

---

# EXTENDING LOGITS-BASED WATERMARKING SCHEMES TO MITIGATE STEALING ATTACKS

---

**Mikko Tripakis**

Khoury College of Computer Sciences  
Northeastern University  
Boston, MA, USA  
tripakis.m@northeastern.edu

December 8, 2024

## ABSTRACT

Logits-based watermarking schemes for large language models are robust to moderate attacks such as editing, paraphrasing, and token replacement. However, watermark stealing attacks pose a serious threat to these schemes, which use a fixed-width token context to seed pseudorandom generator functions. We propose a variable-width context mechanism to increase robustness against stealing attacks while maintaining quality and detectability of watermarked text. We implement our mechanism on the KGW-SelfHash variant of the watermark proposed by Kirchenbauer et al. [1] and evaluate it against the watermark stealing attack developed by Jovanovic et al. [2]. We find that our mitigation successfully degrades the attack’s effectiveness while maintaining high quality and detectability of watermarked text.

## 1 Introduction

Large language models (LLMs), such as ChatGPT, have demonstrated remarkable abilities in generating text, writing code, and completing tasks with human-like proficiency [3, 4]. However, their widespread adoption raises concerns about potential misuse. Malicious actors could exploit these systems for social engineering, election interference, and the dissemination of fake news [5, 6]. For instance, AI-generated deepfakes have been identified as tools that could undermine democracy and public trust [7]. Additionally, LLMs can be used to cheat on academic assignments, compromising educational integrity [8]. The proliferation of AI-generated content also complicates data collection for future model training, as synthetic data may be of lower quality and require detection and exclusion [9, 10].

Therefore, the ability to detect machine-generated text becomes an important component of harm reduction for LLMs. Many solutions for detection exist, from creating databases of LLM completions to training discriminator models like DetectGPT and GPTZero [11, 12]. However, storing LLM completions in databases raises concerns of user privacy, and discriminator models tend to have high false positive rates which make them unreliable for most use cases [11, 12].

LLM watermarking, inspired by watermarking for images, video, and audio, is a field of work which attempts to embed hidden signals into machine-generated text indicating that it was produced by an LLM. Watermarking schemes seek to achieve reliable detectability, robustness to edits and adversarial attacks, and maintain a high quality of generated text. Logits-based watermarks are the state-of-the-art in LLM watermarks. They work by biasing model token distributions at inference time to generate certain tokens more than others, and providing algorithms to detect the watermark through a statistical test [1, 13, 14].

Though current logits-based schemes are robust to most naive attacks such as paraphrasing or token replacement, their weakness lies in attackers being able to extract the watermarking rules, rendering this robustness to weaker attacks less meaningful. Logits-based schemes have a context parameter that determines how "random" the biasing of token distributions is during generation of watermarked text. Kirchenbauer et al. claim that a larger context width increases robustness against spoofing attacks, which attempt to add a watermark to benign text, but makes scrubbing attacks, which attempt to remove watermarks, more effective [1]. However, Jovanovic et al. demonstrate that even with a context

width deemed to be optimal by prior work, watermark stealing attacks still achieve high success rates [2]. Stealing attacks pose a serious threat to watermarks, as they allow the attacker to gain knowledge of the watermarking rules of the scheme, enabling both scrubbing and spoofing attacks.

In this work, we explore the robustness of logits-based watermarks. We propose a modification to these schemes which introduces more randomness to the watermarking process by pseudorandomly varying the context width parameter used to apply the watermark. We modify the scheme proposed by Kirchenbauer et al. [1], and test our mitigation against the watermark stealing attack proposed by Jovanovic et al. [2]. To our knowledge, no prior work has considered this modification.

