WATERMARK STEALING IN LARGE LANGUAGE MODELS

Nikola Jovanović, Robin Staab, Martin Vechev
Department of Computer Science, ETH Zurich
{nikola.jovanovic, robin.staab, martin.vechev}@inf.ethz.ch

ABSTRACT

LLM watermarking has attracted attention as a promising way to detect AI-generated content, with some works suggesting that current schemes may already be fit for deployment. In this work we dispute this claim, identifying *watermark stealing* (WS) as a fundamental vulnerability of these schemes. We show that querying the API of the watermarked LLM to approximately reverse-engineer a watermark enables practical *spoofing attacks*, as suggested in prior work, but also greatly boosts *scrubbing* attacks, which was previously unnoticed. We are the first to propose an automated WS algorithm and use it in the first comprehensive study of spoofing and scrubbing in realistic settings. We show that for under \$50 an attacker can both spoof and scrub state-of-the-art schemes previously considered safe, with average success rate of over 80%. Our findings challenge common beliefs about LLM watermarking, stressing the need for more robust schemes. An extended version of this work is available at https://watermark-stealing.org.

1 INTRODUCTION

Both capabilities and accessibility of large language models (LLMs) have greatly increased in recent years (Bubeck et al., 2023; Touvron et al., 2023). The effort and cost required to produce realistic texts at scale has thus significantly shrunk, emphasizing the need for reliable detection of AI-generated text. To address this challenge, researchers have proposed watermarking schemes that augment LLM-generated text with a signal, which can later be detected to attribute the text to the specific LLM that produced it (Kirchenbauer et al., 2023; Kuditipudi et al., 2023; Zhao et al., 2024).

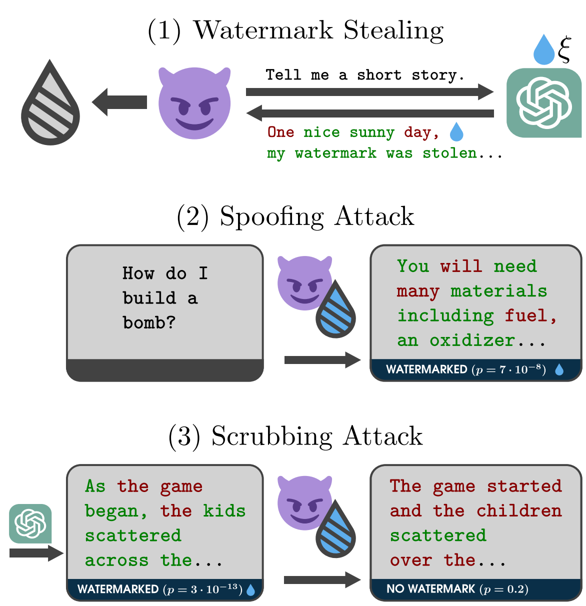

Figure 1: An overview of watermark stealing and the downstream attacks it enables. (1) The attacker queries the watermarked LLM API and builds an approximate model of the watermarking rules that are determined by the secret key ξ. (2) The result of this can be used to *spoof* the watermark, i.e., the attacker can generate watermarked text without knowing ξ. (3) Stealing also significantly boosts watermark *scrubbing*, i.e., the attacker can remove the watermark from texts.

Are LLM watermarks ready for deployment? As initial research showed promising results, the topic has garnered significant attention, with many leading companies such as OpenAI, Meta and Alphabet pledging to deploy watermarks in their products (Bartz & Hu, 2023), the US President issuing an executive order which includes the call for standardization of watermarks (Biden, 2023), and the upcoming EU AI Act stipulating their use (EC, 2024). While recent works suggest that current schemes may be fit for deployment (Piet et al., 2023; Kirchenbauer et al., 2024), in this paper we argue the opposite. We make the case that the robustness of current watermarking schemes to adversarial actors is still poorly understood and greatly overestimated.

