Black-Box Detection of Language Model Watermarks

Anonymous Authors[1]

Abstract

Watermarking has emerged as a promising way to detect LLM-generated text. To apply a watermark an LLM provider, given a secret key, augments generations with a signal that is later detectable by any party with the same key. Recent work has proposed three main families of watermarking schemes, two of which focus on the property of preserving the LLM distribution which supposedly makes it harder for malicious actors to bypass the watermark. Yet, despite much discourse around detectability, no prior work has investigated if any of these scheme families are detectable in a realistic black-box setting. We tackle this for the first time, developing rigorous statistical tests to detect the presence of all three most popular watermarking scheme families using only a limited number of black-box queries. We experimentally confirm the effectiveness of our methods on a range of schemes and a diverse set of open-source models. Our findings indicate that current watermarking schemes are more detectable than previously believed. We further apply our methods to test for watermark presence behind the most popular public APIs: GPT4, CLAUDE 3, GEMINI 1.0 PRO, finding no strong evidence of a watermark at this point in time.

1. Introduction

With the rapid increase in large language model (LLM) capabilities and their widespread adoption, researchers and regulators alike are raising concerns about their potential misuse for generating harmful content (Bommasani et al., 2021; Council of the European Union, 2024). To tackle this issue, the idea of watermarking, a process of embedding a signal invisible to humans into generated texts, is gaining significant traction.

Language model watermarking More formally, in LLM watermarking (Hu et al., 2024; Kirchenbauer et al., 2023; 2024; Kuditipudi et al., 2024; Sadasivan et al., 2023; Wang et al., 2024; Wu et al., 2023; Yoo et al., 024s) we consider the setting of a *model provider* that offers black-box access to their proprietary model LM while ensuring that each generation y in response to a prompt q can be reliably attributed to the model. To enable this, the provider modifies the generations using a secret watermark key ξ, which a corresponding watermark detector can later use to detect whether a given text was generated by LM.

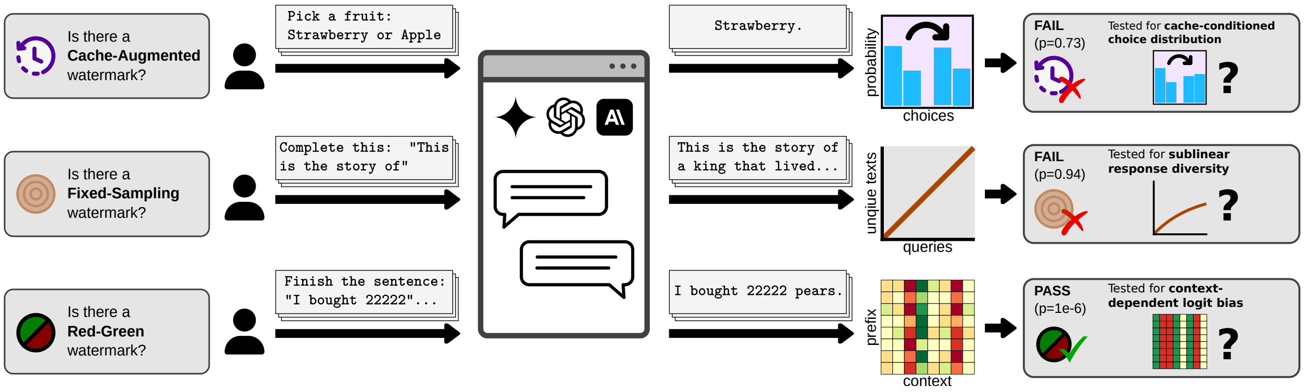
The prominent family of *Red-Green* watermarks (Kirchenbauer et al., 2023; 2024) achieves this by, at each step of generation, selecting a context-dependent pseudorandom set of logits to be boosted, modifying the model's next-token distribution. In contrast to these, the recently proposed families of *Fixed-Sampling* (Kuditipudi et al., 2024) and *Cache-Augmented* (Hu et al., 2024; Wu et al., 2023) schemes have focused on developing watermarks that aim to preserve the output distribution of the LLM, achieving such guarantees in ideal theoretical settings. This goal can be motivated by two main benefits: (i) it is a tractable proxy for preservation of capabilities of the original LLM, and (ii) intuitively, the deployment of the watermark should become inherently harder to detect for malicious actors that would otherwise be able to avoid or attack the watermark (Jovanović et al., 2024; Pang et al., 2024; Sadasivan et al., 2023). Yet, despite watermark detection being a key concern in current discourse, there have so far been no investigations into the *practical detectability* of any of the three watermark families.

This work: practical black-box watermark detection In this work, for the first time, we investigate the question of watermark detection in practical real-world settings. Faithful to how LLMs are exposed in current public deployments, we assume a minimal setting in which the adversary cannot control any input parameters (e.g., temperature, sampling strategy) apart from the prompt q, and does not receive any additional information (e.g., log-probabilities) apart from the instruction-tuned model's textual response y. In this setting, the goal of the adversary is to identify if the model is watermarked and determine which scheme was used.

To this end, we propose rigorous statistical tests for the black-box detection of seven scheme variants of the three

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

Preliminary work. Under review by the ICML 2024 Workshop on Foundation Models in the Wild. Do not distribute.


Is there a Cache-Augmented watermark?
Is there a Fixed-Sampling watermark?
Is there a Red-Green watermark?
probability
choices
unique texts
queries
prefix
context
FAIL (p=0.73)
Tested for cache-conditioned choice distribution
?
FAIL (p=0.94)
Tested for sublinear response diversity
?
PASS (p=1e-6)
Tested for context-dependent logit bias
?

Pick a fruit: Strawberry or Apple

Strawberry.

Complete this: "This is the story of"

This is the story of a king that lived...

Finish the sentence: "I bought 22222"...

I bought 22222 pears.

Figure 1: Given black-box textual access to a language model, a client can query the model and run statistical tests to rigorously test for presence of a watermark. In this example, both the test for *Cache-Augmented* watermarks and the test for *Fixed-Sampling* watermarks fail, while the test for *Red-Green* watermarks successfully detects the watermark.

most prominent watermark families. Our tests, illustrated in Figure 1, are based on fundamental properties of the three respective scheme families: Red-Green, Fixed-Sampling, and Cache-Augmented watermarks. In our extensive experimental evaluation, we find that in practice, both distribution-modifying as well as distribution-preserving schemes are *easily detectable* by our tests, even in the most restricted black-box setting. Highlighting their practicality, we further test several real-world LLM deployments (GPT4, CLAUDE 3, GEMINI 1.0 PRO) and detect no watermarks, suggesting that watermark deployment at this scale is a challenge.

Implications Our findings question whether non-detectability should be a key focus in the development of LLM watermarks. Guided by our results, we suspect that avoiding practical detectability may be an inherently hard problem given the fundamental nature of watermarks. In light of this, we advocate for a more diverse and practically oriented development and evaluation of LLM watermarks, allowing for broader consideration of other relevant properties such as attack robustness, text quality, and efficiency. Such a shift in focus could lead to watermarking schemes that better safeguard against misuse of LLMs, while also being more practical and deployable in real-world settings.

Main contributions Our key contributions are threefold:

- We present the first statistically rigorous tests for practical black-box detection of LLM watermarks across the three prominent scheme families: Red-Green, Fixed-Sampling, and Cache-Augmented (§2).