## 2 Prior Work

### 2.1 Logits-Based Watermarks

Logits-based watermarks operate by biasing the distributions of model outputs [1, 13, 14]. On each generated token, the token vocabulary is partitioned into distinct *red* and *green* lists using a pseudorandom function (PRF) which is parameterized by a secret key  $\xi$  and a hash of some context of  $k$  previously generated tokens. The logits of the green-list tokens are boosted, encouraging the model to output more green-list tokens than red-list tokens. In detection, the proportion of green-list tokens in the final output is used to detect the watermark via a statistical test.

Watermarking schemes from prior work all use a fixed context width  $k$  to seed the PRF and partition the token vocabulary. This predictability renders these schemes vulnerable to watermark stealing attacks, in which attackers analyze a large number of watermarked outputs to learn the watermarking rules, even if the secret key is unknown.

### 2.2 Watermark Stealing

In this work, we focus on the watermark stealing attack proposed by Jovanovic et al. [2].

#### 2.2.1 Threat Model

**Objective** The attacker’s objective is to learn the watermarking rules of an LLM  $M$  which uses a logits-based watermark. Using this knowledge, the attacker aims to mount spoofing attacks and scrubbing attacks. In a spoofing attack, the attacker’s goal is to add the watermark to benign (not watermarked) text so that it is detected as watermarked. In a scrubbing attack, the attacker aims to remove the watermark from text without significantly modifying the structure or meaning of the text.

**Attacker Knowledge** The attacker has black-box knowledge of the watermarked language model. The attacker knows all parameters of the watermarking scheme applied to the model, apart from the secret key  $\xi$ . Importantly, the attacker has knowledge of the context width  $k$ .

**Attacker Capabilities.** The attacker has query access to  $M$ . They also have access to a corpus of responses generated with  $M$  which are not watermarked.

#### 2.2.2 Method

The watermark stealing algorithm builds an approximate model of the watermarking rules by querying  $M$ . This model relies on empirical estimates of conditional distributions that capture the likelihood of a token being in the green list given the preceding tokens. The watermark’s *context width parameter*,  $k$ , determines the number of preceding tokens used to seed the PRF in the watermarking scheme. While the PRF seed relies on the next token candidate and one of the preceding tokens, the attacker can exploit the fact that scores for partial contexts carry the same signal as scores for the full context. Since partial contexts occur more frequently in the queried text, the attacker can compute the scoring function on a larger number of samples, resulting in a more reliable estimate.

#### 2.2.3 Mitigations

Prior work considers the use of multiple secret keys to mitigate stealing attacks [1, 2]. Randomly selecting a different key from a known subset for each token generated indeed decreases the attack’s effectiveness. However, it also increases the overhead in detection, since a detection algorithm needs to test each secret key on every detected token. For this reason, the multiple-keys approach is not the most desirable mitigation.

### 3 Research Questions and Methodology

The primary goal of this work is to reduce the effectiveness of the stealing attack described in Section 2. Our intuition is to introduce more randomness into the watermark generation process, making it more difficult for an attacker to estimate the watermarking rules. We claim that the attacker’s knowledge of the context width parameter  $k$  (see Section 2.2.2) significantly enhances the attack’s effectiveness by allowing the attacker to leverage partial context information, thereby improving their estimates of the green and red lists. Thus, our method focuses on randomizing the context width on every generated token, preventing the attacker from relying on knowledge of the context and making it more difficult for them to build accurate estimates of the watermarking rules.

**Research Questions** Our work attempts to answer the following research questions, meant to assess the robustness of our method against adversarial attacks as well its utility as a watermarking scheme:

**Q1:** Is our proposed variable-width context mechanism robust against watermark stealing attacks?

**Q2:** Does our method maintain the detectability and quality of watermarked text, especially in comparison to methods which use a fixed-width context for seeding?