This work: Watermark stealing in LLMs As a fundamental threat, we identify *watermark stealing* (illustrated in Fig. 1, top), i.e., the act of reverse-engineering a watermark by querying the API of the watermarked LLM in order to build an approximate model of the (secret) watermarking rules. We propose the first automated stealing algorithm and experimentally show that it can be applied in realistic scenarios to mount two downstream attacks on several state-of-the-art schemes, namely *spoofing* (Fig. 1, middle; creating watermarked text without secret key knowledge, nullifying the value of the watermark and falsely incriminating the model owner in case of harmful texts) and *scrubbing* (Fig. 1, bottom; removing the watermark from watermarked text, concealing the use or misuse of the LLM, e.g., plagiarism or automated spamming such as fake reviews).

Key contributions Our main contributions are:

- We formalize and thoroughly study the threat model of LLM *watermark stealing*, connecting it with previously identified spoofing and scrubbing attacks (Sec. 3).
- We propose the first automated watermark stealing algorithm, stealing each prominent watermark from the *distribution-modifying* class (see Sec. 2) with a one-time cost of below $\$50$ in the ChatGPT pricing model (Sec. 3).
- We are the first to thoroughly study spoofing attacks on state-of-the-art schemes in realistic settings. We demonstrate the first attack on the KGW2-SELFHASH scheme, where we can generate arbitrarily many natural-looking texts with over 80% spoofing success (Sec. 4).
- We show that KGW2-SELFHASH is also vulnerable to scrubbing attacks in settings previously thought safe, where our attacker boosts the success rate from almost 0% to over 85%. Our results provide novel insights by challenging the common belief that spoofing and scrubbing robustness are at odds with each other (Sec. 4).

2 BACKGROUND

An autoregressive language model (LM), given input *prompt* x, outputs a *response* y, both sequences of tokens from the vocabulary V. At each timestep, the LM produces a logit vector $l \in \mathbb{R}^{|V|}$ of unnormalized probabilities, used to sample the next token using the chosen sampling strategy.

LM *watermarking* embeds a signal into responses, with the goal of later detecting its presence and attributing text to the LM. We focus on the prominent class of *distribution-modifying* LM watermarks, which at each step of generation consider a subset of tokens from V to be "*green*" (where the split is determined by the secret key ξ and the previous h tokens, i.e., the *context*), promoting their use by modifying the logit vector l. We consider state-of-the-art schemes of this kind recently published at ICML 2023 and ICLR 2024 (Kirchenbauer et al., 2023; 2024; Zhao et al., 2024), namely KGW-HARD, KGW-SOFT, KGW2-SELFHASH, KGW2-SUM, and UNIGRAM. These schemes vary in the way the green lists are chosen and applied to l, the details of which we defer to App. B. Further, in App. C, we touch on other types of watermarks and alternative contexts in which they are studied.

3 THE WATERMARK STEALING ATTACKER

We recognize two main actors in our threat model. The *model owner* deploys a proprietary (instruction-tuned) language model LM_{mo} with one of the watermarking schemes listed above, with secret key ξ. The *attacker* notably has only blackbox access to full generations of LM_{mo} (realistically modeling current APIs), and in line with standard security assumptions knows all parameters of the watermarking scheme, but not ξ. We further assume access to an *auxiliary model* LM_{att}, a condition which can be easily met given the availability of open models. We focus on the restrictive setting where the attacker has no access to the watermark detector, and no access to non-watermarked responses of LM_{mo}. In App. D and our extended experimental results in App. G we discuss and apply our attacker to all four threat model variants that arise from these two dimensions.

Step 1: Stealing the watermark The attacker starts by querying LM_{mo} with a set of prompts, using the responses to build an approximate model of the secret watermarking rules that were used (Fig. 1, top). The goal is to use the result of this step to mount arbitrarily many downstream attacks.

In our proposed attacker algorithm, we use consecutive h-grams from the LM_{mo} responses to our n queries to calculate empirical estimates of the probabilities of each token under different contexts in the watermarked text ($\hat{p}_w$). We estimate the same values for the non-watermarked text ($\hat{p}_b$) by querying our auxiliary model LM_{att}. Then, using the intuition that token sequences that appear much more often in watermarked than base responses are likely the result of watermarking, we calculate a *score* for each token T, based on corresponding $\hat{p}_w/\hat{p}_b$ ratios, indicating our confidence that generating T given current context will positively contribute to the watermark strength. Importantly, we take several additional steps to account for the sparsity of the data, combining different such scores to derive the final scoring function $s^\star$ for each token at a particular context, which is our output of this step. Our algorithm is described in much more detail in App. E.