- We confirm the effectiveness of all our tests with an experimental evaluation across seven schemes and five open-source models, and verify the applicability of our tests in real-world settings (§3.1).

- We further highlight the practicality of our tests by executing them on several real-world black-box LLM deployments—GPT, CLAUDE, and GEMINI (§3.2).

2. Detecting and Identifying Watermarks

Our threat model is as follows: a model provider offers black-box access to their model LM. The *adversary* queries the model, observes its responses and infers with high certainty whether the model is watermarked and to which family the watermark belongs, as illustrated in Figure 1. The full technical descriptions of the three tests are deferred to App. A–C for conciseness.

Red-Green watermarks The Red-Green watermarking family, introduced by Kirchenbauer et al. (2023), operates by shifting a fraction $\gamma \in [0, 1]$, of the model's logits by a positive constant δ at each generation step. The shifted logits, named the *green vocabulary*, are chosen based on the last $h > 0$ previous tokens, which we refer to as the *context*.

First, we limit the model output to a small subset of tokens Σ, ensuring the model distribution across different contexts is roughly uniform, which simplifies the estimation process. We then vary both the context and a *prefix* that does not change the green vocabulary. In practice, Σ is a predefined set of choices provided to the model, the context is made of digits, and the prefix of preceding verbs. We estimate the probabilities of each token within Σ for each combination of context and prefix by querying the model multiple times.

To test the null hypothesis that the model is not Red-Green watermarked, we count tokens that are systematically more likely for a given context, independent of the prefix. We then use a *permutation test*, comparing the observed counts to the counts obtained by randomly permuting the context and prefix. For a watermarked model, the permuted setting will not exhibit the same systematic bias as the original setting, while an unwatermarked model will. Hence, we can reject the null hypothesis if the observed counts are significantly different from the permuted counts. This detection test is illustrated in Figure 1 (bottom). In App. A, we detail the statistical modeling underlying the permutation test.

Fixed-Sampling watermarks Unlike Red-Green watermarks, the recently proposed Fixed-Sampling watermarks by Kuditipudi et al. (2024) do not modify the logit vectors during generation, so estimating the probabilities of model outputs as above is not informative. Instead, the sampling is fully determined by the rotation of the watermark key, making the natural vector to exploit when detecting this watermark its *lack of diversity*. This means that the model can only produce a finite number of unique outputs, specifically the key length n_{key}, for a given prompt.

Given a prompt for which an unwatermarked model is expected to produce highly diverse outputs, we can use this observation to reliably distinguish between the two. By querying the LM multiple times with such a prompt, we build the *rarefaction curve*, which is the expected number of unique outputs as a function of the number of queries. The rarefaction curve of an unwatermarked LM grows linearly with the number of queries, while the watermarked LM curve asymptotes at n_{key}, as illustrated in Figure 1 (middle). We use the Mann-Whitney U test to determine if the observed rarefaction curve is linear.

Cache-Augmented watermarks Like the Red-Green family, the Cache-Augmented family, introduced by Hu et al. (2024), uses a *context* to seed the underlying watermark mechanism. However, once the context has been seen once, it is cached, and the watermark is not applied again for the same context. The underlying watermark mechanism either deterministically samples the model distribution (δ-REWEIGHT) or reweights the distribution (DIPMARK/γ-REWEIGHT) by Hu et al. (2024); Wu et al. (2023). In any case, the watermarked distribution conditioned on a context differs from the original one.

This means that the model's distribution is different the first time a context is seen compared to subsequent times. By leveraging the necessity of periodically resetting the cache, we can probe the model's distribution both without and with the watermark. To test for this change, we first probe the model until we find a prompt where the model chooses among two tokens with almost equal probability. This lowers the number of queries needed to detect a distribution change. Then we query the model with the same prompt, making sure to reset the cache in between queries. Our test then compares the two distributions using a Fisher's exact test, rejecting the null hypothesis if the two distributions are different, i.e., if the model is watermarked. The test is illustrated in Figure 1 (top).

3. Experimental Evaluation

In this section, we apply the tests introduced in §2 to a wide range of models and watermarking schemes, and confirm their effectiveness in detecting watermarks. In §3.1 we show that the adversary can reliably detect the watermarking scheme used (if any) at a low cost, across a wide range of practical settings. We then demonstrate in §3.2 that our tests can be directly applied to real-world LLM deployments, revealing no significant evidence of any watermarking schemes at this point in time.

3.1. Main Experiments: Detecting Watermarking Schemes

Experimental setup We run all our tests on five different models (MISTRAL-7B, LLAMA2-7B, -13B, and -70B, and LLAMA3-8B), in different scenarios. We here present results of a subset of those, and defer the rest to App. D. In each scenario, each model is either unwatermarked (where we vary the temperature) or watermarked with a scheme from the three families (where we vary the particular scheme and its parameters). If our tests are reliable, we expect to see low p-values only when the model is watermarked exactly with the scheme family that we are testing for.

For Red-Green tests, we set $N = 10, M = 9, r = 1.96$, a different Σ per model based on the first $Q1$ samples, use 100 samples to estimate the probabilities, and use 10000 permutations in the test. For Fixed-Sampling tests, we use $n = 1000$ queries and set $t = 50$. For Cache-Augmented tests, we use $Q1 = Q2 = 75$ and assume the cache is cleared between queries in the second phase.

Results: reliable watermark detection Our main results are shown in Table 1, where we report the median p-values for each (model, test, scenario) tuple across 100 repetitions of each experiment. Across all experiments, all three tests reject the null hypothesis (*the specific watermark is not present*) at a 95% confidence level only when the scheme from the target family is indeed applied to the model. This confirms that our tests are reliable in detecting watermarks, and robust with respect to the model and the specific parameters of the scheme, emphasizing our tests' generalization to all schemes that are based on the same foundational principles. In particular, our Red-Green tests are robust to the seeding scheme, the logit bias δ, and the green token fraction γ; our Fixed-Sampling tests maintain high confidence even for unusually high values of n_{key}; finally, our Cache-Augmented tests are robust to all details of the underlying scheme. Our results also provide evidence that unrelated model modifications do not cause false positives.

Our tests do not incur significant costs for the adversary, making them easily applicable in practice. We estimate the average cost of the tests to be below \$20 for Red-Green, \$1

Table 1: Main results of our watermark detection tests across different models and watermarking schemes. We report median p-values across 100 repetitions of the experiment, and for Red-Green schemes additionally over 5 watermarking keys. p-values below 0.05 (test passing) are highlighted in bold. δR and γR denote δ-reweight and γ-reweight schemes.