---

#### Algorithm 1: Text Generation with Variable Context Width Watermark

---

**Input:** prompt,  $s_{(-N_p)} \dots s^{(-1)}$ , language model  $M$

**Input:** green list size,  $\gamma \in (0, 1)$ , hardness parameter,  $\delta > 0$

**Input:** secret key  $\xi$  for the hash function

**for**  $t = 0, 1, \dots$  **do**

1. Apply  $M$  to prior tokens  $s_{(-N_p)} \dots s^{(t-1)}$  to get a logit vector  $l^{(t)}$  over the vocabulary.
2. Pseudorandomly compute the context width  $k$  using  $\xi$  and the current tokens  $s$ :  $k = \text{PRF}(s, \xi)$ .
3. Compute a hash of the previous  $k$  tokens,  $s^{(t-k)} \dots s^{(t)}$ , and use it to seed a random number generator.
4. Using this random number generator, randomly partition the vocabulary into a “green list”  $G$  of size  $\gamma|V|$ , and a “red list”  $R$  of size  $(1 - \gamma)|V|$ .
5. Add  $\delta$  to each green list logit. Apply the softmax operator to these modified logits to get a probability distribution over the vocabulary:

$$\hat{p}_k^{(t)} = \begin{cases} \frac{\exp(l_k^{(t)} + \delta)}{\sum_{i \in R} \exp(l_i^{(t)}) + \sum_{i \in G} \exp(l_i^{(t)} + \delta)}, & k \in G \\ \frac{\exp(l_k^{(t)})}{\sum_{i \in R} \exp(l_i^{(t)}) + \sum_{i \in G} \exp(l_i^{(t)} + \delta)}, & k \in R \end{cases}$$

6. Sample the next token,  $s^{(t)}$ , using the watermarked distribution  $\hat{p}^{(t)}$ .
- 

**Approach** Algorithm 1 displays our approach. The watermark is applied in largely the same fashion as the soft red-list watermark variant proposed by Kirchenbauer et al. [1]. The algorithm takes in a prompt  $s$  and language model  $M$ , a green list size  $\gamma$ , representing the fraction of the token vocabulary to include in the green list, as well as a hardness parameter  $\delta$ , indicating the strength of the watermark (how much to bias the green-list logits). The algorithm also takes a secret key  $\xi$  used to privately compute our variable context width.

For each token  $t$  to be generated, we first apply  $M$  to prior tokens to obtain the logits for  $t$ . The key step in our algorithm is step 2, where, using  $\xi$  and the current tokens  $s$ , we compute a new value of  $k$  for the current token. We then compute a hash of the previous  $k$  to seed a PRF, and partition the vocabulary into red and green lists based on  $\gamma$ . Finally, we bias the green list logits using  $\delta$  and get a probability distribution over the vocabulary, and sample the next token.

In practice, we need to ensure that  $k$  falls within some reasonable bounds. We set the minimum value of  $k$  to 1. The maximum value for  $k$  could theoretically be the length of  $s$ , although in accordance with prior work, we use a maximum value of 5 in our evaluation in order to increase robustness to attack amplification<sup>1</sup>.

We use the same detection algorithm as Kirchenbauer et al. [1], with one key difference. For each token  $t$  in the output, we first use  $\xi$  to compute the value of  $k$  that was used in generation to get the correct red and green lists for  $t$ . We

---

<sup>1</sup>Watermarks with higher context widths are more vulnerable to scrubbing attacks which replace or modify tokens in the watermarked text, aiming to reduce the number of green-list tokens in the output. With a context width of  $k$ , changing one token has the potential to impact the green-list generation for the next  $k$  tokens in the sequence. As  $k$  increases, changes to individual tokens disturb the green lists for an increasing number of downstream tokens [1].

then proceed with detection as usual, using the fraction of green-list tokens in the output to detect the presence of the watermark.

## 4 Evaluation

### 4.1 Experimental Setup

We evaluate our method by applying it to the scheme proposed by Kirchenbauer et al. [1], and attacking it with the stealing attack proposed by Jovanovic et al. [2].

