Y}$, which may be edited from the original watermarked text.
4. If the published text is deemed unsafe or malicious, the detector inspects $\tilde{Y}$ (i) to determine whether the watermark is present (zero-bit detection) and (ii) decode the multi-bit message to take further measure.

3.1 Zero-bit Watermarking (Kirchenbauer et al., 2023a)

As a preliminary, we briefly review zero-bit watermarking introduced by Kirchenbauer et al. (2023a) and elaborate on extending this method to multi-bit watermarking. An auto-regressive language model $p(y|x)$ predicts the probability distribution over the next token $\Delta(\mathcal{V})$ given arbitrary length prefix tokens where $\mathcal{V}$ is the vocabulary. A zero-bit watermark is embedded by biasing the language model to output a certain subset of tokens. That is, the message encoding function $\mathcal{E} : \Delta(\mathcal{V}) \rightarrow \Delta(\mathcal{V})$ generates another probability distribution that alters the original distribution of $p(y|x)$.

For Kirchenbauer et al. (2023a), the message encoding function pseudo-randomly chooses a subset of tokens at each token step t to form a green list $\mathcal{G}_t$. The logit scores $l_t \in \mathbb{R}^{|\mathcal{V}|}$ are modified towards selecting the green-listed tokens in favor of the other tokens by adding a bias term δ to the logits in $\mathcal{G}_t$. Instead of fixing the greenlist using rule-based heuristics such as spelling or synonym variations (He et al., 2022), the greenlist is selected pseudo-randomly at each time step to minimize a noticeable shift in text distributions. At each time step, a seed s is outputted depending on the previous h tokens using a pseudo-random function $f : \mathbb{N}^h \rightarrow \mathbb{N}$, and s is used to sample $\mathcal{G}_t$ from $\mathcal{V}$.

We dub this message encoding function as `Greenlist`. Given $t-1$ prefix tokens $X_{1:t-1}$, and pseudo-random function f, the t^{th} token is generated by

```
Greenlist
  1. Compute hash of tokens $s = f(X_{t-h:t-1})$.
  2. Permute vocabulary $\mathcal{V}_t$ using $s$ as seed for a random number generator (RNG).
  3. Let $\mathcal{G}_t$ be the first $\gamma|\mathcal{V}|$ tokens from $\mathcal{V}_t$
  4. Add $\delta$ to token logits in $\mathcal{G}_t$.
```

To determine the presence of the watermark, the detector inspects the ratio of the green-listed token. A watermarked text will ideally have a high ratio of green tokens as shown in Fig. 1 Right.

3.2 MPAC: Extending to Multi-bit Watermark

The objective of multi-bit watermarking is to embed and extract a message $\mathbf{m} \in \Sigma^b$ where Σ denotes the r-nary possible strings, or more commonly referred to as the alphabet. For a binary message, $\Sigma = \{0, 1\}$. We let $p \in \{0, \ldots, b-1\}$ denote the position of the message and $\mathbf{m}[p] \in \{0, \ldots, r-1\}$ the message content at position p. Hereafter, we use $[a]$ to denote the integer set $\{0, \ldots, a-1\}$.

Our proposed method **M**ulti-bit watermarking via **P**osition **A**llo**c**ation (`MPAC`) works by allocating the tokens to message positions. First, notice that zero-bit watermarking can be viewed as watermarking a single bit of information stating the existence of a watermark ($\mathbf{m}$=0). In essence, each token generated by the language model is a signal that reinforces the watermark.

Our message encoding function $\mathcal{E} : \Sigma^b \times \Delta(\mathcal{V}) \rightarrow \Delta(\mathcal{V})$ alters the probability distribution dependent on the message. We first assign a position p using a random number generator seeded with s. Then the message content $m = \mathbf{m}[p] \in [r]$ is encoded by permuting $\mathcal{V}$ and favoring the m^{th}

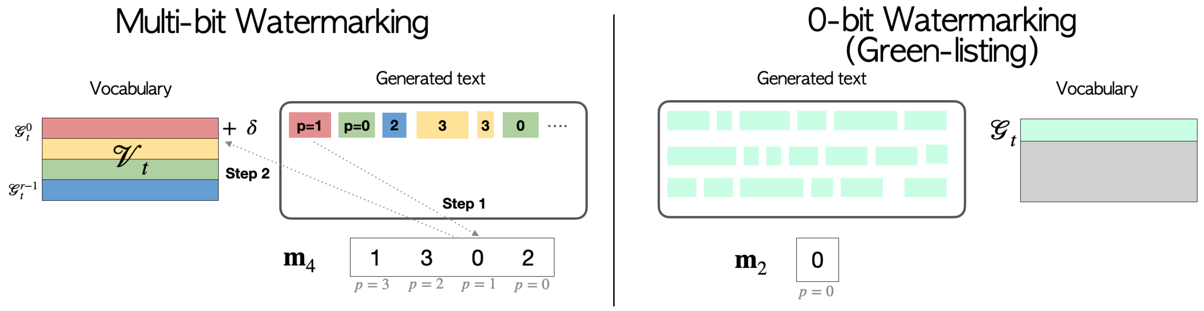


Figure 1: An overview of our method MPAC. The number inside a token (e.g. $\boxed{p=1}$) denotes the allocated position, while the color signifies the message content at that position. At Step 1, a position is sampled prior to generating a token. At Step 2, the message at that position determines the token subsets to favor. *Right:* Zero-bit watermarking can be viewed as a special case of multi-bit watermarking.

subset. Our message encoding function is extremely easy to implement over the `Greenlist` scheme. We highlight the steps in colors that are specific to ours. All other steps are identical to (Kirchenbauer et al., 2023a):

`MPAC`

1. Compute $s = f(X_{t-h:t-1})$.
2. $p \leftarrow$ `sample`$([b])$ using s as seed.
3. $m \leftarrow \mathbf{m}[p]$
4. Permute vocabulary $\mathcal{V}_t$ using s as seed.
5. Partition $\mathcal{V}_t = [\mathcal{C}_t^0, \cdots, \mathcal{C}_t^{r-1}]$ discarding remainders if any.
6. Add δ to token logits in $\mathcal{C}_t^m$.

Here r is the number of available partitions. The number of vocabulary partitions is determined by the greenlist proportion γ, i.e. $r = \lfloor \frac{1}{\gamma} \rfloor$. When $r > 2$, we can further increase the load capacity by taking advantage of all the 'colored' lists (hence, the notation $\mathcal{C}$), instead of only using the greenlist. Given a binary message of length b, the message is convereted to radix r attaining $\mathbf{m}_r \in [r]^{\tilde{b}}$ where $\tilde{b} = \lceil \frac{b}{\log_2 r} \rceil$. In Figure 1 Left, we illustrate the case of $r = 4$ and $b = 8$, where the 8-bit message is converted into radix 4, resulting in an effective message length of 4 ($\tilde{b} = 4$)[2].

At each token generation, the message content at the assigned position p determines which colorlist to add δ to. If the message content is '0', the tokens from the first list (red in Fig. 1) are favored. Note that zero-bit watermarking can be seen as a special case of embedding the same single bit message ($b = 1, \mathbf{m} = 0$) as shown in Figure 1-Right.

Message Decoding Given a watermarked language model output, we determine the position and which colorlist each token is from and increment the number of tokens in the colored lists. For instance, for the t^{th} token with message position $p = i$ and the j^{th} colorlist $\mathcal{C}_t^j$, we increment the counter $\mathbf{W}_i[j]$. After computing this on the entire text segment, we predict the message content by taking the colorlist with the most tokens for each position. A more detailed algorithm is shown in Algorithm 1.

3.3 Detecting Machine Text

To distinguish between a watermarked text and a non-watermarked (human-written) text, we count the number of tokens assigned to the predicted message. This corresponds to w in Line 12 of Algorithm 1. We model the number of tokens in the argmax colorlist of position i as a random variable $C_i \overset{H_0}{\sim} \text{Binomial}(T_i, \gamma)$ where T_i is the number of tokens assigned to position i. As $C_0, \ldots, C_{b-1}$ are independent for a fixed set of trials $(T_i, \ldots, T_{b-1})$ and have the same success probability parameter, the sum of these is a binomial random variable as well:

$$C = C_0 + \cdots + C_{b-1} \overset{H_0}{\sim} \text{Binomial}(T, \gamma) \quad (1)$$

where $T = T_0 + \cdots + T_{b-1}$. This reduces to the same random variable used in zero-bit watermarking and we can compute the z-statistics. More discussions regarding the details of the z-statistic and other possible statistics are outlined in Appendix A.2.

3.4 Comparison to Other Works

The message encoding function of existing works use the entire message $\mathbf{m}$. After permuting $\mathcal{V}_t$, Fernandez et al. (2023a) cyclically shift the vocabulary m_{10} times where m_{10} is the radix-10 form of $\mathbf{m}$. This modifies Step 2 of `Greenlist`. Wang et al. (2023a) hashes $\mathbf{m}$ to attain a seed s' to permute the

[2] Hereafter, we use b instead of $\tilde{b}$ to denote the effective message length (dimension of $\mathbf{m}_r$).