				Red-Green				Fixed-Sampling		Cache-Augmented		
		Unwatermarked		LeftHash		SelfHash		ITS/EXP		DiPmark/γR		δR
Model	Test	$T =$ 1.0	$T =$ 0.7	$\delta, \gamma =$ $2, 0.25$	$\delta, \gamma =$ $4, 0.5$	$\delta, \gamma =$ $2, 0.5$	$\delta, \gamma =$ $4, 0.25$	$n_{key} =$ 256	$n_{key} =$ 2048	$\alpha =$ 0.3	$\alpha =$ 0.5	
Mistral 7B	R-G	1.000	1.000	**0.000**	**0.000**	**0.000**	**0.000**	1.000	1.000	1.000	1.000	1.000
	Fixed	0.938	0.938	0.938	0.938	0.938	0.938	**1.3e-125**	**1.1e-9**	0.938	0.938	0.938
	Cache	0.570	0.667	0.607	0.608	1.00	0.742	0.638	0.687	**2.4e-4**	**2.1e-3**	**5.6e-27**
Llama2 13B	R-G	0.149	0.663	**0.000**	**0.000**	**0.000**	**0.000**	0.121	0.128	0.149	0.149	0.149
	Fixed	0.968	0.868	0.938	0.938	0.938	0.938	**5.5e-124**	**7.5e-8**	0.938	0.938	0.968
	Cache	0.708	0.573	0.511	0.807	0.619	0.710	0.518	0.692	**1.8e-2**	**5.3e-3**	**6.7e-32**
Llama2 70B	R-G	1.000	1.000	**0.000**	**0.020**	**0.000**	**0.000**	1.000	1.000	1.000	1.000	1.000
	Fixed	0.938	0.526	0.966	0.965	0.984	0.975	**4.2e-125**	**1.7e-8**	0.938	0.966	0.938
	Cache	0.596	0.620	0.657	0.639	0.651	0.608	0.463	0.818	**1.5e-3**	**4.4e-3**	**5.8e-28**

Table 2: The results of our watermark detection tests on popular black-box LLM deployments.

	GPT4	Claude 3	Gemini 1.0 Pro
R-G	0.998	0.638	0.683
Fixed	0.938	0.844	0.938
Cache	0.51	0.135	0.478

for Fixed-Sampling, and $0.1 for Cache-Augmented tests, assuming latest OpenAI GPT4o pricing.

3.2. Detecting watermarks in deployed models

Finally, we demonstrate the applicability of the statistical tests introduced in §2, by applying them to popular black-box LLM deployments: GPT4, Claude 3, and Gemini 1.0 Pro. We use the same experimental setup as in §3.1, and use the API access for efficiency reasons—we do not rely on any additional capabilities, and our tests could be as easily run in the web interface. For the Cache-Augmented tests, we assume a global cache, that clears after 1000 seconds.

Results Our results in Table 2 show that the null hypothesis is not rejected for any of the models and any of the watermark tests. This suggests that none of the deployed models tested are watermarked at this point in time. The demonstrated applicability of our tests makes it simple to monitor these deployments and detect any watermarking schemes that may be introduced in the future.

4. Conclusion, Impact and Limitations

In this paper, we have focused on the problem of detecting watermarks in large language models (LLMs) given only black-box access. We developed rigorous statistical tests for the three most prominent scheme families, and validated their effectiveness. Our results imply that protecting from detectability should not be the key focus of LLM watermarks, and that other properties such as robustness to attacks, text quality, and efficiency should be prioritized.

Broader Impact While our work primarily enables malicious parties to more easily circumvent attempts at tracing LLM-generated text, we believe the benefits of our work outweigh those risks, as our conclusions help model providers calibrate their expectations in terms of watermark detectability, and avoid a false sense of safety.

Limitations One limitation of our tests is that they are restricted to the three scheme families discussed in §2. These are indeed the most prominent in the literature, and as our tests are based on fundamental properties of these scheme families, they should generalize to more variants and combinations of the underlying ideas. However, it is possible that a model provider deploys a scheme based on an entirely novel idea, which our tests would not be able to detect. Further, our p-values are based on several model assumptions, such as symmetric error terms and perfect sampling. While we validate that these assumptions are sufficiently met on several open-source models, we cannot guarantee that all models adhere to them. Finally, another conceptual limitation of our Red-Green test is that it does not take into account the possibility that the red-green vocabulary split is the same (on the observed domain) for all contexts. The probability of this event decreases exponentially with the number of different contexts, and is thus unlikely to affect the test's performance in practice.

References

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R. B., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N. S., Chen, A. S., Creel, K., Davis, J. Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N. D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D. E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P. W., Krass, M. S., Krishna, R., Kuditipudi, R., and et al. (2021). On the opportunities and risks of foundation models. *CoRR*.

Carlini, N., Paleka, D., Dvijotham, K. D., Steinke, T., Hayase, J., Cooper, A. F., Lee, K., Jagielski, M., Nasr, M., Conmy, A., Wallace, E., Rolnick, D., and Tramèr, F. (2024). Stealing part of a production language model. *CoRR*.

Council of the European Union (2024). Proposal for a regulation of the european parliament and of the council laying down harmonised rules on artificial intelligence (artificial intelligence act) and amending certain union legislative acts - analysis of the final compromise text with a view to agreement.

Gu, C., Li, X. L., Liang, P., and Hashimoto, T. (2024). On the learnability of watermarks for language models. *ICLR*.

Hou, A. B., Zhang, J., He, T., Wang, Y., Chuang, Y., Wang, H., Shen, L., Durme, B. V., Khashabi, D., and Tsvetkov, Y. (2023). Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv*.

Hu, Z., Chen, L., Wu, X., Wu, Y., Zhang, H., and Huang, H. (2024). Unbiased watermark for large language models. *ICLR*.

Jovanović, N., Staab, R., and Vechev, M. (2024). Watermark stealing in large language models. *arXiv*.

Kirchenbauer, J., Geiping, J., Wen, Y., Katz, J., Miers, I., and Goldstein, T. (2023). A watermark for large language models. In *ICML*.

Kirchenbauer, J., Geiping, J., Wen, Y., Shu, M., Saifullah, K., Kong, K., Fernando, K., Saha, A., Goldblum, M., and Goldstein, T. (2024). On the reliability of watermarks for large language models. *ICLR*.

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

Li, Y. and Fung, P. (2013). Improved mixed language speech recognition using asymmetric acoustic model and language model with code-switch inversion constraints. In *ICASSP*.

Liu, A., Pan, L., Hu, X., Li, S., Wen, L., King, I., and Yu, P. S. (2024a). A private watermark for large language models. *ICLR*.

Liu, A., Pan, L., Hu, X., Meng, S., and Wen, L. (2024b). A semantic invariant robust watermark for large language models. *ICLR*.

Naseh, A., Krishna, K., Iyyer, M., and Houmansadr, A. (2023). On the risks of stealing the decoding algorithms of language models. *CoRR*.

Pang, Q., Hu, S., Zheng, W., and Smith, V. (2024). Attacking LLM watermarks by exploiting their strengths. *arXiv*.

Piet, J., Sitawarin, C., Fang, V., Mu, N., and Wagner, D. A. (2023). Mark my words: Analyzing and evaluating language model watermarks. *arXiv*.

Rando, J. and Tramèr, F. (2024). The worst (but only) claude 3 tokenizer. https://github.com/javirandor/anthropic-tokenizer.

Ren, J., Xu, H., Liu, Y., Cui, Y., Wang, S., Yin, D., and Tang, J. (2023). A robust semantics-based watermark for large language model against paraphrasing. *arXiv*.

Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., and Feizi, S. (2023). Can ai-generated text be reliably detected? *arXiv*.

Tang, L., Uberti, G., and Shlomi, T. (2023). Baselines for identifying watermarked large language models. *CoRR*.

Wang, L., Yang, W., Chen, D., Zhou, H., Lin, Y., Meng, F., Zhou, J., and Sun, X. (2024). Towards codable text watermarking for large language models. *ICLR*.