**Dataset and Model** For generating watermarked text, we use a real news-like split of the C4 dataset [15], in accordance with prior work [1]. Logits-based watermarks are difficult to apply to low-entropy text due to its deterministic nature, so we use news articles to generate higher-entropy sequences. We use Llama-2-7B for generating both watermarked and benign text [16].

**Watermark Configuration** The baseline watermark for our evaluation is the KGW-SelfHash watermark, a variant of the soft red-list watermark proposed by Kirchenbauer et al. [1]. We apply our variable-context mechanism to the baseline watermark, setting the minimum context width to 1 and the maximum context width to 5. We set detection threshold for watermarked text to  $z = 4$ . A Z score output greater than 4 from the detector API means that the watermark was detected. We set the green-list size  $\gamma$  to 0.25, and the watermark strength  $\delta$  to 4, in accordance with the parameters used in [2].

**Attacker Configuration** The attacker is configured in the *B1D0* setting, meaning that the attacker has access to both watermarked and benign outputs, but does not have access to the watermark detector API. The attacker makes 12,600 queries to the target model. Each watermarked output has length  $T \leq 200$  tokens. The attacker assumes a context width of  $k = 3$  (the average context width given a minimum of 1 and maximum of 5).

The attacker aims to spoof the watermark, using Gemma-2B-instruct to generate spoofed text [17]. We run the spoofing attack four times, attempting to spoof 100 samples on each run for a total of 400 spoofing attempts.

### 4.2 Experimental Results

#### 4.2.1 Attack Evaluation

Table 1 and Figure 1 display the results of the attack on the baseline method (labeled as KGW) and our method (Ours). The average attack success rate (ASR) is 68.43% against our method and 76.77% against the baseline. The distribution of Z scores in Figure 1 shows that the attacker’s attempts to spoof our watermark (in blue) are slightly less successful than attempts on the baseline method (in orange). However, the distributions largely overlap and lie mostly to the right of the attack threshold of 4, indicating that the attack was successful on both watermarks.

Table 2 shows how our mitigation affects the spoofed text’s quality, measured by perplexity. Our mitigation increases the variance in the quality of the spoofed text, with a standard deviation of 9.8 compared to 3.2 for the baseline.

|                | KGW   | Ours         |
|----------------|-------|--------------|
| ASR (Mean, %)  | 76.77 | <b>68.43</b> |
| ASR (Best, %)  | 81.82 | <b>75.76</b> |
| ASR (Worst, %) | 74.75 | <b>62.63</b> |

Table 1: Average, Best, and Worst ASR (over four runs of the attack)

|      | Mean         | Std          | Min         | 25%         | 50%         | 75%          | Max           |
|------|--------------|--------------|-------------|-------------|-------------|--------------|---------------|
| KGW  | 9.604        | 3.231        | <b>3.71</b> | <b>7.25</b> | <b>9.17</b> | 11.53        | 18.14         |
| Ours | <b>10.91</b> | <b>9.835</b> | 3.17        | 6.81        | 8.57        | <b>11.86</b> | <b>107.31</b> |

Table 2: Perplexity of Spoofed Text

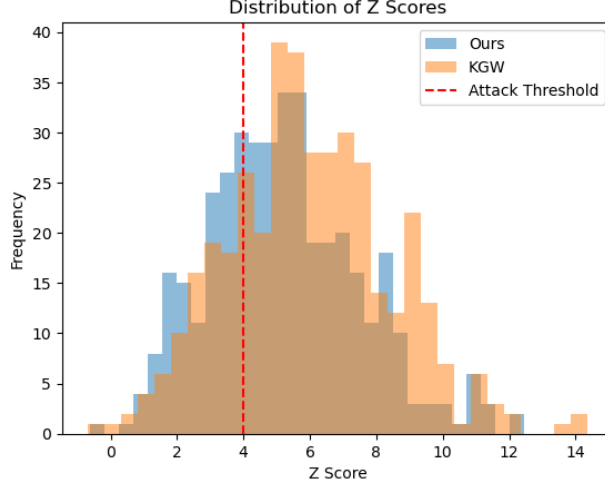


Figure 1: Distribution of attack Z scores for the baseline method and our mitigation.