Algorithm 1: Message Decoding

```
Input: Watermarked text $X_{1:T}$, hash context width $h$, effective message length $\tilde{b}$
Output: Predicted message $\hat{\mathbf{m}}$, number of colorlisted tokens $w$
/* Initialize counter */
$\mathbf{W}_p[m] = 0 \; \forall p, m$
/* Count tokens in colorlists */
for $t$ in $[h+1, T]$ do
    $s = f(X_{t-h:t-1})$
    $p = \texttt{sample}([\tilde{b}])$
    for $m$ in $[r]$ do
        Permute $\mathcal{V}_t$ using $s$ as seed
        if $X_t \in \mathcal{G}_t^m$ then
            $\mathbf{W}_p[m]$ += 1
/* Predict message */
$\hat{\mathbf{m}}_r =$ " "
$w = 0$
for $p$ in $[\tilde{b}]$ do
    $w$ += $\texttt{max}(\mathbf{W}_p[m])$
    $\hat{m} = \texttt{argmax}_m(\mathbf{W}_p[m])$
    $\hat{\mathbf{m}}_r$ += $\texttt{str}(\hat{m})$
Get bit message $\hat{\mathbf{m}}$ by converting $\hat{\mathbf{m}}_r$
return $\hat{\mathbf{m}}$, $w$
```

vocabulary along with the seed attained from prefix tokens, modifying Step 1.

`Cyclic-Shift`

2'. Permute $\mathcal{V}_t$ using s as seed. Then, cyclic shift m_{10} times.

`Message-Hash`

1'. $s' \leftarrow$Hash (s+Hash (m_{10}))

Using the entire message leads to two key characteristics that diverge from ours. First, the hamming distance between two messages is not necessarily preserved after applying the encoding function. As an example, consider `Message-Hash`. Using the final seed s' created from $\mathbf{m} = 0000$ does not guarantee an output from the RNG that is any closer to that of $\mathbf{m} = 0001$ (hamming distance of 1) as it is to $\mathbf{m} = 1111$ (hamming distance of 4). This leads to an all-or-nothing behavior where either the entire message is extracted without error or is a completely random message. In the presence of high corruption, which reflects the real-world case, we show this behavior is not desirable as it lacks enough signal to correctly predict the message.

In addition, the exponential number of messages ($\mathcal{O}(2^b)$) should be considered during message decoding to find the optimal message, which renders decoding of long messages ($\geq$ 32-bit) computation-heavy[3] . For Fernandez et al. 2023a, the bit-width affects the *encoding phase* due to the cyclic shift operation, which is more problematic as it affects the end users. MPAC encodes and decodes each bit position of the message independently, which brings a negligible increase in the computation as the message length is increased.

The simplicity of our multi-bit watermark scheme via position allocation makes it easy to apply it on top of other methods. For example, using the position allocation scheme, we decompose the multi-bit message into blocks and hierarchically embed them using the message encoding scheme of Fernandez et al. (2023a). Details are in Appendix A.3. In addition, the message encoding function of `MPAC` can be generalized to other zero-bit watermark approaches that uses the exponential minimum sampling approach (Aaronson and Kirchner, 2023; Kuditipudi et al., 2023). The scheme is provided in Appendix A.4.

3.5 Techniques for Practical Use

List decoding is a well-established field in coding theory that decodes a list of messages that are within a certain hamming distance (Elias, 1991; Guruswami and Rudra, 2008; Guruswami, 2004). Inspired by this, we alter our decoding function to output candidate messages sorted by the level of confidence. In practice, list decoding is especially useful because provenance tracing via watermarking is far from finding an exact solution, but narrowing down the possible leakage points for a more detailed inspection that may be costly. For instance, when watermarking the timestamp of activity, it is useful to have a likely set of timestamps for which the practitioners to manually inspect, rather than a single candidate. This technique is not unique to ours and can be applied to other methods as long as the decoding stage is computationally feasible. We detail on how confidence score can be computed in Appendix A.3.

[3] See Section 7.5 of Wang et al., 2023a.

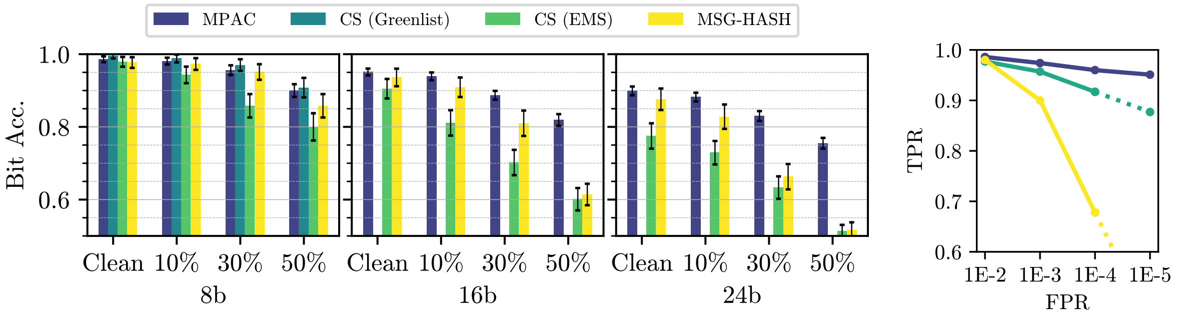


Figure 2: *Left*: Comparison with prior works without corruption (clean) and in the presence of copy-paste attack with $p\%$. On 24-bit, only 100 samples were watermarked for `Cyclic-Shift` and `Message-Hash` due to lengthened encoding / decoding time. *Right*: TPR for various FPR thresholds.

4 Experiments

4.1 Experimental Settings

For our main experiments, we use LLaMA-2-7B (Touvron et al., 2023) to generate sequences using the newslike subset of the Colossal Common Crawl Cleaned corpus (C4) dataset (Raffel et al., 2020) following previous work (Kirchenbauer et al., 2023a). This simulates the scenario of generating fake news given a certain topic. For watermarking and text generation, we follow the configurations used in Kirchenbauer et al. (2023b) unless otherwise denoted: bias $\delta = 2.0$, greenlist ratio $\gamma = 0.25$, which have shown a good trade-off between the detection performance and generation quality. Since $\gamma = 0.25$, the number of colors r is 4. We embed a random b-bit message onto >500 samples and report the mean metrics across samples.

When using the term 'bit' or 'bit-width', this denotes the initial message length and the effective message length is determined by r. When necessary, we also show the three standard error ranges. More details are in Appendix A.5.

Metrics To measure the performance of multi-bit watermarking, we use bit accuracy following previous works in the watermarking literature (Zhu et al., 2018; Luo et al., 2020; Yang et al., 2022; Yoo et al., 2023) to measure how much of the embedded bits can be extracted without error. For zero-bit watermark performance (i.e. machine-text detection), we use area under the ROC curve (AUROC) and the true positive rate (TPR) at various false postive rate thresholds. For text quality, we use the automatic metrics used in Kirchenbauer et al. (2023b) such as perplexity (PPL) using a larger oracle model (LLaMA-2-13B) and semantic similarity based on a paraphraser model (Wieting et al., 2022, P-SP). We further discuss the validity of the metrics in Appendix A.6.

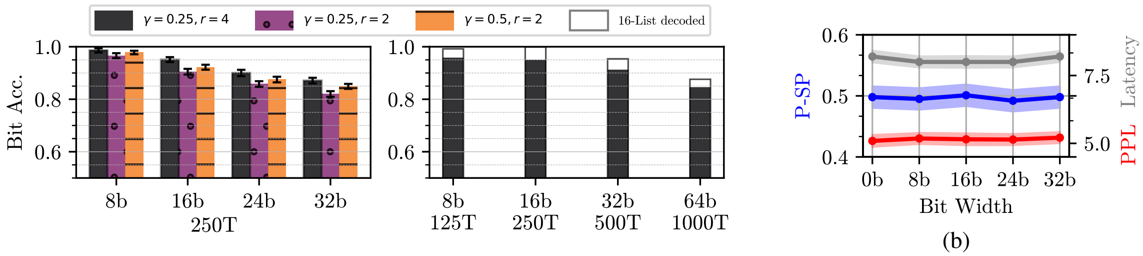
Threat Model In the real world, a user often edits the generated text before publishing either to enhance and/or in an attempt to evade the watermark. We study two types of attacks studied in the past work (Kirchenbauer et al., 2023b): *copy-paste* mixes the watermarked text and human text and *paraphrasing* uses another language model to paraphrase the watermarked text. For the copy-paste attack, we randomly interleave the generated watermarked text into a non-watermarked text, mixing a p percentage of non-watermarked texts while maintaining the total length. For paraphrasing, we use GPT-3.5-turbo (the prompt is shown in Table 15). Both attacks do not maintain the start and end tokens of the watermarked text.

4.2 Results

For numerical results, see the tables in Appendix A.10.

Comparison with Other Works. We compare `MPAC` with Fernandez et al. (2023a, `Cyclic-Shift`) and Wang et al. (2023a, `Message-Hash`). We do not compare with other stenography and post-processing works as they are extremely fragile in real-world corruption settings. Please refer to Sec. 2 for details. For `Cyclic-Shift`, the bit-width is bounded by $\log_2 |\mathcal{V}| \approx$15 bits, since the cyclic-shift operation is only unique up to the size of the vocabulary. Due to this, we also experiment with extending `Cyclic-Shift` to another zero-bit watermark method scheme called exponential minimum sampling (Aaronson and Kirchner, 2023, EMS), which does not have a theoretical upperbound. We call this `Cyclic-Shift` (EMS).