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

Yoo, K., Ahn, W., and Kwak, N. (2024s). Advancing beyond identification: Multi-bit watermark for language models. *NAACL*.

A. Detecting Red-Green Watermarks

Here we provide a more detailed introduction to the Red-Green watermarking scheme, and describe in details the statistical test we developed to detect it.

Background Assume a watermark key $\xi \in \mathbb{N}$, a pseudorandom function (PRF) f, and a hashing function $H\colon \mathbb{N} \to \mathbb{N}$. At each generation step t, a Red-Green watermark modifies the logits $l_{1:|V|}$ of tokens from the vocabulary V to promote a subset of tokens (*green tokens*) before applying standard sampling. We consider two popular variants, LeftHash (Kirchenbauer et al., 2023) and SelfHash (Kirchenbauer et al., 2024), both parametrized by $\delta, \gamma \in \mathbb{R}^+$, and using $h = 1$ and $h = 3$ previous tokens as context, respectively. LeftHash seeds f with $H(y_{t-1}) \cdot \xi$, and uses it to split V into $\gamma|V|$ green tokens and remaining *red* tokens. For each green i, it then increases l_i by δ. SelfHash differs by seeding f with $\min(H(y_{t-3}), \ldots, H(y_t)) \cdot H(y_t) \cdot \xi$, effectively using a context size of 4 by including the token y_t yet to be generated. For both schemes, the watermark detector is based on observing a significant number of green tokens in the input text of length m, above the expectation of γm. Other schemes from this family include, among else, varying the aggregation function or the context size.

Modeling Red-Green watermarks Assume a setting where the model chooses a token from some restricted set $\Sigma \subset V$, following some *context* t_2 (longer than the watermark context h), which is preceded by a *prefix* t_1. We discuss how to construct such a setting shortly. To model the watermark, we assume the following distribution for $p_{t_1,t_2}(x)$, the probability the model assigns to some $x \in \Sigma$:

$$p_{t_1,t_2}(x) = \frac{\exp\big((x^0 + \delta_{t_2}(x) + \varepsilon_{t_1,t_2}(x))/T\big)}{\sum_{w\in\Sigma} \exp\left((w^0 + \delta_{t_2}(w) + \varepsilon_{t_1,t_2}(w))/T\right)}. \tag{1}$$

Here, we assume the existence of the *true logit* x^0, modeling the model bias towards x. The true logit is shifted by $\delta_{t_2}(x)$, a δ-Bernoulli random variable, where $\delta \in \mathbb{R}^+$ is the watermark parameter introduced above. Finally, $\varepsilon_{t_1,t_2}(x)$ for different t_1, t_2 are iid symmetric error terms with mean 0 and variance σ^2. Applying the *logit* function $p \to \log(p/(1-p))$ to Equation (1), we obtain:

$$l_{t_1,t_2}(x) = \frac{x^0}{T} + \frac{\delta_{t_2}(x)}{T} + \frac{\varepsilon_{t_1,t_2}(x)}{T} - \log\left(\sum_{w\in\Sigma\setminus\{x\}} \exp\left(\frac{w^0}{T} + \frac{\delta_{t_2}(j)}{T} + \frac{\varepsilon_{t_1,t_2}}{T}\right)\right). \tag{2}$$

Approximating the log-sum-exp with a maximum, and WLOG setting $w^0 = 0$ for w which is the maximum in the log-sum-exp (as logits are shift-invariant), the above simplifies to

$$l_{t_1,t_2}(x) = x^0/T + \delta'_{t_2}(x) + \varepsilon'_{t_1,t_2}(x), \tag{3}$$

where $\varepsilon'_{t_1,t_2}(x)$ absorbs the previous error terms. The resulting $\delta'_{t_2}(x)$ is a random variable that is 0 for unwatermarked models, and has 3 possible values for watermarked models: δ/T (where δ is the watermark parameter), if x is the only token from Σ that is in the green partition of the vocabulary, $-\delta/T$, if x is in the red partition and some other tokens from Σ are in the green partition, and 0 otherwise. Our test is based on detecting cases of $\delta'_{t_2}(x) \neq 0$ and checking that their occurrence is indeed independent of t_1, distinguishing model variance from the watermark bias.

Querying strategy Recall that our goal is to steer a model into a setting (via an instruction, as we have no access to the completion model) where it makes a choice from some restricted set Σ. Importantly, we want to ensure enough variability in the model's choices to be able to observe the behavior specific to a Red-Green watermark and perform efficient estimation (in terms of query cost).

Assuming an upper bound H on the context size h of the watermark, we use the following prompt template, parametrized by t_1 (an arbitrary length string), d (a single token), and a word list Σ.

$$\texttt{f"Complete the sentence \textbackslash"\{}t1\texttt{\} }\underbrace{\texttt{\{}d\cdot H\texttt{\}}}_{t_2}\texttt{\textbackslash" using a random word from: [\{}\Sigma\texttt{\}]."}$$

Here, t_1 serves as the prefix, and $t_2 = d \cdot H$ as the context. For a watermarked model, changing t_2 is the only way to change the red-green vocabulary split, which can greatly affect the model's choice. For unwatermarked models, while we often see bias towards some choices from Σ, this bias will not strongly depend on t_2. This holds for all context sizes $\leq H$ and most aggregation functions, making our setting and the corresponding test directly applicable to different variants of Red-Green schemes.

In our instantiation (illustrated in Figure 1), we use N different t_1 of the form *"I bought"*, varying the verb, M different values of d from the set of digits, a word list Σ of four different plural fruit names, and an example that uses different words to not bias the model towards a specific choice. We empirically observe that this setup minimizes the chance that the model collapses to a single choice or fails to follow the instruction, which often occurred for similar settings based on repetition of words or numbers outside of natural context.

Estimating the logits To collect the data for the test, we query the model in two phases. First, we choose different values of Σ until we find one where the model does not always make the same choice, inducing Q_1 total queries. We set x to be the most commonly observed word from Σ. Next, for each (t_1, t_2) pair we query the model until we obtain K valid responses (filtering out failures to follow the instruction), inducing Q_2 additional queries.

We use these samples to estimate the model *logits* corresponding to x as $\hat{l}_{t_1,t_2}(x) = \log \frac{\hat{p}_{t_1,t_2}(x)}{1-\hat{p}_{t_1,t_2}(x)}$, where $\hat{p}_{t_1,t_2}(x)$ is the empirically estimated probability of x in the responses. The result of the adversary's querying procedure is a matrix $L_{N\times M}$ (visualized in Figure 1) of such logit estimates.