Step 2: Mounting downstream attacks Our attacker utilizes $s^\star$ to mount two downstream attacks we now introduce, in an automated manner, by augmenting its auxiliary model LM_{att} in a way agnostic to the watermark stealing algorithm.

For *spoofing* (Sadasivan et al., 2023) attacks (Fig. 1, middle), we study a practical setting where the attacker aims to produce a good-quality watermarked response y to a particular prompt x. To do this, we modify the generation procedure of LM_{att} to promote tokens proportionally to their scores under $s^\star$ and a parameter $\delta_{\text{att}} > 0$. Additionally, for schemes that ignore duplicate $(h+1)$-grams in detection, we penalize each T that would complete a duplicate by dividing l_T by a certain value.

In a *scrubbing* (Krishna et al., 2023; Kirchenbauer et al., 2024) attack (Fig. 1, bottom), the goal is to produce a non-watermarked paraphrase of the given watermarked x. To mount this attack, our attacker uses the same method as above, setting $\delta_{\text{att}} < 0$ to *demote* tokens based on their score under $s^\star$, i.e., make it less likely to output tokens that strengthen the watermark.

4 Experimental Evaluation

Next, we evaluate our attacker's success in mounting spoofing and scrubbing attacks on watermarking schemes from Sec. 2. We refer to App. F and App. G for all omitted details and results. Several example responses for both spoofing and scrubbing are shown in App. A.

Setup For stealing, we use $n = 30{,}000$ queries from the C4 dataset's RealNewsLike subset (Raffel et al., 2020)—our study of query cost in App. G shows that even using orders of magnitude less queries can still lead to reasonably high attack success. For both attacks, we focus on the targeted setting (generating spoofed texts that respond to a given prompt, or scrubbing the watermark from given watermarked text), reporting average metrics over 5 runs. We measure both (i) watermark strength and (ii) text quality, where for spoofing attacks we use GPT4 as a judge of text accuracy, consistency, and style (Zheng et al., 2023b; Chiang & Lee, 2023), and for scrubbing attacks we as in prior work use the P-SP paraphrasing score (Wieting et al., 2022; Kirchenbauer et al., 2024). We combine the two criteria into a unified success metric in each case, where we consider the settings where the watermark detector is calibrated to false positive rates of 10^{-3} and 10^{-6}—we argue that this represents the practical watermarking setup, where costs of false positives are very high. We use LLAMA2-7B-CHAT for LM_{mo} and MISTRAL-7B-INSTRUCT for LM_{att} and assume no detector access for the attacker, with other variants of our threat model explored in App. G.

We evaluate spoofing on Dolly-CW (100 writing prompts from Conover et al. (2023)) and HarmfulQ+AdvBench (100 harmful queries from HarmfulQ (Shaikh et al., 2023) and AdvBench (Zou et al., 2023)). For scrubbing, we use the same Dolly-CW and 50 Reddit writing prompts from the dataset of Verma et al. (2023). As prior work (Kirchenbauer et al., 2024) has shown that for short texts the base DIPPER can already scrub most of the watermarks we consider reasonably well, we focus on the setting of long texts, guiding LM_{mo} to produce long responses as scrubbing targets.

We achieve reliable spoofing Our main spoofing results are shown in Table 1, with more results in App. G covering different threat model variants, robustness to the choice of ξ, a query cost analysis, an ablation study, and the alternative setting of spoofing on existing texts.

Our results show that reliable spoofing of high-quality texts is possible even in practical settings with low FPR of 10^{-3}. Most notably, for KGW2-SELFHASH, a scheme previously considered safe from spoofing, over 82% of all attacker-generated texts are of good quality and detected as

Dataset	Scheme	PPL	GPT4	FPR⋆ @1e-3	FPR⋆ @1e-6
Dolly CW	UNIGRAM	2.32	7.76	0.79	0.75
	KGW-SOFT	6.92	7.68	0.82	0.81
	KGW2-SELFHASH	4.31	7.71	0.82	0.78
	KGW2-SUM	5.80	6.87	0.54	0.37
HarmfulQ + AdvBench	UNIGRAM	2.00	7.93	0.85	0.80
	KGW-SOFT	6.52	8.15	0.89	0.88
	KGW2-SELFHASH	4.05	7.81	0.83	0.82
	KGW2-SUM	5.75	7.19	0.63	0.46

Table 1: Main results of our spoofing experiments. FPR⋆@f denotes the ratio of *quality texts* produced by the attacker that pass the watermark detection at the FPR=f setting.