The results in Fig. 2 show the clean and robust multi-bit accuracy in the presence of the copy-paste attack. At 8-bit, all methods achieve nearly 100% accuracy and do fairly well even in the pres-



(a) (b)

Figure 3: (a) Clean bit accuracy with 3 standard errors for a fixed number of tokens (left) and fixed BPT (right). (b) Text quality (PPL, P-SP) and encoding latency across bit widths. 3 standard errors are shown.

ence of corruption. At higher bit-width, MPAC outperforms others in both clean and robust accuracy. As corruption rate is increased, the other methods show dramatic degradation. In contrast, MPAC can withstand them due to position allocation, which independently encodes each position. In Fig. 2 Right, we compare the watermark detection performance at 8-bit. For Cyclic-Shift and Message-Hash , we use 10,000 negative samples and the TPRs@FPR=1e−5 are linearly interpolated due to the lengthened decoding time. The results demonstrate that MPAC outperforms prior works at low FPR thresholds. Notably, at FPR=1e-5, our true positive rate is .951.

Enlarging the message length comes at the cost of computation for prior works. Increasing the bit-width from 16-bit→24-bit, lengthens the generation time of Cyclic-Shift by roughly 3.6x (14 seconds → 50 seconds) per sample, while MPAC does not have increased latency (Fig. 3b). Message-Hash does not suffer from latency overhead during encoding, but the computation and memory overhead increase exponentially during decoding.

Colorlisting improves multibit performance. Next, we verify the effectiveness of 'colorlisting', which takes advantage of the surplus vocabulary partitions. Fig. 3a demonstrates the gain in the load capacity by using r=4 colorlists as opposed to r=2 given a fixed γ. Besides the 8-bit case, which already achieves high accuracy, the performance of $\gamma = 0.25$, r=4 is statistically significant at p=1e−2 than the second best variant. We further discuss the implications of varying γ, r in Section 5.

Next, we increase the number of tokens (T) and bit width accordingly to demonstrate the feasibility of embedding longer messages. While the performance degrades as we increase the bit-width, the watermark does not entirely break, demonstrating the benefits of decomposing the message by positions. Moreover, the degradation can be partially compensated for by using list decoding. For 32-bit, the best possible message in the list achieves 95% bit acc. by verifying only 16 out of 2^{32} possible messages.

MPAC can maintain the watermark under various corruptions. The full results of copy-paste attack in Appendix Fig. 10. Even at 32-bit, our watermark is not entirely destroyed as we encode each position of the watermark independently, which shows that it can benefit from error correction codes. We found paraphrasing to be much more challenging than the copy-paste attack and thus, experimented with only 8-bit messages and increasing the token lengths (Fig. 10b). With T=500, the bit accuracy reaches nearly 80% and with 16-list decoding, we are able to attain 90% bit accuracy across all token lengths. More attacks are considered in Appendix A.8.

Detection performance is affected by bit-width. To get a clearer picture of the detection performance, we compute AUC vs. the number of tokens observed in Fig. 4a following Kirchenbauer et al. (2023b). We see that the detection performance decreases as the message bit is increased. This phenomenon is similarly observed in other works as the increase in the number of "hypotheses" required to check leads to an increase in the false positive rate (Fernandez et al., 2023b). We further discuss the reasons behind this in the subsequent section. Note, however, that a watermarked text with 32-bit message reaches AUC over 0.99 once observing 200 tokens (≈ 150 words). The TPR at FPR=1e−3 for b=$\{0, 8, 16, 24, 32\}$ are 0.98, 0.98, 0.95, 0.93, and 0.91, respectively (shown in Table 7).

Text quality is not affected by bit-width. MPAC extends zero-bit watermarking by allocating tokens to message positions and partitioning vocabularies, which would otherwise be allocated to a single position and a single vocabulary partition. Consequently, given the same δ and γ, it only alters

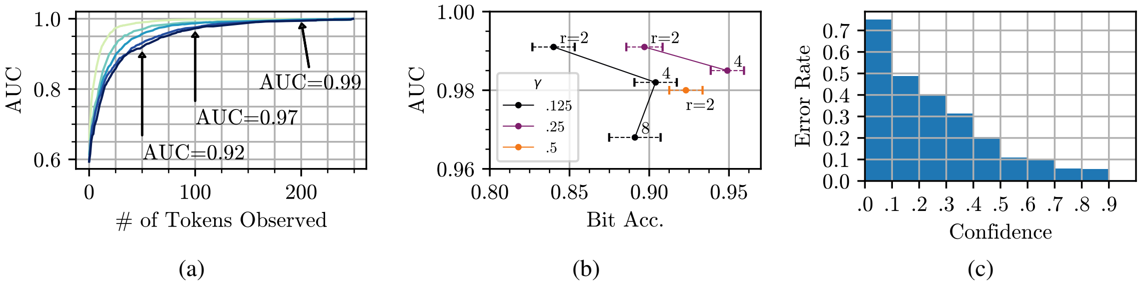


Figure 4: (a) AUC@number of tokens observed for b=$\{0, 8, 16, 24, 32\}$. Darker colors denote larger bit-widths. (b) Zero-bit and multi-bit watermark performance for varying γ and r for 1000 samples at T=100,b=8. (c) Error rate as a function of confidence.

the text distribution to an extent that zero-bit watermarking does regardless of the bit-width. Indeed, our empirical results in Fig. 3b demonstrate that the text quality is statistically indistinguishable across bit-widths. We also show that the encoding latency, which directly experiences user experience, does not increase with bit-width. Three standard error ranges are shown.

Across Model Scales, Datasets, Hash Schemes. We further experiment with other pretrained models (Jiang et al., 2023; Zhang et al., 2022) and their finetuned versions. Table 6 demonstrates Mistral and OPT also achieve a similar performance, showing that our method is not limited to a specific pretrained model. We also find that the finetuned versions are also capable of watermarking, though the RLHF-tuned LLaMA model show a slight drop-off. The results for larger models (13B, 70B) and other datasets are in Appendix A.9. To summarize, we found that text distributions with low entropy inherently have lower load capacity as observed similarly in prior works. However, our results consistently show that multi-bit watermarking is possible for open-form generation – which resembles disinformation generation – across model types and scales. We also present results for using another hash scheme with a longer context width in Appendix Table 12 and 13, which shows a similar level of performance.

5 Discussions

Load capacity and detection performance trade-off. As noted above, embedding longer messages degrades the watermark detection performance due to overestimating the statistics of non-watermarked human texts (Fig. 6). This is because computing the statistics involved finding the maximum cell value for each position. One natural solution is to use a better statistic that models the maximum cell value of a multinomial distribution. Empirically, we found that this performed on par or even slightly worse compared to the current approach, which may be due to the approximation error when using a small sample size. We give a more detailed discussion on this in Appendix A.2.

Radix and Colorlist proportion How do radix and colorlist proportion γ influence multi-bit watermark performance? For γ=.125, the benefits of enlarging r to 8 are saturated and show no statistical significance to r=4. While larger r allows more tokens to be assigned to each position by reducing the effective length of the message, it challenges the problem by increasing the number of possible answers (digits) per position. Additionally, we observed that increasing radix trade-offs zero-bit performance for multi-bit performance. The observations are illustrated in Fig. 4b.

List Decoding Ablation In Fig. 4c, we show a plot of bit error rate stratified by confidence. While not properly calibrated (under-estimation), having higher confidence leads to lower error rate. We also highlight the effectiveness of this technique by comparing it with randomly outputting candidate messages from scratch in Table 2. We also observed that randomly altering a single position provides a good list as the best candidate message is already a good starting point.

6 Conclusion

Our findings demonstrate the viability of embedding any information into the outputs of language models while having the capability to distinguish between machine text and human text. This unveils a novel prospect of counteracting high-stake misuse of large language models via API. One limitation of our approach is the reduced separability of machine and human text when embedding longer messages. Overhauling this limitation can be a major step towards deploying multi-bit watermark in the real world.

7 Ethics Statement

Watermarking can mitigate malicious use cases by being able to trace back to the malicious user. This will enable holding accountability on adversaries for their malfeasance. However, ordinary users may find the idea discomforting as it may give the sense that the API provider can know what outputs are fed to the individual users. This is not the case unless the content is published to the public by the user, which – in many cases – is already done in an environment where the user can be identified (e.g. social media). All in all, the identification of machine-generated texts and tracing their provenance can enhance the accountability of API access of large language models without breaching individual users' privacy.

References

Scott Aaronson and Hendrik Kirchner. 2023. Watermarking gpt outputs. https://www.scottaaronson.com/talks/watermark.ppt. Accessed: 2023-09-14.

Sahar Abdelnabi and Mario Fritz. 2021. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 121–140. IEEE.

Palmer Annie. 2023. People are using a.i. chatbots to write amazon reviews. *CNBC*.

Md Asikuzzaman and Mark R Pickering. 2017. An overview of digital video watermarking. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(9):2131–2153.

Mikhail J Atallah, Victor Raskin, Michael Crogan, Christian Hempelmann, Florian Kerschbaum, Dina Mohamed, and Sanket Naik. 2001. Natural language watermarking: Design, analysis, and a proof-of-concept implementation. In *International Workshop on Information Hiding*, pages 185–200. Springer.

Adam Badawy, Emilio Ferrara, and Kristina Lerman. 2018. Analyzing the digital traces of political manipulation: The 2016 russian interference twitter campaign. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 258–265.

Elwyn R Berlekamp. 1964. *Block coding with noiseless feedback*. Ph.D. thesis, Massachusetts Institute of Technology.

Abbas Cheddad, Joan Condell, Kevin Curran, and Paul Mc Kevitt. 2010. Digital image steganography: Survey and analysis of current methods. *Signal processing*, 90(3):727–752.

Miranda Christ, Sam Gunn, and Or Zamir. 2023. Undetectable watermarks for language models. *arXiv preprint arXiv:2306.09194*.

Christian Schroeder de Witt, Samuel Sokota, J Zico Kolter, Jakob Nicolaus Foerster, and Martin Strohmeier. 2023. Perfectly secure steganography using minimum entropy coupling. In *The Eleventh International Conference on Learning Representations*.