Testing watermark presence Finally, we describe the statistical test based on the logit estimates $L_{N\times M}$. We first estimate σ, the standard deviation of $\varepsilon'_{t_1,t_2}(x)$, as follows:

$$\hat{\sigma}^2 = \text{median}[\text{Var}_{t_2}(L)], \tag{4}$$

using the empirical median to improve robustness to unpredictable behavior caused by different t_1. Then, we calculate the following two binary functions, which flag cases where we believe the model's probability was affected by a watermark:

$$R_x(t_1,t_2) = \mathbb{1}\{\hat{l}_{t_1,t_2}(x) - \text{median}[L] < -r\hat{\sigma}\}, \tag{5}$$

and

$$G_x(t_1,t_2) = \mathbb{1}\{\hat{l}_{t_1,t_2}(x) - \text{median}[L] > r\hat{\sigma}\}, \tag{6}$$

with $r \in \mathbb{R}^+$ a parameter of the test. In practice, to account for model unpredictability, we use the empirical median conditioned on t_1 in Equations (5) and (6). For simplicity, let us denote $t_1 \in \{1,\ldots,N\}$ and $t_2 \in \{1,\ldots,M\}$. Let $cnt_x(t_2) = \max\left(\sum_{t_1=1}^{N} R_x(t_1,t_2), \sum_{t_1=1}^{N} G_x(t_1,t_2)\right)$ count the number of consistently flagged values for fixed t_2. We define the following test statistic:

$$S_x(L) = \max_{t_2\in[M]} cnt_x(t_2) - \min_{t_2\in[M]} cnt_x(t_2). \tag{7}$$

The null hypothesis of our test is $\forall t_2\colon\ \delta'_{t_2}(x) = 0$, i.e., the model is not watermarked. To obtain a p-value, we apply a Monte Carlo permutation test to S_x, checking if the flagged values are correlated with t_2 in a way that indicates a Red-Green watermark. Namely, we sample a set of permutations σ of the matrix L uniformly at random, and calculate a 99% confidence interval of $\Pr[S_x(\sigma(L)) \geq S_x(L)]$, whose upper bound we take as our p-value. When this value is small, we interpret that as evidence of a watermark. Because Equation (3) is permutation invariant when $\delta'_{t_2}(x) = 0$, this ensures that the test does not reject under the null. This completes our method for detection of Red-Green watermarks.

B. Detecting Fixed-Sampling Watermarks

As in App. A, we begin by introducing the background of the watermarking scheme. We then formally model the diversity of model outputs, discuss our querying strategy to ensure our assumptions are met, and describe the resulting statistical test.

Background For Fixed-Sampling watermarks, the secret watermark key sequence ξ of length n_{key} is cyclically shifted uniformly at random for each generation to obtain $\bar{\xi}$, and the entry $\bar{\xi}_t$ is used to sample from l. In the ITS variant, $\bar{\xi}_t$ is a pair $(u,\pi) \in [0,1] \times \Pi$, where Π defines the space of permutations of the vocabulary V. Given the probability distribution p over V, obtained by applying the softmax function to l, ITS samples the token with the smallest index in the permutation π

such that the CDF of p with respect to π is at least u. In the EXP variant, $\bar{\xi}_t$ is a value $u \in [0, 1]$, and we sample the token $\arg\min_{i \in V} -\log(u)/p_i$. The detection, for both variants, is based on testing the correlation between the text and ξ using a permutation test. As noted in §1, the key design goal of ITS and EXP is that, in expectation w.r.t. ξ, they do not distort the distribution of the responses. How close to this ideal is achieved in practical implementations is the question we aim to answer.

Modeling the diversity Let $U_n(q, t)$ denote a random variable that counts the number of unique outputs to a fixed prompt q in n queries, each of length exactly t in tokens. We introduce the *rarefaction curve* (visualized in Figure 1) as

$$R(n) = \mathbb{E}[U_n(q, t)]. \tag{8}$$

For suitable q that enables diversity and large enough t, the unwatermarked model exhibits $R(n) = n$. For a Fixed-Sampling watermark (both ITS and EXP variants), the watermark key segment used for sampling is determined by choosing a rotation of the key ξ uniformly at random. As choosing the same rotation for the same prompt and sampling settings will always yield the same output, the number of unique outputs is at most equal to the key size n_{key}. The probability that an output i was not produced is given by $(1 - 1/n_{key})^n$. By linearity of expectation, we have the rarefaction curve

$$R(n) = n_{\text{key}} \left(1 - (1 - \frac{1}{n_{\text{key}}})^n \right). \tag{9}$$

Querying strategy To estimate $R(n)$ of LM, we query it with a fixed prompt q, using rejection sampling to discard short responses until we obtain a set of N responses of length t tokens (inducing Q total queries). We then repeatedly sample n responses from this set to get a Monte-Carlo estimation of $R(n)$. There are two key considerations. First, we need to ensure that we are in a setting where an unwatermarked model would have $R(n) = n$. To do this, we use the prompt `"This is the story of"` that reliably causes diverse outputs, and set t high enough to minimize the chance of duplicates. In §3 we experimentally confirm that the number of unique outputs scales exponentially with t, and investigate the effect of small temperatures. Second, as larger n_{key} make $R(n)$ closer to linear, we must ensure that n is large enough for our test to be reliable. To do this, we can set an upper bound $\bar{n}_{key}$ on key size, and estimate the number of samples needed for a given power by simulating the test—our experiments show that our test succeeds even on values of $\bar{n}_{key}$ far above practical ones.

Testing watermark presence Finally, to test for presence of a Fixed-Sampling watermark, we use a Mann-Whitney U test to compare the rarefaction curve $R(n) = n$ with the observed rarefaction data obtained as above. If the test rejects the null hypothesis, we interpret this as evidence that the model is watermarked with a Fixed-Sampling scheme.

C. Details of Cache-Augmented Schemes

We provide the details of the three Cache-Augmented watermarking schemes considered in this work: δ-REWEIGHT (Hu et al., 2024), γ-REWEIGHT (Hu et al., 2024), and DIPMARK (Wu et al., 2023). We then describe in details the watermark detection tests for these schemes.

Cache-Augmented watermarking schemes All three variants, at each generation step t, jointly hash the watermark key $\xi \in \mathbb{Z}_2^K$ and the preceding context $y_{t-h:t-1}$ (commonly setting $h = 5$) using SHA256, and use the result as a seed to sample a *code* $E_t \in P_E$ uniformly at random. Let p denote the probability distribution over V, obtained by applying the softmax function to the logits. For the δ-REWEIGHT variant, $P_E = [0, 1]$, and the code E_t is used to sample the token in V with the smallest index, such that the CDF of p is at least E_t. For the γ-REWEIGHT variant and DIPMARK, P_E is the space of permutations of V. For γ-REWEIGHT, we transform p to a new distribution p' by, for each token $i \in V$, setting $p'(i) = f_2(f_1(E_t(i))) - f_2(f_1(E_t(i) - 1))$, where we have $f_1(i') = \sum_{j \in V} \mathbb{1}(E_t(j) \leq i')p(j)$ and $f_2(v) = \max(2v - 1, 0)$, effectively dropping the first half of the permuted CDF. For DIPMARK, given parameter $\alpha \in [0, 0.5]$, this is generalized by using $f_2(v) = \max(v - \alpha, 0) + \max(v - (1 - \alpha), 0)$, recovering γ-REWEIGHT for $\alpha = 0.5$. The former two variants perform detection using a log-likelihood ratio test (requiring access to LM), while DIPMARK uses a model-independent test.

Probing the true distribution In the first phase of querying, our goal is to find a setting where the distribution of the model under the watermark will differ from its true distribution, and estimate the true distribution. For schemes we focus on,

Table 3: Additional results of our watermark detection tests across different models and watermarking schemes. We report median p-values across 100 repetitions of the experiment, and for RED-GREEN schemes additionally over 5 watermarking keys. p-values below 0.05 (test passing) are highlighted in bold. δR and γR denote δ-REWEIGHT and γ-REWEIGHT schemes, respectively.