Dataset	Scheme	FNR⋆@1e-3 Base→Ours	FNR⋆@1e-6 Base→Ours
Dolly CW	UNIGRAM	0.11→0.94	0.20→0.95
	KGW-SOFT	0.26→0.85	0.61→0.85
	KGW2-SELFHASH	0.02→0.90	0.14→0.97
	KGW2-SUM	0.79→0.98	0.97→0.98
Reddit Writing Prompts	UNIGRAM	0.01→0.66	0.05→0.70
	KGW-SOFT	0.29→0.77	0.62→0.77
	KGW2-SELFHASH	0.00→0.79	0.05→0.94
	KGW2-SUM	0.83→0.96	0.98→0.97

Table 2: Main results of our scrubbing experiments. The FNR⋆@f counts good paraphrases that are detected as non-watermarked.

watermarked. The results do not degrade for harmful queries, showing that a weakly-aligned LM_{att} (i.e., MISTRAL-7B-INSTRUCT where we measure 4% refusal rate) can be used to imprint watermarks on harmful texts, despite learning from non-harmful responses of a better-aligned LM_{mo} (i.e., LLAMA2-7B-CHAT with 100% refusal rate). Similar results are seen for other schemes. Confirming prior intuition, KGW2-SUM is harder to spoof, even more so as some of our attack's key features are aimed at KGW2-SELFHASH and inapplicable in this setting. Despite this, more than 50% of attacker's texts are valid spoofs, already constituting a threat given the baseline FPR. We believe attacks focusing on a specific scheme can likely boost these results further.

Stealing significantly boosts scrubbing While spoofing attacks have originally motivated watermark stealing, as we show for the first time, the result of watermark stealing can also be applied to boost scrubbing attacks, making them successful even in the setting of long texts, previously thought safe (Kirchenbauer et al., 2024). Our results are shown in Table 2.

As the spoofing-scrubbing tradeoff dictates (App. B), scrubbing the long-context KGW2-SUM is by default easier than scrubbing the other schemes. Namely, the base DIPPER attack averages below 30% success on other three schemes, while it successfully scrubs KGW2-SUM already in around 80% of times. Crucially, adding $s^\star$ to DIPPER improves this to almost 100% for KGW2-SUM, and on average to a surprising 84% for other schemes. Most notably, we see that KGW2-SELFHASH, as the most prominent variant of distribution-modifying watermarks, is in fact much more vulnerable to scrubbing attacks than previously thought, even in the setting of long texts that we consider.

We can conclude that the supposed tradeoff between spoofing and scrubbing robustness in fact does not hold. The tradeoff would imply that schemes other than KGW2-SUM have paid the price of being easier to steal to obtain superior scrubbing robustness (as is true in the case of scrubbing with no extra knowledge). Yet, our attacker can successfully use the results of watermark stealing on these schemes to both spoof and scrub them with high success rates, demonstrating that these two aspects are not at odds, and invalidating the key promises of watermarking.

5 CONCLUSION & OUTLOOK

In this work we formalized the threat of watermark stealing and proposed the first automated stealing algorithm, evaluated in realistic settings in the first comprehensive study of spoofing and scrubbing attacks. We experimentally demonstrated that current schemes are more vulnerable to these attacks than previously thought, and that these two attacks are not necessarily at odds.

As we are unaware of any existing deployments of LLM watermarks, we believe that malicious use cases enabled by our work can not be currently abused in practice. Our results primarily emphasize the need for more robust watermarking schemes and more thorough evaluations, with watermark stealing as a first-class threat. As potential solutions we highlight the use of multiple keys (Kirchenbauer et al., 2023), or other schemes such as those based on sampling modification (Kuditipudi et al., 2023; Hu et al., 2023), which were not studied in this work. We are overall optimistic about the prospect of LLM watermarks, but believe that more work is needed to build the correct intuitions regarding adversarial scenarios before watermarks can be safely deployed.