Peter Elias. 1991. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37(1):5–12.

Tina Fang, Martin Jaggi, and Katerina Argyraki. 2017. Generating steganographic text with lstms. In *Proceedings of ACL 2017, Student Research Workshop*, pages 100–106.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. 2023a. Three bricks to consolidate watermarks for large language models. *arXiv preprint arXiv:2308.00113*.

Pierre Fernandez, Guillaume Couairon, Hervé Jégou, Matthijs Douze, and Teddy Furon. 2023b. The stable signature: Rooting watermarks in latent diffusion models. *arXiv preprint arXiv:2303.15435*.

Philip Gage. 1994. A new algorithm for data compression. *C Users Journal*, 12(2):23–38.

Meghal Gupta, Venkatesan Guruswami, and Rachel Yun Zhang. 2023. Binary error-correcting codes with minimal noiseless feedback. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1475–1487.

Venkatesan Guruswami. 2004. *List decoding of error-correcting codes: winning thesis of the 2002 ACM doctoral dissertation competition*, volume 3282. Springer Science & Business Media.

Venkatesan Guruswami and Atri Rudra. 2008. Explicit codes achieving list decoding capacity: Error-correction with optimal redundancy. *IEEE Transactions on information theory*, 54(1):135–150.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. 2022. Protecting intellectual property of language generation apis with lexical watermark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10758–10766.

Krystal Hu. 2023. Chatgpt sets record for fastest-growing user base - analyst note. *Reuters*.

Guang Hua, Jiwu Huang, Yun Q Shi, Jonathan Goh, and Vrizlynn LL Thing. 2016. Twenty years of digital audio watermarking—a comprehensive review. *Signal processing*, 128:222–242.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023a. A watermark for large language models. *arXiv preprint arXiv:2301.10226*.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. 2023b. On the reliability of watermarks for large language models. *arXiv preprint arXiv:2306.04634*.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. 2023. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. *arXiv preprint arXiv:2303.13408*.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2023. Who wrote this code? watermarking for code generation. *arXiv preprint arXiv:2305.15060*.

Bruce Levin. 1981. A representation for multinomial cumulative distribution functions. *The Annals of Statistics*, pages 1123–1126.

Xiyang Luo, Ruohan Zhan, Huiwen Chang, Feng Yang, and Peyman Milanfar. 2020. Distortion agnostic deep watermarking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13548–13557.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Eric Mitchell, Yoonho Lee, Alexander Khazatsky, Christopher D Manning, and Chelsea Finn. 2023. Detectgpt: Zero-shot machine-generated text detection using probability curvature. *arXiv preprint arXiv:2301.11305*.

Travis Munyer and Xin Zhong. 2023. Deeptextmark: Deep learning based text watermarking for detection of large language model generated text. *arXiv preprint arXiv:2305.05773*.

Francesco Pierri, Luca Luceri, Nikhil Jindal, and Emilio Ferrara. 2023. Propaganda and misinformation on facebook and twitter during the russian invasion of ukraine. In *Proceedings of the 15th ACM Web Science Conference 2023*, pages 65–74.

Vidyasagar M Potdar, Song Han, and Elizabeth Chang. 2005. A survey of digital image watermarking techniques. In *INDIN'05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005.*, pages 709–716. IEEE.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Christoph Schuhmann. 2022. Huggingface datasets: Christophschuhmann/essays-with-instructions. `https://huggingface.co/datasets/ChristophSchuhmann/essays-with-instructions`. Accessed: 2023-09-14.

Mercan Topkara, Giuseppe Riccardi, Dilek Hakkani-Tür, and Mikhail J Atallah. 2006a. Natural language watermarking: Challenges in building a practical system. In *Security, Steganography, and Watermarking of Multimedia Contents VIII*, volume 6072, pages 106–117. SPIE.

Mercan Topkara, Cuneyt M Taskiran, and Edward J Delp III. 2005. Natural language watermarking. In *Security, Steganography, and Watermarking of Multimedia Contents VII*, volume 5681, pages 441–452. SPIE.

Umut Topkara, Mercan Topkara, and Mikhail J Atallah. 2006b. The hiding virtues of ambiguity: quantifiably resilient watermarking of natural language text through synonym substitutions. In *Proceedings of the 8th workshop on Multimedia and security*, pages 164–174.

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

Sebastián Valenzuela, Daniel Halpern, and Felipe Araneda. 2022. A downward spiral? a panel study of misinformation and media trust in chile. *The International Journal of Press/Politics*, 27(2):353–373.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2023a. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992*.

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Chenxi Whitehouse, Osama Mohammed Afzal, Tarek Mahmoud, Alham Fikri Aji, et al. 2023b. M4: Multi-generator, multi-domain, and multi-lingual black-box machine-generated text detection. *arXiv preprint arXiv:2305.14902*.

Stephen B Wicker and Vijay K Bhargava. 1999. *Reed-Solomon codes and their applications*. John Wiley & Sons.

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. 2022. Paraphrastic representations at scale. In *Proceedings of the The 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 379–388.

Xi Yang, Jie Zhang, Kejiang Chen, Weiming Zhang, Zehua Ma, Feng Wang, and Nenghai Yu. 2022. Tracing text provenance via context-aware lexical substitution. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11613–11621.

KiYoon Yoo, Wonhyuk Ahn, Jiho Jang, and Nojun Kwak. 2023. Robust multi-bit natural language watermarking through invariant features. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2092–2115.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

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

Zachary Ziegler, Yuntian Deng, and Alexander M Rush. 2019. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215.

A Appendix

Table of Contents

A.1 Decoding Algorithm

```
Algorithm 2: Message Decoding
Input: Watermarked text $X_{1:T}$, hash context width $h$, effective message length $\tilde{b}$
Output: Predicted message $\hat{\mathbf{m}}$, number of colorlisted tokens $w$
/* Initialize counter */
$\mathbf{W}_p[m] = 0 \; \forall p, m$
/* Count tokens in colored lists */
for $t$ in $[h+1, T]$ do
    $s = f(X_{t-h:t-1})$
    $p = \texttt{sample}([\tilde{b}])$
    for $m$ in $[r]$ do
        Permute $\mathcal{V}_t$ using $s$ as seed
        if $X_t \in \mathcal{G}_t^m$ then
            $\mathbf{W}_p[m]$ += 1
/* Predict message */
$\hat{\mathbf{m}}_r =$ " "
$w = 0$
for $p$ in $[\tilde{b}]$ do
    $w$ += $\texttt{max}(\mathbf{W}_p[m])$
    $\hat{m} = \texttt{argmax}_m(\mathbf{W}_p[m])$
    $\hat{\mathbf{m}}_r$ += $\texttt{str}(\hat{m})$
Get bit message $\hat{\mathbf{m}}$ by converting $\hat{\mathbf{m}}_r$
return $\hat{\mathbf{m}}$, $w$
```

A.2 Analysis on Watermark Detection

A.2.1 Watermark Detection

The presence of a watermark is determined by counting the number of tokens in the greenlist. For a human-generated text that has no knowledge of the greenlist rule, a token will be from the greenlist with the probability $\gamma \leq 0.5$, the proportion of the greenlist size compared to the entire vocabulary. Without the knowledge of the greenlist (null hypothesis), the number of tokens in the greenlist (g) follows a binomial distribution. (Kirchenbauer et al., 2023a) used the normal approximation to the binomial distribution to compute the z-statistics for a text with T tokens: $z = \frac{g-\gamma T}{\sqrt{\gamma(1-\gamma)T}}$.

Next, we further analyze how bit-width of the message and radix affect detection performance. Our analysis stems from the observation that as we increase the bit-width the detection score for the non-watermarked text increases more rapidly than that of the watermarked text. Consequently, the difference in the two scores *decreases* as larger bit-width is used, leading to reduced seperability. The results are in Fig. 6. Notice that the difference between the scores of watermarked and non-watermarked texts decreases for larger bit-width.

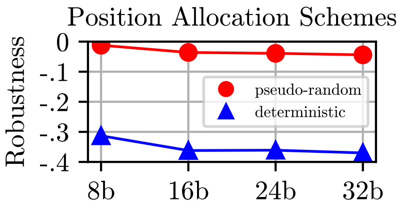


Figure 5: Performance difference between watermark extraction with and without corruption. "Deterministic" denotes sequentially encoding each position of the message as done in Ziegler et al. (2019) in the `Greenlist` framework. Mixing 20% of non-watermarked text makes the bit accuracy of sequential encoding scheme nearly random.