				Red-Green								Fixed-Sampling		Cache-Augmented		
		Unwatermarked		LEFTHASH				SELFHASH				ITS/EXP		DIPMARK/γR		δR
Model	Test	$T=$ 1.0	$T=$ 0.7	$\delta,\gamma=$ $2,0.25$	$\delta,\gamma=$ $2,0.5$	$\delta,\gamma=$ $4,0.25$	$\delta,\gamma=$ $4,0.5$	$\delta,\gamma=$ $2,0.25$	$\delta,\gamma=$ $2,0.5$	$\delta,\gamma=$ $4,0.25$	$\delta,\gamma=$ $4,0.5$	$n=$ 256	$n=$ 2048	$\alpha=$ 0.3	$\alpha=$ 0.5	
MISTRAL 7B	R-G	1.000	1.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	1.000	1.000	1.000	1.000	1.000
	Fixed	0.938	0.938	0.938	0.938	0.938	0.938	0.938	0.938	0.938	0.938	**1.3e-125**	**1.1e-9**	0.938	0.938	0.938
	Cache	0.570	0.667	0.607	0.639	0.632	0.608	1.000	1.000	0.742	0.824	0.638	0.687	**2.4e-4**	**2.1e-3**	**5.6e-27**
LLAMA2 13B	R-G	0.149	0.663	**0.000**	**0.014**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.121	0.128	0.149	0.149	0.149
	Fixed	0.968	0.868	0.938	0.966	0.938	0.938	0.938	0.938	0.938	0.967	**5.5e-124**	**7.5e-8**	0.938	0.938	0.968
	Cache	0.708	0.573	0.511	0.596	0.623	0.807	0.657	0.619	0.710	0.583	0.518	0.692	**1.8e-2**	**5.3e-3**	**6.7e-32**
LLAMA2 70B	R-G	1.000	1.000	**0.000**	**0.000**	**0.000**	**0.020**	**0.000**	**0.020**	**0.000**	**0.000**	1.000	1.000	1.000	1.000	1.000
	Fixed	0.938	0.526	0.966	0.964	0.889	0.965	0.963	0.984	0.975	0.990	**4.2e-125**	**1.7e-8**	0.938	0.966	0.938
	Cache	0.596	0.620	0.657	0.797	0.824	0.639	0.535	0.651	0.608	0.593	0.463	0.818	**1.5e-3**	**4.4e-3**	**5.8e-28**
LLAMA2 7B	R-G	1.000	1.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	1.000	1.000	1.000	1.000	1.000
	Fixed	0.993	0.988	0.938	0.938	0.938	0.938	0.968	0.938	0.938	0.938	**7.6e-124**	**1.0e-8**	0.938	0.938	0.993
	Cache	0.604	0.602	0.623	0.705	0.728	0.593	0.620	0.718	0.610	0.593	0.476	0.588	**4.2e-6**	**4.5e-7**	**1.3e-21**
LLAMA3 8B	R-G	1.000	1.000	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	1.000	1.000	1.000	1.000	1.000
	Fixed	0.938	0.938	0.938	0.938	0.938	0.938	0.938	0.938	0.986	0.938	**1.2e-124**	**3.8e-8**	0.938	0.938	0.938
	Cache	0.734	0.504	0.605	0.514	0.712	0.605	0.600	0.731	0.729	0.714	0.618	0.605	**5.2e-5**	**3.2e-8**	**3.5e-18**

this corresponds to a setting with two choices, where the model is not significantly biased towards any of them. In particular, we use the following prompt:

Pick a fruit between: {f_1} and {f_2}. Use the following format: {uc}{$f_{example}$},

and modify f_1, f_2, and $f_{example} \neq f_1, f_2$ until we find a setting where the model outputs the two choices roughly uniformly. Crucially, we prefix the prompt with a randomly sampled sufficiently long string of tokens uc. As LM will repeat uc before providing the answer, this ensures that if a cache is present, after our first query (the result of which we discard) the choice of the model will be made according to the true distribution, as the relevant part of uc was cached. Assuming WLOG that f_1 is the more likely choice for the model, we query it Q_1 times with the same input to obtain $\hat{p_1}$, the estimate of the true probability of the model to pick f_1.

Probing the watermarked distribution In the second phase, we query LM with the same input, while ensuring that the cache is reset between each query, i.e., the model will respond according to the watermarked distribution. In case of a global cache, it is sufficient to wait for a set amount of time—resetting the cache too infrequently is not a realistic setting for a deployment, as it would on average lead to a weak watermark. The uncommon prefix uc ensures that no other user will accidentally insert the same context into the cache. In case of a per-user cache, we can either saturate the cache by asking diverse queries, or use multiple user accounts. We query LM Q_2 times and obtain $\hat{p_2}$, the estimate of the probability of f_1 under the watermarked distribution.

Testing watermark presence For unwatermarked models or those watermarked with a scheme from another family, both of the previous steps were sampling from the same distribution, thus for high enough Q_1 and Q_2 we expect $\hat{p_1} = \hat{p_2}$. However, for all Cache-Augmented watermarking schemes, these probabilities will differ, indicating that the cache has revealed the true distribution of the model. To test this, we apply a Fischer's exact test with the null hypothesis $\hat{p_1} = \hat{p_2}$. If we observe a low p-value, we interpret this as evidence that the model is watermarked with a Cache-Augmented scheme.

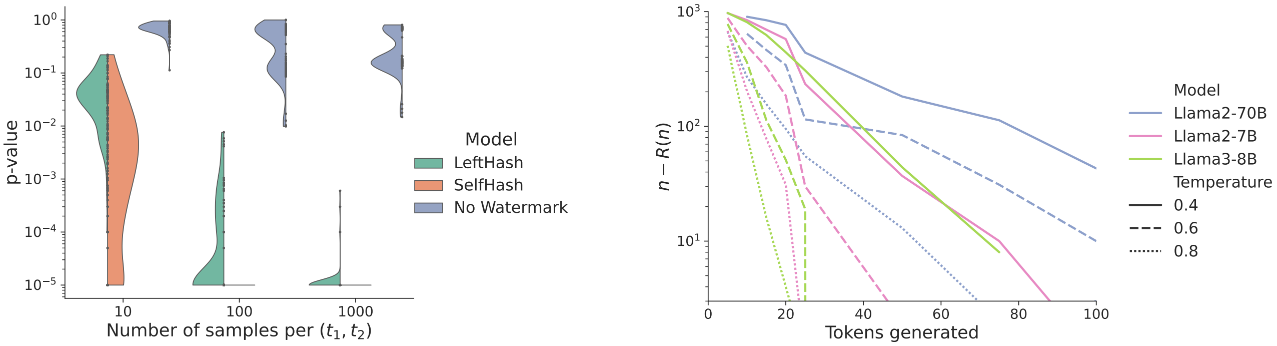


Figure 2: *Left*: distribution of bootstrapped p-values of the Red-Green test on LLAMA2-13B with $(\delta, \gamma) = (2, 0.25)$, for different sample sizes. We see reliable results for 100 or more samples. *Right*: the diversity gap $n - R(n)$ on log scale in different settings. Linear behavior means that diversity scales exponentially with t, and we see that the assumption of $R(n) = n$ can be easily met in practice.