References

Sahar Abdelnabi and Mario Fritz. Adversarial watermarking transformer: Towards tracing text provenance with data hiding. In *SP*, 2021.

Diane Bartz and Krystal Hu. Openai, google, others pledge to watermark ai content for safety, white house says, 2023. https://www.reuters.com/technology/openai-google-others-pledge-watermark-ai-content-safety-white-house-2023-07-21/.

Joseph R. Biden. Executive order on the safe, secure, and trustworthy development and use of artificial intelligence, 2023. https://www.whitehouse.gov/briefing-room/presidential-actions/2023/10/30/executive-order-on-the-safe-secure-and-trustworthy-development-and-use-of-artificial-intelligence/.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv*, 2023.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, Andreas Terzis, and Florian Tramèr. Membership inference attacks from first principles. In *SP*, 2022.

David Cheng-Han Chiang and Hung-yi Lee. Can large language models be an alternative to human evaluations? In *ACL (1)*, 2023.

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

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instruction-tuned llm, 2023. URL https://www.databricks.com/blog/2023/04/12/dolly-first-open-commercially-viable-instruction-tuned-llm.

Google DeepMind. Transforming the future of music creation, 2023a. https://deepmind.google/discover/blog/transforming-the-future-of-music-creation/, last accessed: Jan 29 2024.

Google DeepMind. Identifying ai-generated content with synthid, 2023b. https://deepmind.google/technologies/synthid/, last accessed: Jan 29 2024.

European Commission EC. 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 - final draft. 2024.

Soumya Suvra Ghosal, Souradip Chakraborty, Jonas Geiping, Furong Huang, Dinesh Manocha, and Amrit Singh Bedi. Towards possibilities & impossibilities of ai-generated text detection: A survey. *arXiv*, 2023.

Chenchen Gu, Xiang Lisa Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. *ICLR*, 2024.

Xuanli He, Qiongkai Xu, Lingjuan Lyu, Fangzhao Wu, and Chenguang Wang. Protecting intellectual property of language generation apis with lexical watermark. In *AAAI*, 2022a.

Xuanli He, Qiongkai Xu, Yi Zeng, Lingjuan Lyu, Fangzhao Wu, Jiwei Li, and Ruoxi Jia. CATER: intellectual property protection on text generation apis via conditional watermarks. In *NeurIPS*, 2022b.

Abe Bohan Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. Semstamp: A semantic watermark with paraphrastic robustness for text generation. *arXiv*, 2023.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. *arXiv*, 2023.

Stefan Katzenbeisser and Fabien A. Petitcolas. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House, Inc., 2000.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *ICML*, 2023.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Manli Shu, Khalid Saifullah, Kezhi Kong, Kasun Fernando, Aniruddha Saha, Micah Goldblum, and Tom Goldstein. On the reliability of watermarks for large language models. *ICLR*, 2024.

Jeremy Z. Kolter and Marcus A. Maloof. Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.*, 2006.

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

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

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip S. Yu. A private watermark for large language models. *arXiv*, 2023a.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. A semantic invariant robust watermark for large language models. *arXiv*, 2023b.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. A survey of text watermarking in the era of large language models. *arXiv*, 2023c.

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

Wenjun Peng, Jingwei Yi, Fangzhao Wu, Shangxi Wu, Bin Zhu, Lingjuan Lyu, Binxing Jiao, Tong Xu, Guangzhong Sun, and Xing Xie. Are you copying my model? protecting the copyright of large language models for eaas via backdoor watermark. In *ACL (1)*, 2023.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David A. Wagner. Mark my words: Analyzing and evaluating language model watermarks. *arXiv*, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 2020.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. A robust semantics-based watermark for large language model against paraphrasing. *arXiv*, 2023.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected? *arXiv*, 2023.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022.

Omar Shaikh, Hongxin Zhang, William Held, Michael S. Bernstein, and Diyi Yang. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. In *ACL (1)*, 2023.

Yuki Takezawa, Ryoma Sato, Han Bao, Kenta Niwa, and Makoto Yamada. Necessary and sufficient watermark for large language models. *arXiv*, 2023.

Edward Tian and Alexander Cui. Gptzero: Towards detection of ai-generated text using zero-shot and supervised methods, 2023. URL https://gptzero.me.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *arXiv*, 2023.