To grasp a hint of what is going on, we do away with the language model and other complexities by modeling this only through statistical distributions. To recap, our detection statistic (Eq. 1) was computed by aggregating the number of tokens in each position of the message. Letting C_i as the number of tokens in the colorlist for the position i, we can write the aggregated form as

$$C = C_0 + \cdots + C_{p-1} \overset{H_0}{\sim} \text{Binomial}(T, \gamma) \quad (2)$$

However, note that during decoding the ground truth message is unknown and thus, is predicted by taking the colorlist that has the max number of tokens. This is problematic when decoding for non-watermarked text as it biases the statistic to be higher when bit-width is increased. We let $W_i = [w_0, \ldots, w_{r-1}]$ be the number of tokens in r colorlists (strength of watermark) for position i. For a non-watermarked text, we can assume that this is a random variable with equal probability for each colorlist

$$W_i \sim \text{Multinomial}(n_i, [\gamma \cdots \gamma]) \quad (3)$$

where n_i is the number of tokens allocated to position i. Our decoding method takes the maximum cell value of this, which makes itself a random variable:

$$W_i^{\max} = \max(W_i) = \max([w_0, \ldots, w_{r-1}]). \quad (4)$$

Our final statistic used for our detection score is

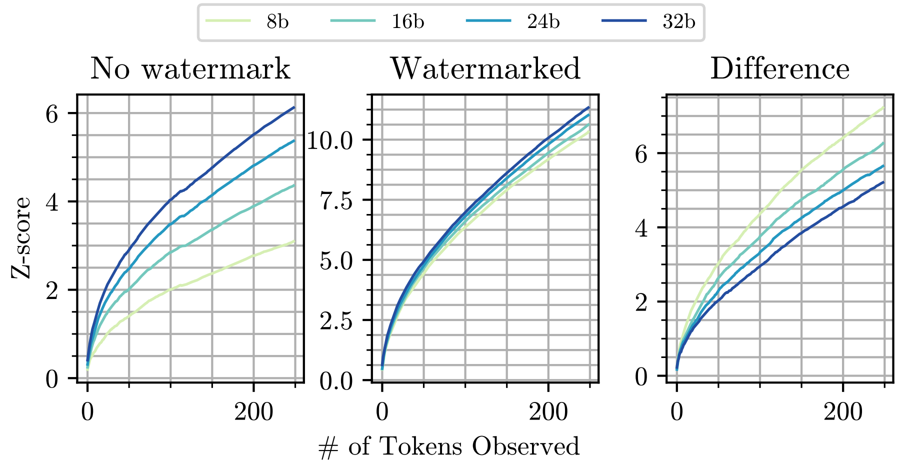


Figure 6: The detection scores of non-watermarked texts, watermarked texts and their difference as a function of number of tokens observed. We see that the difference in the scores decreases as bit-width increases, leading to reduced seperability.

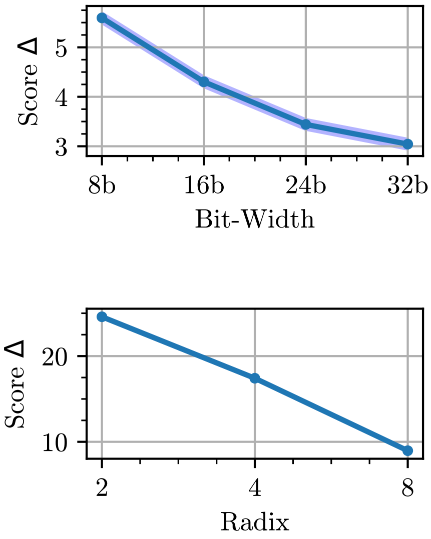


Figure 7: Simulation of the difference between (unormalized) scores for watermarked and non-watermarked multinomial distributions. Higher score signify higher seperability, hence higher detection performance. We use ϵ=0.1. For right, we use γ=.125 to allow more radix.

the sum of this variable over the entire positions:

$$W^{\max} = \sum_{i}^{p} W_i^{\max} \tag{5}$$

We see that our statistic is dependent upon the number of candidates when selecting the maximum cell (i.e. radix) through Eq. 4 and the number of positions (i.e. bit-width) through Eq. 5.

To verify the effect of bit-width and radix on the detection score, we compare the difference in the statistics for a uniform multinomial distribution, which signify non-watermarked text, and a multinomial distribution with a slightly modified probability $[\gamma + \epsilon, \gamma, \ldots, \gamma]$ to signify the added bias term for the watermarked distribution. We sample 1000 samples of $W^{\max}$ and compute the difference in the detection scores for the two distributions. The results in Fig. 7 corroborate that an increase in bit-width / radix decreases the separability of the detection scores.

In an attempt to overhaul this, we tried computing the likelihood of W_i^{rm} before aggregating them using an approximation of (Levin, 1981) (More details in the next section). However, this only led to on par or slightly worse performance. This may be because n_i is small for cases when T is small compared to the length of the message. Other than this, some of the approaches we attempted were:

- Computing test statistic per position or weighting the statistic of each position with n_i before aggregating.
- Computing the p-value of the binomial random variables rather than using the normal approximation, i.e. regularized incomplete beta function.
- Computing the p-value under the null hypothesis that the distribution of the colorlists follows a uniform distribution, i.e. Chi-square Goodness of Fit test

All the approaches either led to on-par or slightly worse results.

A.2.2 Approximating Max Multinomial Cell Distribution

We used the approximation of (Levin, 1981) for modeling the distribution of the maximum cell frequency. For completeness, we present the steps used for the approximation adapted to our case. For a multinomial distribution with sample size N and probability vectors $[p_0, \ldots, p_{r-1}]$, Let a be the maximum cell value, then the cumulative distribution function of having a maximum value of a can be approximated for any real number $s > 0$

$$P(a) = \frac{N!}{s^N e^{-s}} \{\prod_i^{r-1} P(X_i \leq a)\} P(W = N) \tag{6}$$

where $X_i \sim$Poisson(sp_i) and $W = \sum_i^{r-1} = Y_i \sim$Truncated Poisson$(sp_i)$ with range $0, 1, \ldots, a$. Following Example 1 of (Levin, 1981), we set $s = N$ and use Stirling's approximation for $N!$. We also approximate W using the normal approximation to the Poisson distribution.

A.3 List Decoding and Other Techniques

The decomposition of the message into each bit position bounds the computation during decoding to the number of tokens. This allows MPAC to output a *list* of most likely messages without exhaustively considering all the possible messages. We alter our decoding function to output candidate messages sorted by the level of confidence. Denoting the predicted message for position i by $\hat{m}$, and the observed number of tokens in the colored list (strength of the watermark) by $w = \mathbf{W}_i[\hat{m}]$, the confidence of $\hat{m}$ should be higher if w deviates from the expected mean under the null hypothesis that all colored lists are equally likely to be sampled. We define confidence at position i as $c_i \propto \Pr(W_i^{\max} \leq w | H_0)$ where $W_i^{\max}$ is the maximum cell value of $W_i \overset{H_0}{\sim}$ Multinomial$(T_i, [\gamma \cdots \gamma])$ where T_i is the number of tokens assigned to position i. The distribution of $W_i^{\max}$ is approximated using techniques from Levin (1981) (See Appendix A.2.2).

Our algorithm can be parameterized by the confidence bound on each position:

- Input: Best prediction $\hat{\mathbf{m}}$ found by majority voting via Alg. 1, confidence bound c_0
- Output: $\hat{\mathbf{m}}_1, \cdots, \hat{\mathbf{m}}_{|\mathbb{L}|} \in \mathbb{L}$ whose predictions are altered on positions with confidence under c_0

	Bit Accuracy		
δ	0.5	1	2
No feedback	.626	.766	.948
$\tilde{\delta} = \delta + 1$	.769	.860	.960

Table 1: Results for using feedback for adapting bias on T=100,b=8

	Accuracy Gained			
	8b	16b	24b	32b
c_i-sorted list	1.1%	3.7%	6.0%	5.6%
Random list	0.6%	0.4%	0.5%	0.3%

	Latency (seconds/250 tokens)				
	0b	8b	16b	24b	32b
Encoding (7.9)	8.19	7.98	8.01	7.96	8.24
Decoding (.09)	.08	.09	.09	.09	.10

Table 2: Comparison of absolute improvement in bit accuracy when using confidence-based list decoding and random list.

Empirically, we determine c_0 by constraining $|\mathbb{L}|$. Note that since $\hat{\mathbf{m}}$ is always the most confident message, we comprise $\mathbb{L}$ with the next confident messages. To do this, we greedily alter the positions with the lowest confidence to the colorlist with the second largest number of tokens. Note that this list decoding technique is not unique to ours and can be applied to other methods as long as the decoding stage is computationally feasible.

A.3.1 Results

We show absolute accuracy gained using confidence-based list decoding ($|L|$=16) compared with random decoding. We further compare the encoding and decoding latency for sequences with $\sim$ 250 tokens using a single Nvidia A100 when using an additive left hash scheme with context width 1. The results are in Table 2.The latency *does not* proportionally increase with message bit length, making it scalable to long messages. When using an efficient hashing scheme watermarking has a negligible increase in both encoding and decoding compared to vanilla generation, which requires 7.9 seconds and 0.09 seconds, respectively.

A.3.2 Message Correction with Feedback

One key characteristic of our $p(y|x)$ is that we can instantly check whether the message was correctly transmitted by examining whether the sampled token is in the correct colorlist. This property resembles the settings of error correcting codes with feedback, in which the receiver can send feedback to the sender after receiving the message(Berlekamp, 1964; Gupta et al., 2023). One can take advantage of this property by adapting the magnitude of the bias during encoding when the majority vote of a given position differs from the actual message.

We provide some preliminary results of taking advantage of feedback during message encoding. One simple scheme is adapting the magnitude of the bias so that when the message is not correctly encoded, we enlarge the bias. Concretely, for $0 \leq t \leq T$ that is allocated to position p, if the current max colorlist does not match the actual message content, i.e. $\mathbf{m}[p] \neq \text{argmax}_j \mathbf{W}[j]$, we use a larger bias $\tilde{\delta} > \delta$. The results in Table 1 show that all lead to an increase in the multi-bit accuracy. However, we observed this came with a degradation in text quality measured by automatic metrics. We leave finding better methodology as a future work.