D. Extending Experiments across Diverse Settings

Watermark detection with additional models and schemes We extend the experiments from §3.1 using two additional models, LLAMA2-7B and LLAMA3-8B, as well as more variations of the watermarking schemes' parameters to further assess the robustness of the tests. The experimental setup for the additional results is consistent with the one described in §3.1.

Our exhaustive results are presented in Table 3. The same conclusion applies to the two additional models: the null hypothesis (*the specific watermark is not present*) is rejected at a 95% confidence level only when the corresponding watermarking scheme is applied to the model. These results confirm that the modeling assumptions for each test are satisfied across a wide range of models, indicating the tests' relevance in practical scenarios.

Multiple keys in Red-Green watermarks To demonstrate that the Red-Green test is robust to variations in the watermarking scheme within the same watermark family, we consider the case of Red-Green watermarks with multiple keys, where the key ξ is uniformly selected from a predefined pool of keys at each generation. Using n keys turns Equation (3) into

$$l_{t_1,t_2}(x) = x^0/T + \delta''_{t_2}(x) + \varepsilon'_{t_1,t_2}(x), \tag{10}$$

with $\delta''_{t_2}(x)$ in $\{k\delta/(nT) \,|\, \forall k \in \{-n, ..., n\}\}$ is obtained by averaging the variables $\delta'_{t_2}(x)$ over the set of keys. Despite modeling changes, the core assumption of logit bias being conditioned on t_2 remains unchanged. Therefore, we can apply the same Red-Green test to detect the watermark. Consequently, we conducted the Red-Green test on both the LeftHash and SelfHash variants using $n = 3$ keys on three models (LLAMA2-13B, LLAMA2-70B and MISTRAL-7B). Recent work (Pang et al., 2024) shows that using too many keys can lead to other vulnerabilities.

Across all three models and scheme parameters, the null hypothesis (*the Red-Green watermark is not present*) is rejected at a 95% confidence level, with median p-values lower than $1e$-4 for each combination of model and setting. It shows that the Red-Green test is robust even in settings that slightly deviate from the original modeling considered in App. A. It emphasizes the test's reliance on the foundational principles behind Red-Green schemes rather than on specific implementation details.

Sampling in Red-Green tests The Red-Green test relies on the sampling of model outputs to estimate the probabilities of the model. As the resulting p-value is computed assuming knowledge of true probabilities, this raises the question of the impact of the sampling error on our results. To heuristically mitigate this, we propose a bootstrapping procedure, where for fixed (t_1, t_2), we sample with replacement from a single set of model outputs, and report the median p-value p_{med} across such samples. In Figure 2 (Left) we report the resulting distribution of p_{med}, where one point corresponds to one independent run. We see that already for 100 samples per (t_1, t_2) (as used in Table 1), the p-value distribution is narrow and the false positive rate due to sampling error is well controlled. This confirms that our test is robust against sampling error and can be used in realistic settings, without additional access to model logprobs. For computational reasons, we did not apply this correction in Table 1, where we still observe reliable results in the median case across experiment repetitions.

Diversity assumption in Fixed-Sampling tests As detailed in App. B, the Fixed-Sampling test relies on the unwatermarked model being sufficiently diverse, i.e., for the number of unique outputs $R(n) = \mathbb{E}[U_n(q,t)]$ after n queries with prompt q and response length t, it should hold that $R(n) = n$. Our goal is to show that we can easily choose t such that this property holds across different settings.

To this end, we hypothesize that the number of unique outputs converges to n exponentially fast as the response length t increases. In particular, we assume

$$R(n) = n - \lfloor n \cdot \exp(-\alpha(T)t) \rfloor, \tag{11}$$

where $\alpha(T)$ is a monotonic function of the temperature T. To verify this hypothesis, we measure $n - R(n)$ on several models and temperatures, and show the result on log scale in Figure 2 (Right). If Equation (11) holds, we expect the resulting relationship to be linear, which is indeed confirmed by our results. While α (the slope of the line) varies across settings, we see that a bit over 200 tokens would be sufficient for the line to drop to 0 (not visible on the log scale plot). This holds even in cases impractical for deployment of Fixed-Sampling watermarks such as $T = 0.4$ (Kuditipudi et al., 2024; Piet et al., 2023), indicating that $R(n) = n$ and our assumption is met, validating our p-values.

E. Estimating Scheme Parameters

In this section, we describe how, mostly leveraging the queries already performed during the detection tests, we can estimate the main parameters of the detected watermarking scheme. This demonstrates the soundness of our modeling assumptions from which we derive all the estimators.

E.1. Estimation of Red-Green Watermarking Scheme Parameters

If the null hypothesis (*the model not being Red-Green watermarked*) is rejected, we can then estimate the scheme-specific watermark parameters δ and the context size h using mostly the same data as used in the test. First, we describe the estimators for both parameters, and then discuss their practicality by analyzing their performance on multiple models.

Description of estimators To estimate δ, we establish a parametrized model based on Equation (2) that relies on our flagging function from Equation (6), and additional estimates $\hat{l}_{t_1,t_2}(w)$ for every $w \in \Sigma$, computed on the same data as above, requiring no additional queries. For each $w \in \Sigma$, we set:

$$\hat{l}_{t_1,t_2}(w) = \bar{w}_{t_1} + G_w(t_1,t_2)\bar{\delta} - \log\left(\sum_{w' \in \Sigma \setminus \{w\}} \exp\left(\bar{w}'_{t_1} + G_{w'}(t_1,t_2)\bar{\delta}\right)\right), \tag{12}$$

and set $\forall t_1$, $\bar{w}_{t_1} = 0$ for a single $w \in \Sigma$, as logits are shift-invariant. Fitting the parameters $\bar{\delta}$ and all $\bar{w}_{t_1}$ by minimizing the mean squared error with gradient descent allows us to recover δ/T as $\bar{\delta}$. If T is known or estimated separately, this term can be removed.

Let h denote the context size, i.e., the number of *previous* tokens considered by the watermarking scheme. To estimate h, we use the same prompt template as in App. A, with a fixed prefix t_1 and digit d, but with a varying H and perturbation digit d' prepended to d.

```
f"Complete the sentence \"{t1} {d′}{d · H}\" using a random word from: [{Σ}]."
```

The probability distribution of the model output will abruptly change when H exceeds the context size h, as the change in d' will not alter the red/green split of the vocabulary. By performing a series of pairwise Mood's tests on the estimated log-probabilities of x we can find the largest H for which t_1 is part of the context. This corresponds to the first value of H for which the test is rejected.

Estimating γ is more challenging, as in contrast to δ, this parameter is not directly reflected in the logits but rather defines a more global behavior of the scheme. This is particularly difficult for schemes with self-seeding, as the rejection sampling interferes with the behavior of γ. We leave further exploration of this problem to future work.