#### 4.2.2 Ablation Study: Text Quality

We compare the quality of watermarked text with our mitigation and with the baseline watermark. GPT-2 Large is used for evaluating both texts [18]. As shown in Figure 2, utilizing a variable  $k$  does not significantly impact the quality of watermarked text compared to the baseline.

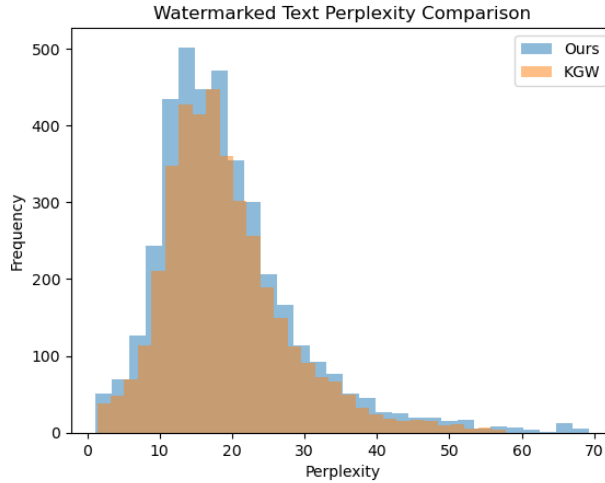


Figure 2: Distribution of perplexities for text watermarked with our mitigation and the baseline.

## 5 Discussion

**Takeaways** To answer research question Q1, we examine the success rate of the evaluated stealing attack. Applying a pseudorandom context width to the watermark *did* decrease the effectiveness of the attack, albeit by a small margin. The attack was still successful over 68% of the time, so our method is not robust against it. One reason the mitigation might not have worked as well as was hypothesized was that the range of context widths utilized was small, from 1 to 5. As discussed in Section 3, these bounds are used in order to increase the method’s robustness to attack amplification. The small range of  $k$  means that the attacker, who assumes  $k = 3$ , is still able to guess  $k$  correctly 20% of the time. Additionally, the results indicate that the attacker’s usage of no-context scores as part of their learning algorithm provide a significant benefit, and make the algorithm robust to cases where the context-based scores are based off of an incorrect context width.

An interesting side effect of the variable context width mechanism was the degradation in the quality of attacker-generated text. From Table 2, we see that the standard deviation in the perplexity of spoofed text is over three times that against our method compared to the baseline. While more extended evaluations are needed to confirm this phenomenon, it is possible that the variable context-width is able to disturb the attacker’s green and red list estimates enough for the spoofed text to be generated with lower quality.

To answer Q2, we examine the quality of the watermarked text. We find that utilizing a variable context width does not degrade it. Notably, the perplexity of generated text does not change when adding the variable context width (Figure 2). Appendix A provides some examples of watermarked text with our method and with the baseline watermark, along with results from the detector API. We are able to detect watermarked text generated using a variable context width with high confidence, even with texts of around 50 tokens in length. This shows that just 2-3 sentences of text are enough to detect the watermark.

**Challenges** The primary challenge of this project was a combination of limited computational resources and time to implement and test the proposed mitigation. Evaluating the stealing attack on the watermark required generating over 24,000 samples of watermarked text, with each sample needing to be long enough (around 200 tokens) so as not to weaken the attack too much. Just this generation of watermarked text had an estimated runtime of around 200 hours. Due to GPU usage limits on our research cluster, this computation had to be split into jobs of 8 hours at a time, with long wait times, sometimes over 48 hours, in between jobs. Actually running the attack was faster, at around 2 hours per run for a total of 8 hours.