A.4 Extending `MPAC` to other methods

Block Allocation Instead of allocating a single position as done in `MPAC`, we can allocate a block of message, after which techniques of `Cyclic-Shift` can be used to encode the block message. This ensemble approach enables the prior works to embed longer messages. Deriving it name from Position Allocation, we dub this as Block Allocation.

```
Block Allocation

  1. Compute $s = f(X_{t-h:t-1})$.
  2. Chunk message in $n$ blocks. $\mathbf{m} = [\mathbf{m}_1, \dots, \mathbf{m}_n]$ where $\mathbf{m}_n \in \Sigma^{\frac{b}{n}}$
  3. $p \leftarrow$ sample($[n]$) using $s$ as seed.
  4. Run Cyclic-Shift with message as $\mathbf{m}_p$
```

At decoding, we predict the message for each block and concatenate them. As a preliminary experiment, we use `Block Allocation` with `Cyclic-Shift` using n=4 blocks. `Block Allocation` can embed 24-bit messages with .901 bit accuracy (c.f. `Cyclic-Shift` achieves .775) and 32-bit with .871 accuracy.

Extension to Other Zero-bit Watermarking Aaronson and Kirchner (2023) is another line of work in zero bit watermarking that modifies the sampling process by generating a secret vector $\mathbf{r} \in [0,1]^{|\mathcal{V}|}$ based on the random seed s. Given the original probability distribution $\mathbf{p}^{|\mathcal{V}|}$, the token with both large p_v and $\mathbf{r}_v$ is favored by choosing

$$x = \text{argmax}_{v \in \mathcal{V}} \mathbf{r}_v^{1/\mathbf{p}_v}. \tag{7}$$

We can adapt our position allocation method to this as well by preceding the above step with position allocation. Then, the secret key can be modified depending on the message content by the following rule:

$$\mathbf{r} = \begin{cases} \mathbf{r} & \text{if } \mathbf{m}[p] = 0 \\ \mathbf{1} - \mathbf{r} & \text{if } \mathbf{m}[p] = 1 \end{cases} \tag{8}$$

where $\mathbf{1}$ is a vector with 1 in all the elements. Analogous to favoring mutually exclusive colorlists, this allows favoring different tokens depending on the message content. At decoding time, we can similarly maintain a counter for each position for the two cases.

A.5 Implementation, Hardware, Code Details

We follow (Kirchenbauer et al., 2023a) in most experimental settings. For the hashing scheme in the main paper, we use LeftHash scheme with context window $h = 1$. In the appendix, we provide results for the SelfHash scheme. For further discussions regarding the hash scheme see Appendix A.7. To generate sequences with the desired token length T, we generate with the max token set as T. Then we filter out the watermarked and non-watermarked sequences with token lengths under $T_{\text{low}} = T - \tau$. We set τ=25, except for the LFQA dataset, which was set to τ=50 as it has instructions that state to generate answers with 200-300 words. For generation, we use sampling with a temperature of 0.7. For each bit-width, a new set of generations had to be made as the length of the message differed.

For the copy-paste attack, we sample a random non-watermarked text and truncate to have the same length. Then, a position is randomly sampled to insert a p percentage of the watermarked text into the non-watermarked text. We experiment with varying degrees of p (10%$\sim$ 50%).

We used `float16` for all our models during generation. Our experiment was run on a single NVIDIA A100. For T=250, generating around 500 watermarked and non-watermarked samples took approximately 200 minutes for the left hash scheme. When using the self-hash scheme, this took significantly longer ($\sim$ 550 minutes). Our implementation is based on the official codebase of (Kirchenbauer et al., 2023a): `https://github.com/jwkirchenbauer/lm-watermarking`. We will be releasing our code to reproduce our experiments.

For baselines, we use the official repository of

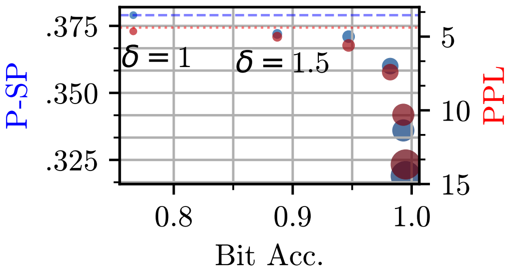


Figure 8: Text quality vs. δ across bias@T=100,b=8

Fernandez et al. (2023a)[4] and (Wang et al., 2023a)[5]. For Message-Hash , following the same configuration presented in their work (GPT-2 as the proxy model) cannot watermark the outputs of LLaMA-based models due to the difference in the tokenizers. Consequently, we resort to the Vanilla Marking scheme. This makes all the other factors equivalent for the three methods (MPAC, Message-Hash, Cyclic-Shift) except the message encoding function $\mathcal{E}$ described in §3. Besides, we believe this has little to no effect on the watermark performance, since the use of proxy model is intended to enhance the quality of the text (in terms of perplexity) rather than the strength of the watermark.

A.6 Metrics: Bit Accuracy, Text Quality

Text Quality Metrics Using P-SP, we measure the semantic similarity between the human text and watermarked text given the same prompt. While human evaluation is considered to be the golden label, our main purpose is to show that our multi-bit watermarking does not degrade the quality compared to zero-bit watermarking. Moreover, the effect of watermarking on the text quality *compared to no watermarking* shows promising results in human evaluations when sufficiently large models are used for open-ended generation by Kirchenbauer et al. 2023b (Appendix A.2 and A.9). Additionally, Fernandez et al. (2023a) also demonstrate that watermarking does not lead to noticeable performance degradation even on benchmarks with non-ambiguous answers such as coding and math especially with sufficiently larger models, albeit at a small bias. We further show in Fig. 8 the trade-off curve between bit accuracy and text quality. The size indicates the magnitude of bias ({1, 1.5 2, 3, 4, 5}) and horizontal dashed lines indicate non-watermarked counterparts. Analysis of text quality shows $\delta = 2$ lies at a good trade-off point.

Bit Accuracy for Multi-bit Watermark In our experiments, we used bit accuracy (error) as our metric for multi-bit watermark performance. This is a general metric that is independent of the downstream application or the encoding scheme. However, computing the exact match of a message should be done dependent on the context. To illustrate this, we start with some examples. First, consider the case where the encoding scheme to identify users is simply assigning a message to each user. Then, by embedding 4-bit message one can encode 2^4 different users : $\mathbf{m}$='0000' for Bob, $\mathbf{m}$='0001' for Alice, and so on. For such a scenario, one might be interested in computing the exact match of the 4-bit message, also known as the packet error ratio. While this encoding scheme enables tracing back to the exact users at low load capacity, this is extremely inflexible as it cannot handle influx or outflux of users.

Conversely, one can turn to a more flexible encoding scheme by encoding each character. Using UTF-8, this requires 8 bits per character, which would mean 40 bits is required just for encoding 5 character user ID. For this scenario, one might be more interested in computing the packet error ratio of each character or the entire 40-bit message. A more realistic encoding scheme will be somewhere between the middle, which uses a more efficient representation, e.g. by merging often-used bytes as done in Byte pair encoding (Gage, 1994). Added with error correction codes such as the Reed-Solomon code (Wicker and Bhargava, 1999), this allows a more robust representation. Since focusing on a single type of encoding scheme – and more fundamentally, what information to embed – narrows down the potential applications, we present bit accuracy in our main experiments as done in previous works in the literature (Zhu et al., 2018; Luo et al., 2020; Yang et al., 2022; Yoo et al., 2023; Fernandez et al., 2023b). For T=250, the packet error ratio for the 8-bit message was 7.1%, which is +5.7 % higher than the bit error rate. With 16-list decoding, this is reduced to 2.4%.

Another metric considered in Table III of Fernandez et al. (2023a) was combining the detection scheme and packet error ratio. In this scenario, they assume an encoding scheme of assigning each user to a single message and compute the percentage of finding the exact user given a fixed false positive rate. At FPR=$1e^{-3}$ and using 8-bit message (256

[4] https://github.com/facebookresearch/three_bricks

[5] https://github.com/lancopku/codable-watermarking-for-llm

Ratio Sampled Position (Sorted)				
LeftHash (h=1)	0.319	0.251	0.235	0.195
SelfHash (h=4)	0.264	0.257	0.242	0.238

Table 3: Ratio of the sampled position for b=8,r=4 (four positions total) for the two hashing schemes for position allocation.

users), we can correctly identify 90.5% cases. Our true positive rate was computed by the setting used in Table 7.

A.7 Discussion on the Hashing Scheme

The hashing scheme for generating the seed plays a significant role in watermarking. For our MPAC, the hashing scheme is employed once for position allocation and once for permuting the vocabulary list. Here, we discuss some implications of the design choices.

To recap, the function $f(X_{t-h:t-1})$ is used to hash h most recent tokens before generating the t^{th} token. Following the terminology of Kirchenbauer et al. (2023b), LeftHash takes the leftmost token, while SelfHash is determined in a slightly more complex way that is dependent on the t^{th} token (see Algorithm 1 of Kirchenbauer et al. (2023b)). The context width and the hashing scheme determine robustness and quality (diversity) trade-offs. For our experiments, we use the two configurations (LeftHash with h=1 and SelfHash with h=4) proposed in the previous work found to be effective in the two aspects without further fine-tuning.

As expected by the trade-off, the perplexity was slightly higher for LeftHash compared to SelfHash (5.1 vs. 4.9 on average for 250 tokens), while P-SP was at the same level. One clear distinction between the two schemes was the encoding time latency. As SelfHash iteratively searches for tokens, this took significantly longer than the LeftHash scheme, which had nearly no overhead compared to no watermarking (appendix A.5 and Table 2). In addition, we observed that the sampled positions were not uniform for LeftHash with $h = 1$ as shown in Tab. 3 due to the reduced diversity of the tokens in the context width. Despite this, the multi-bit performance was similar for the two schemes (Table 12 and 13). A possible direction for improvement may be using different hashing schemes for position allocation (more robust) and vocabulary partitioning (more quality-focused).