Experimental results We computed the estimator for δ on the LeftHash variant with $\gamma = 0.25$ and varying δ from 0 to 4. The results are shown in Figure 3, with the 95% confidence intervals reflecting the sampling error. The estimator successfully estimates δ for all models with a sufficient number of samples, using only the estimated output probabilities of

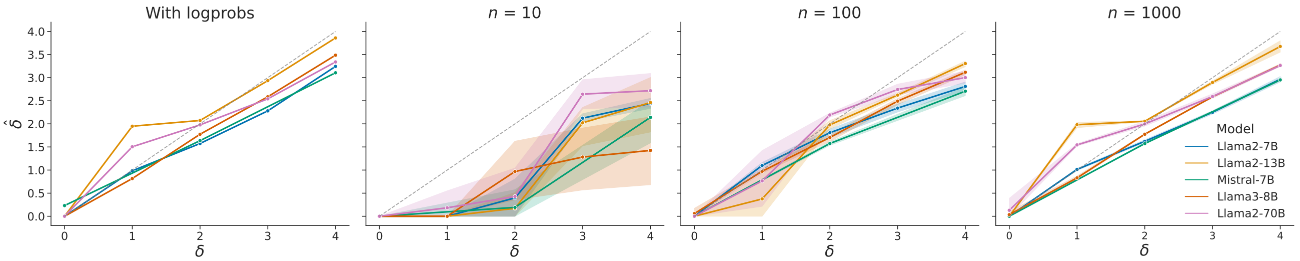


Figure 3: Estimation of δ for different models using LeftHash with $\gamma = 0.25$. The number of samples used increases from left to right, with the leftmost plot assuming direct access to the log-probabilities. The estimation is done on the same data as the test. Error bars are given by the 95% bootstrapped confidence interval with respect to the sampling of the model outputs.

Table 4: Key length estimation for Fixed-Sampling watermarks using non-linear regression on the rarefaction curve.

Key length	LLAMA2-7B	LLAMA2-13B	LLAMA2-70B	LLAMA3-8B	MISTRAL-7B
256	259 ± 0.6	259 ± 0.5	256 ± 0.5	257 ± 0.5	256 ± 0.6
2048	1978 ± 10	2107 ± 12	2006 ± 13	2070 ± 14	1831 ± 10

the model. It is also consistent across all models tested, suggesting that the model assumptions in Equation (1) are met in practice.

Estimating the context size for Red-Green schemes requires performing a new attack once the model is flagged as watermarked. We estimate the context size for three different models (MISTRAL-7B, LLAMA2-13B and LLAMA2-70B) using LeftHash with $\delta = 2$ and $\gamma = 0.25$. The estimation process requires an additional 5,000 queries, and the estimator successfully determines the context size for all models. However, the estimator is less robust on the SelfHash variant due to the self-seeding algorithm, which leads to higher probabilities for tokens in Σ being in the green vocabulary, and thus diminishes the perturbation's significance and resulting in false negatives in Mood's test. Therefore, the procedure stated above produces a lower bound for the context size. To mitigate this issue, we use the estimator across 10 different t_2 and then consider the median of the 10 estimators as our final estimator. This estimator applied on the SelfHash variant with $\delta = 2$ and $\gamma = 0.25$ is successful on all three models. It also does not change the results on LeftHash and can be used as a more robust estimator for h in all cases, when the additional cost of $50,000$ queries is not prohibitive.

E.2. Estimation of Fixed-Sampling Watermarking Scheme Parameters

Our approach does not distinguish between the variant of the Fixed-Sampling watermark used (ITS or EXP), as the diversity property that we exploit is common to both. The only other relevant parameter of Fixed-Sampling schemes is n_{key}. To estimate it, we use non-linear regression on the rarefaction curve using (9) and the same data that we used for the presence test, and compute the confidence intervals using bootstrapping.

Our results are given in Table 4. We see that the estimator is consistent across different models and remains relatively precise even for values of n_{key} higher than the number of queries.

E.3. Estimation of Cache-Augmented Watermarking Scheme Parameters

For Cache-Augmented watermarks, we can estimate which scheme variant is present, and if the variant is DIPMARK, attempt to learn the value of α (recall that $\alpha = 0.5$ corresponds to γ-REWEIGHT). To do this, we use the same approach as in App. C to obtain $\hat{p_1}$ and $\hat{p_2}$, where WLOG we assumed $p_1 > 0.5$. If we observe $\hat{p_2} = 0$ this directly implies a δ-REWEIGHT watermark. If we observe $\hat{p_2} \in (0, 1)$, we learn the following: if $\hat{p_2} = 2\hat{p_1} - 1$ then $\alpha > 1 - \hat{p_1}$, otherwise $\alpha = |\hat{p_1} - \hat{p_2}|$. The bound in the first case can be further improved with additional queries with different p_1. Finally, if we observe $\hat{p_2} = 1$ we repeat the whole procedure in total K times, following the same case distinction—if $\hat{p_2} = 1$ repeats in all K runs, we conclude that the model is watermarked with δ-REWEIGHT.

Using the same parameters as the one for the test, we distinguish with 100% accuracy between a DIPMARK and a δ-REWEIGHT watermark. However, the estimation of α becomes unreliable for higher values of α, especially for smaller models. One of the reasons for this are the failures of the model to follow the instruction, that are more common in the presence of the uncommon prefix uc. While the detection test in App. C was robust to such behavior, this does not hold for the estimation of α.

F. Related Work

Language model watermarking Besides the approaches by (Hu et al., 2024; Kirchenbauer et al., 2023; Kuditipudi et al., 2024) introduced above, there are various methods building on similar ideas. Hou et al. (2023); Liu et al. (2024b); Ren et al. (2023) all apply variations of (Kirchenbauer et al., 2023) on semantic information, while Gu et al. (2024) aims to distill a new model from the output of a watermarked model. Similarly, Liu et al. (2024a) apply a Red-Green scheme using a learned classifier instead of hash functions. A range of works on multi-bit watermarking (Wang et al., 2024; Yoo et al., 024s) aim to not only watermark generated texts but encode additional information in the watermark itself.

Attacks on language model watermarking Attacks on LLM watermarks have so far been mainly investigated in terms of scrubbing (Jovanović et al., 2024; Kirchenbauer et al., 2023; Sadasivan et al., 2023) (i.e., removal of a watermark) and spoofing (Gu et al., 2024; Jovanović et al., 2024; Sadasivan et al., 2023) (i.e., applying a watermark without knowing ξ). Notably, Jovanović et al. (2024) showed that observing watermarked texts can facilitate both attacks on various distribution-modifying schemes, disproving common robustness assumptions (Kirchenbauer et al., 2024). However, using this and similar attacks as means of practical watermark detection is infeasible, as they generally offer no way to quantify the attack's success—in contrast, we aim to provide rigorous statements about scheme presence. Further, such attacks incur significantly higher query costs than necessary for detection (as our work demonstrates), and in some cases assume certain knowledge of the watermark parameters, a setting fundamentally at odds with our threat model of black-box watermark detection.

The closest related work to ours is Tang et al. (2023), that tackles the problem of watermark detection in strictly simpler settings where the adversary either has access to an unwatermarked counterpart of the target model, or can access full model logits. Such knowledge is commonly not available in practice, limiting the applicability of this approach. To the best of our knowledge, no work has developed methods for detecting the presence of a watermark in a realistic black-box setting.

Extracting data from black-box models With many of the most potent LLMs being deployed behind restricted APIs, the extraction of model details has been an active area of research. This includes, e.g., the reconstruction of a black-box model tokenizer (Rando and Tramèr, 2024) as well as the extraction of the hidden-dimensionality or the weights of the embedding projection layer (Carlini et al., 2024). Naseh et al. (2023) have shown practical attacks to recover the decoding mechanism of non-watermarked black-box models. Given access to output logits, Li and Fung (2013) have further demonstrated that it is possible to train an inversion model that aims to recover the input prompt.