All in all, these difficulties meant that there was unfortunately not enough time to fully test and evaluate our mitigation to its fullest extent. For instance, we would have liked to try configuring the watermark with a wider range of minimum and maximum context widths, as well as see the effects of varying other parameters such as the green list size  $\gamma$  and watermark strength  $\delta$ .

**Next Steps** First, this study is limited in the sense that the proposed mitigation is only evaluated using one watermark scheme and against one watermark stealing attack. Future work should aim to evaluate the mitigation on multiple logits-based watermarks and with a range of attacks, including stealing attacks and other generative or post-hoc attacks.

Next, several questions remain unanswered from this work. The proposed mitigation, evaluated with these models and parameters, did not fully mitigate the watermark stealing attack which had a strong but reasonably realistic threat model. We encourage future work to try to find improvements to the method proposed here, in order to improve its efficacy. For instance, one idea is to use a wider range of  $k$ s for the algorithm than evaluated in this work. Hopefully, introducing yet more randomness would make it more difficult for the attacker, although it is not clear if this is a scalable strategy. Another possible improvement is to use a fixed value of  $k$ , but instead of just using the  $k$  preceding tokens in the context, pseudorandomly select a  $k$ -token n-gram from a wider context window, which can be much longer than the single-digit contexts typically seen in logits-based watermarking schemes.

Ultimately, the exploration of variable context widths in LLM watermarking highlights both the challenges and opportunities in enhancing robustness against attacks. While the proposed mitigation offers a promising direction, it underscores the need for further research into scalable strategies that balance security and computational efficiency. By addressing these gaps, future advancements can strengthen watermarking schemes, ensuring their resilience in increasingly adversarial environments.

## References

- [1] John Kirchenbauer et al. A watermark for large language models, 2023.
- [2] Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models, 2024.
- [3] Tom B. Brown et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.
- [4] Cade Metz. Openai’s gpt-4 exhibits ‘human-level performance’ on professional benchmarks. *Ars Technica*, 2023.
- [5] Dorsaf Sallami, Yuan-Chen Chang, and Esma Aïmeur. From deception to detection: The dual roles of large language models in fake news, 2024.
- [6] Adav Noti. How artificial intelligence influences elections and what we can do about it, 2024.
- [7] Karen Hao. Deepfakes are becoming the hot new tool for election disinformation. *MIT Technology Review*, 2023.
- [8] David Feldman. How ai is changing the landscape of academic integrity. *The Chronicle of Higher Education*, 2023.

- [9] Rishi Bommasani et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.
- [10] Abeba Birhane et al. The values encoded in machine learning research. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 173–184, 2022.
- [11] Scott Aaronson. Watermarking of large language models. Talk at the Simons Institute for the Theory of Computing, August 2023.
- [12] Junchao Wu et al. A survey on llm-generated text detection: Necessity, methods, and future directions, 2023.
- [13] Sumanth Dathathri et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, October 2024.
- [14] Xuandong Zhao et al. Provable robust watermarking for ai-generated text, 2023.
- [15] Colin Raffel et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.
- [16] Hugo Touvron et al. Llama: Open and efficient foundation language models, 2023.
- [17] Thomas Mesnard et al. Gemma: Open models based on gemini research and technology, 2024.
- [18] Alec Radford et al. Language models are unsupervised multitask learners, 2019.

## A Watermarked Text Examples

Examples of watermarked text are provided below. The prompt is shown in gray, and the LLM output in green. The results from the watermark detector API are in the yellow box. Watermarked text is generated with OPT-1.3B. The watermark is detected with 100% confidence and easily surpasses the detection threshold of  $z = 4$ , even with a relatively small number (53) of tokens.