A.8 More on Robustness: Other Attacks, Detection

We also test our watermark against DIPPER (Krishna et al., 2023), which is a specialized paraphrasing model. DIPPER is parameterized by two scalers, which control lexical diversity and token order diversity. We first present the results across bit-width with a lexical diversity of 20 (out of 100). We see that the watermark fares considerably better than using GPT-3 attack in Table 4.

To see the magnitude of semantic drift of the two paraphrasing methods, we compute the P-SP between the original watermarked text and its paraphrased counterpart. We also compute the absolute change in the number of words. Table 5 demonstrates that paraphrasing using GPT-3.5 changes the semantic and the number of words greater than the setting used in Table 4, which may explain why the multi-bit watermark performance is lower for GPT-3.5. When we control the diversity parameters of DIPPER, this is able to degrade the watermark performance as well as GPT-3.5.

Some other forms of possible attacks considered in the literature are word substitution, insertion, and deletion. Word substition is very similar to the copy-paste attack considered in the main paper. Our watermark scheme is also robust to partial insertion and deletion of words as MPAC relies on the local context to synchronize the positions of the message and the ordering of the vocabulary.

Robustness of zero-bit Watermark Here we provide results for the detection performance under corrptuion. We use the copy-paste attack with the attack percentage ranges of {10%, 20%, 30%, 40%, 50%} and compare the AUC vs. number of tokens observed curve similar to Fig. 9. While the detectability is noticeably affected, the final AUC is recovered to a large degree only after observing 250 tokens. In order of the attack strength, the final AUC's are .992, .987, ,980, ,971, .942, respectively. For the zero-bit counterpart, all the scores are over .990.

A.9 Ablations on Datasets and Model Sizes

We show additional results on other datasets and model sizes in Fig. 11. C4 news-like subset is the dataset we used for our main experiment. "Long-form Question-Answering" (LFQA) is a dataset curated by Krishna et al. (2023) on the Reddit's "Explain Like I'm Five" (ELI5) forum. The Essays dataset comprises paris of instructions

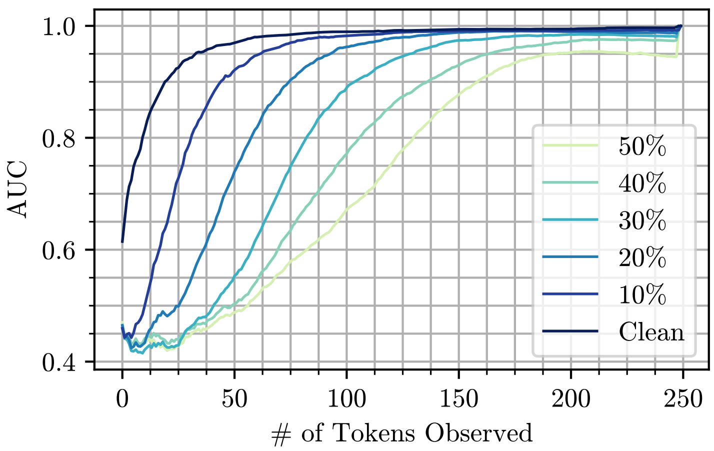


Figure 9: AUC vs. number of tokens observed when corrupted with copy-paste attack for 8-bit message.

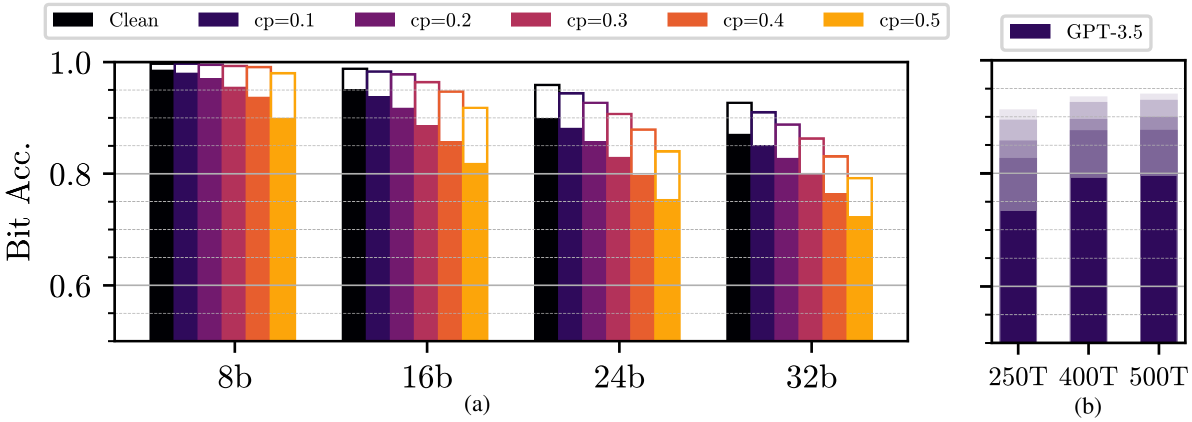


Figure 10: Corrupted bit accuracy for (a) copy-paste attack controlled by the human text percentage at T-250 and (b) paraphrasing attack using GPT-3.5 embedding 8-bit messages at varying token lengths. For (b), we show multiple sizes of list ($|L| \in$ {2, 4, 8, 16}) by color gradation as 8-bit has relatively small output space.

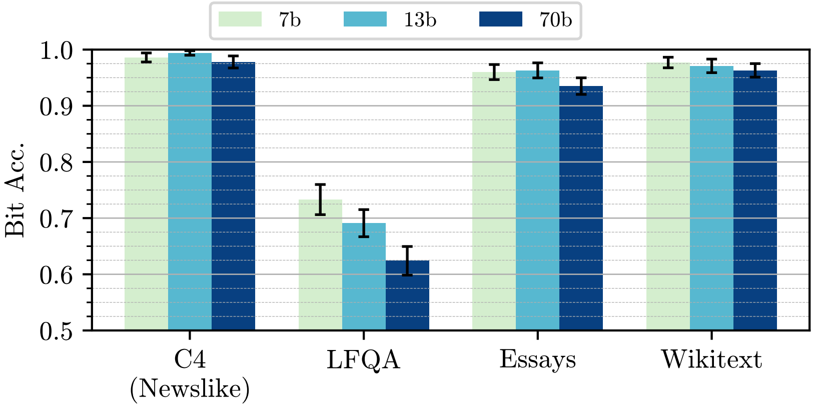


Figure 11: Multi-bit performance across datasets and model sizes.

Bit Acc. after Paraphrasing with DIPPER				
Bit-width	8	16	24	32
Best Prediction	.922 (.13)	.825 (.12)	.778 (.12)	.736 (.10)
16-List Decoded	.982 (.05)	.924 (.08)	.864 (.10)	.801 (.09)

Table 4: Robustness under paraphrasing using DIPPER (Lexical diveristy=20)

		DIPPER			
	GPT-3.5	Lex.=20	Lex.=40	Lex.=60	Lex.=60 Ordering=60
P-SP	.815	.933	.897	.844	.827
Absolute Change in # of Words	36	13	16	19	20
Bit Acc.	.733	.922	.849	.757	.719

Table 5: Comparison of the two paraphrasing method on text quality.

Model	Bit Acc.
LLaMA-2-7b	.986 (.06)
+ Chat	.922 (.13)
Mistral-7b	.987 (.06)
+ Chat	.977 (.08)
OPT-1.3b	.982 (.07)

Table 6: Performance on other pretrained models and their SFT and RLHF variants (*Llama-2-7b-chat-hf* and *Mistral-7B-Instruct-v0.1*). Results on b=8, T=250.

True Positive Rate					
Bit-width	0	8	16	24	32
FPR=1e^{-2}	0.999	0.986	0.974	0.964	0.958
FPR=1e^{-3}	0.997	0.974	0.956	0.943	0.915
FPR=1e^{-4}	0.997	0.96	0.934	0.905	0.88
FPR=1e^{-5}	0.994	0.951	0.907	0.851	0.793

Table 7: True positive rate at a fixed false positive rate across bit-widths. We use $\sim$ 500 positive sample and $\sim$100,000 negative samples. We only count the unique tokens following (Kirchenbauer et al., 2023a; Fernandez et al., 2023a). This has an effect of removing outlier human text samples that have exceptionally high scores.

and essays (Schuhmann, 2022). Wikitext (Merity et al., 2016) comprises Wikipedia article. We use the 'wikitext-2' subset. For LFQA, we use the finetuned version, LLaMA-2-Chat, specialized for chats as they explicitly have questions or instructions as prompts.

It is apparent that the watermark performance is affected by the text distribution. When the entropy of the vocabulary distribution is low (low diversity), there is little room for encoding the message with a fixed bias, which has been observed in zero-bit watermarking as well where the watermark performance suffers for low entropy text distributions such as coding (Lee et al., 2023; Kirchenbauer et al., 2023b). For our multi-bit case, this means the load capacity is inherently low for such text distributions. This is especially observed for LFQA, in which the model consistently starts the response by restating the question (e.g. *"The reason for [Question] is . . . "*). Across the model scale, the trend is not as apparent although we found that the largest model consistently has a lower performance. This hints that the entropy of the vocabulary distribution is lower for the largest model, which might explain the higher text quality in general when we increase the model size. Larger models might have the capacity to form high-quality sequences even when the text distribution is altered by increasing the entropy via temperature or explicitly increasing the magnitude of the bias during watermarking. We leave this as a future work.

A.10 Tabular Results

Here we present the numerical results of the experiments done in the main paper. Numbers in the parenthesis signify the standard deviation.

- Table 8 and 9 ↔ Figure 2 show the comparisons with baseline methods.
- Table 10 ↔ Figure 8 show the relationship between δ vs. text quality and watermark strength.
- Table 11 ↔ Figure 3 left compare the different configurations of radix and colorlist proportion.

- Table 12 ↔ Figure 3 left show the multibit watermark performance on a fixed token length.
- Table 13 ↔ Figure 3 right show the multibit watermark performance on a fixed load capacity (bits per token).
- Table 14 ↔ Figure 10a show the multibit watermark performance under copy-paste corruption.
- Table 15 ↔ Figure 10b show the multibit watermark performance under paraphrasing.

A.11 Generation Samples

We show below in Table 16 generated samples.

	B=8,T=250			
Copy-Paste (p)	Clean	cp=10%	cp=30%	cp=50%
Ours	.986 (.06)	.981 (.07)	.956 (.10)	.900 (.13)
FCT+EMS	.979 (.10)	.943 (.17)	.858 (.24)	.800 (.28)
FCT+Greenlist	.995 (.05)	.988 (.08)	.970 (.12)	.908 (.20)
CTWL	.977 (.11)	.973 (.12)	.951(.16)	.858(.24)

Table 8: Comparison of multibit watermark performance with other methods on clean and corrupted settings. For corruption, we use the copy-paste attack. *The load capacity of FCT+Greenlist is limited to 15-bit.

	B=16,T=250				**B=24,T=250**			
Copy-Paste (p)	Clean	cp=10%	cp=30%	cp=50%	Clean	cp=10%	cp=30%	cp=50%
Ours	.951 (.07)	.939 (.08)	.887 (.09)	.819 (.12)	.899 (.09)	.882 (.09)	.830 (.10)	.755 (.11)
FCT+EMS	.905 (.20)	.811 (.26)	.702 (.26)	.601 (.23)	.775 (.26)	.729 (.24)	.633 (.23)	.513 (.13)
CTWL	..936 (.18)	.909 (.20)	.810 (.26)	.614 (.22)	.876 (.22)	.828 (.25)	.663 (.26)	.516 (16)

Table 9: Comparison of multibit watermark performance with other methods on clean and corrupted settings.

δ	0.5	1	1.5	2	3	4	5
Bit Acc.	.626 (.19)	.766 (.18)	.887 (.15)	.947 (.11)	.982 (.08)	.993 (.05	.995 (.05)
P-SP (w/ reference)	.385 (.15)	.379 (.15)	.372 (.15)	.371 (.15	.360 (.14)	.336 (.13)	.319 (.13)
P-SP (w/ non-wm.)	.526 (.18)	.460 (.16)	.433 (.15)	.417 (.15)	.388 (.14)	.349 (.14)	.330 (.13)
PPL	4.41 (1.5)	4.64 (1.8)	5.01 (2.0)	5.6 (2.0)	7.41 (2.7)	10.3 (4.1)	13.67 (5.9)

Table 10: Bit accuracy and text quality on embedding 8 bit-width message on T=250 across various magnitudes of bias δ.

Bit Accuracy @ T=250				
Bit	8	16	24	32
γ=.25,r=4	.986 (.06)	.951 (.07)	.900 (.09)	.871 (0.08)
γ=.25,r=2	.966 (.07)	.905 (.08)	.858 (.08)	0.820 (.08)
γ=.50,r=2	.978 (.05)	.922 (.07)	.875 (.08)	0.849 (.07)

Table 11: Multibit watermark performance measured by bit accuracy for varying configurations of colorlist proportion and radix.

Bit Acc. @ T=250				
Bit	8	16	24	32
LeftHash($h = 1$)	.986 (0.06)	.951 (.07)	.900 (.09)	.871 (0.08)
SelfHash($h = 4$)	.976 (.08)	.905 (.08)	.895 (.09)	.862 (.09)

Table 12: Bit accuracy for two different hash schemes for a fixed token length.

Bit Acc. @ BPT=.064					
T	63	125	250	500	1000
Bit	4	8	16	32	64
LeftHash($h = 1$)	.961 (.13)	.958 (.09)	.951 (.07)	.913 (.08)	.846 (.09)
SelfHash($h = 4$)	.952 (.13)	.953 (.10)	.945 (.08)	.911 (.08)	.850 (.08)

Table 13: Bit accuracy for two different hash schemes for a fixed bits per token.

		Copy-paste Attack					
Attack Strength		Clean	10%	20%	30%	40%	50%
8-bit	Best	.986 (.06)	.981 (.07)	0.971 (.08)	.956 (.10)	.938 (.12)	.900 (.13)
	+16-List	.997 (.02)	.997 (.02)	.995 (.03)	.993 (.03)	.991 (.04)	.980 (.05)
16-bit	Best	.951 (.07)	.939 (.08)	.918 (.09)	.887 (.09)	.858 (.11)	.819 (.12)
	+16-List	.988 (0.04)	.983 (.04)	.978 (.05)	.964 (.06)	.947 (.07)	.918 (.08)
24-bit	Best	.899 (.09)	.882 (.09)	.858 (.10)	.830 (.10)	.797 (.11)	.755 (.11)
	+16-List	.959 (.06)	.944 (.06)	.927 (.08)	.907 (.08)	.879 (.09)	.840 (.09)
32-bit	Best	.871 (.08)	.851 (.09)	.828 (.09)	.801 (.09)	.765 (.09)	.723 (.1)
	+16-List	.927 (.07)	.910 (.08)	.888 (.08)	.863 (.08)	.831 (.09)	.792 (.09)

Table 14: Robustness when certain percentage of human text is mixed into the watermarked text.

		GPT-3.5 Paraphrasing		
Token Length		250T	400T	500T
8-bit	Best	.733 (.19)	.792 (.19)	.795 (.19)
	+2-List	.825 (.16)	.874 (.15)	.875 (.15)
	+4-List	.856 (.14)	.894 (.13)	.898 (.13)
	+8-List	.893 (.12)	.924 (.11)	.928 (.11)
	+16-List	.911 (.10)	.934 (.09)	.939 (.09)

Prompt

"As an expert copy-editor, please rewrite the following text in your own voice while ensuring that the final output contains the same information as the original text and has roughly the same length. Please paraphrase all sentences and do not omit any crucial details. Additionally, please take care to provide any relevant information about public figures, organizations, or other entities mentioned in the text to avoid any potential misunderstandings or biases."

Table 15: Robustness when paraphrased with GPT-3.5.

Prompt	Non-Watermarked	Watermarked
A blind man in his 60s is searching for the young man who pulled him back from an approaching train and saved his life last Thursday, Nov. 9. Mike Wyatt stood at the stairs to the Peoria train station in Aurora, Colorado, on Monday, looking for the young man who saved his life days earlier	“He told me he was 20 years old and he was going to college," Wyatt told reporters. “He said he was going to be a doctor." Wyatt, who is blind and has been for most of his life, was at the train station ready to catch a train to the airport when he slipped and almost fell in front of an oncoming train.[...]	and has given him new life. Inside the station, he spoke to the counter clerk, hoping the clerk might have information that might lead him to the young man, who jumped at the chance to help Wyatt when he saw he was in danger. But, he said, he found nothing.[...]
[...] Shortly after the release of the lead single “Life’s About to Get Good,” it hit number one on the Billboard 200. Of course, fans were simply dying to see how their favourite country icon was doing, so	the album was released on CD, vinyl, and digital. But, Shania took things to the next level and released a deluxe version with added tracks. [...]	the song was released a few months earlier in June. “OK so I am sooooo proud that my song Life’s About to Get Good is now the number one song on Billboard country,” she tweeted back in June.[...]
[..] Amazon’s offices will be located within an already-established special tax district where a portion of the property tax revenue goes toward infrastructure improvements such as parks and wider sidewalks. The incentive agreement says that half of any new revenue from that district starting in 202	3 would go to Amazon to help pay for employee benefits and transportation, but the county has said it would not be required to pay those benefits.[...]	7 would go to Amazon through a 15-year payment schedule. This amount would grow to more than 60 percent if the company exceeds 10.24 million square feet of office space. Amazon is expected to reach that level in 2027.[...]
[...] "Below the surface of the Sun a dynamo process is working creating magnetic field," Harra explained in an email interview. "When this becomes buoyant it can rise to the surface of the Sun,	and this is what we call a sunspot." The new research shows that the initial explosion was actually a series of smaller blasts that combined to create the X-7 flare.	and form coronal holes and sunspots. This happens in a fairly homogeneous area on the Sun, so there can be several sunspots with a single magnetic field underneath.
[...]The merge listing the most important changes to Linux 3.8’s sound subsystem includes some other changes to audio drivers. The kernel now includes a driver for human interface devices (HIDs) that use I2C (1, 2 and others), using the "HID over I2C" protocol designed by Microsoft and implemented in WindowsÂ	7 and later versions of the operating system. The kernel now has a driver for the Samsung Galaxy S III smartphone’s touchscreen (1, 2 and others), and the rt2800usb driver, for the RaLink RT2800USB WLAN chip, now supports devices that have Bluetooth 3.0 (1, 2).[...]	7 and Windows Vista. The drivers can read out data from HIDs and set the appropriate commands to them. An example of such a device is a BT-USB adapter. The sound subsystem now supports two new, high-quality audio codecs (1, 2):[...]

Table 16: Randomly sampled examples of watermarked texts on the C4 newslike subset with 100% bit accuracy. Samples are truncated for readability.