Prompt:

The diamondback terrapin or simply terrapin (*Malaclemys terrapin*) is a species of turtle native to the brackish coastal tidal marshes of the Northeastern and southern United States, and in Bermuda.[6] It belongs to the monotypic genus *Malaclemys*. It has one of the largest ranges of all turtles in North America, stretching as far south as the Florida Keys and as far north as Cape Cod.[7] The name 'terrapin' is derived from the Algonquian word *torope*. [8] It applies to *Malaclemys terrapin* in both British English and American English. The species is

Watermarked Output:

known under a name used only in American literature – the 'diamondback terrapin'. There are five recognized subspecies, of varying degree of variation. The most numerous are *Malaclemys gossypi*, and the other two are *Malaclemys fuscus* (G. n. *fuscus*) and *Malaclemys fuscus* (G. n. *fuscus longirostris*). [8]

| Metric                     | Value             |
|----------------------------|-------------------|
| Tokens Counted (T)         | 88                |
| # Tokens in Greenlist      | 53                |
| Fraction of T in Greenlist | 60.2%             |
| z-score                    | 7.63              |
| p-value                    | $1.16 * 10^{-14}$ |
| prediction                 | watermarked       |
| confidence                 | 100.00%           |

Prompt:

The speculation is that Apple is designing a new iWatch that connects to iPhones, relays messages, and displays status alerts. But would the company that reinvented the computer, the phone, and the way we consume media be aiming so low? Ever since

Watermarked Output:

the iPod was introduced way back in 2001, the only way the iPod was ever really “revolutionary” was thanks to the iPod Touch. While there were plenty of other options out in the prior iPod generation (like the iPod nano), iPod Touch and iPod nano combined Apple’s strengths (and iPod nano could be upgraded through the iTunes Store) with Android/Apple’s strengths (and the ability to make phone calls through the iTunes App Store).

| Metric                     | Value             |
|----------------------------|-------------------|
| Tokens Counted (T)         | 91                |
| # Tokens in Greenlist      | 63                |
| Fraction of T in Greenlist | 69.2%             |
| z-score                    | 9.74              |
| p-value                    | $9.77 * 10^{-23}$ |
| prediction                 | watermarked       |
| confidence                 | 100.00%           |

Prompt:

Biologists have documented a previously unknown species of mammals that emit a soft, bioluminescent glow. Found deep in the Amazon rainforest, this discovery is sparking new questions about evolutionary biology and adaptation. Researchers are studying these creatures to understand their role in the ecosystem and the genetic mechanisms behind their glow. Some of the

## Watermarked Output:

ichnodontid ichnodonts are believed to resemble the ichnodont marsupials. This finding suggests the ichnodontid ichnodonts are closely related to marsupials. Photo by Richard Chappat.

| Metric                     | Value             |
|----------------------------|-------------------|
| Tokens Counted (T)         | 53                |
| # Tokens in Greenlist      | 33                |
| Fraction of T in Greenlist | 62.3%             |
| z-score                    | 6.27              |
| p-value                    | $1.86 * 10^{-10}$ |
| prediction                 | watermarked       |
| confidence                 | 100.00%           |

## Prompt:

The printing press is a mechanical device invented in the mid-15th century by Johannes Gutenberg. It revolutionized the production of books by introducing movable type, allowing for the mass production of written materials for the first time in Europe. The invention played a pivotal role in the spread of knowledge during the Renaissance and is often credited with accelerating the Reformation and the Scientific Revolution. Gutenberg's press used a screw mechanism inspired by wine presses and

## Watermarked Output:

was the first 'modern' printer. It produced a series of paper copies of the Bible, the Gutenberg Bible. These copies have been used as the starting point for the development of the printed book. It is estimated that over 600 copies of the Gutenberg Bible are currently held in libraries and archives.

| Metric                     | Value             |
|----------------------------|-------------------|
| Tokens Counted (T)         | 61                |
| # Tokens in Greenlist      | 42                |
| Fraction of T in Greenlist | 68.9%             |
| z-score                    | 7.91              |
| p-value                    | $1.29 * 10^{-15}$ |
| prediction                 | watermarked       |
| confidence                 | 100.00%           |