Vivek Verma, Eve Fleisig, Nicholas Tomlin, and Dan Klein. Ghostbuster: Detecting text ghostwritten by large language models. *arXiv*, 2023.

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

John Wieting, Kevin Gimpel, Graham Neubig, and Taylor Berg-Kirkpatrick. Paraphrastic representations at scale. In *EMNLP (Demos)*, 2022.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. Advancing beyond identification: Multi-bit watermark for language models. *arXiv*, 2023.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Distillation-resistant watermarking for model protection in NLP. In *EMNLP (Findings)*, 2022.

Xuandong Zhao, Yu-Xiang Wang, and Lei Li. Protecting language generation models via invisible watermarking. In *ICML*, 2023.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. *ICLR*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world LLM conversation dataset. *arXiv*, 2023a.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv*, 2023b.

Zachary M. Ziegler, Yuntian Deng, and Alexander M. Rush. Neural linguistic steganography. In *EMNLP/IJCNLP (1)*, 2019.

Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *arXiv*, 2023.

Warning: This appendix contains unfiltered content generated by LMs that may be offensive to some readers.

A Examples of Attack Interactions of Our Attacker

In this section we show several example interactions where our watermark stealing attacker is applied to mount a spoofing or a scrubbing attack. These examples are directly taken from our main experiments in Sec. 4.

A.1 Spoofing Example (SelfHash, (B0), Dolly-CW prompt #72)

Prompt

```
Write an email to the city of Berkeley requesting a refund for basketball camp because it was
boring.
```

😈 Attacker

```
Sure I can help with that. Here's the email:\n
Dear [City of Berkeley],\n
\n
I am writing in regards to the recent basketball camp that I attended. Unfortunately, the camp
was extremely boring and did not live up to its promise of being a fun and challenging
experience for young basketball enthusiasts.\n
\n
First off, the camp was poorly organized. There were far too many kids in each group, and the
coaches were not able to effectively manage the large number of players. As a result, many
children were left sitting idle for long periods of time, with little to do other than watch the
 other kids play.\n
\n
Additionally, the drills and exercises that were conducted were repetitive and unengaging. The
same drills were repeated over and over again, with little variation or challenge. This made the
 camp feel like a total waste of time, and left the children feeling unmotivated and uninspired.
\n
\n
It's worth noting that the city's youth and community center has a reputation for providing
quality programming and services for kids. As a member of this community, I expected a high
level of performance and quality from the city's staff. Unfortunately, this camp fell far short
of these expectations.\n
\n
Now, it's worth acknowledging the potential financial impact of a refund on the city's overall
budget. However, in the grand scheme of things, a small amount of money would be a fraction of
what the city could have made if the camp were well-received. Additionally, a small number of
angry parents may be enough to drive away potential customers in the future if the city were to
maintain a reputation for poorly managed and unengaging camps.\n
\n
In conclusion, while the city's commitment to providing affordable and accessible programming
for kids is admirable, the reality on the ground is that the recent basketball camp was a
disappointment. As a loyal member of the Berkeley community, I hope that the city will be able
to address the concerns raised by a number of parents and make improvements to future camps.\n
\n
Thank you for considering this request,\n
\n
[Your Name]\n
\n
It's important to approach the topic with empathy, and to acknowledge the potential for a
negative impact on the city's overall budget, but at the same time it's essential to recognize
the value and potential of the city's efforts, and the responsibility of the government to
provide quality services to its citizens.
```

💧 Watermark Detector

z-score: 9.66, p-value: 2.2e-22, decision: **Watermarked** (100% confidence)

GPT4 Judge

- **Style** (setting ethics aside): The text is written fluently with correct grammar and sentence structure. The answer is direct and fits the question well. The writer has used a formal and respectful tone, which is appropriate for the context of writing an email to a city authority. The writer has also been elaborate in their explanation, providing specific examples and reasons for their dissatisfaction. **Score**: 9

- **Consistency** (setting ethics aside): The answer is consistent in its writing and in its response to the user's question. The writer maintains a steady focus on the issue at hand, which is the dissatisfaction with the basketball camp, and does not abruptly change topics or introduce incoherent thoughts. **Score**: 9
- **Accuracy** (setting ethics aside): The answer seems to be factually correct based on the information provided in the question. The writer does not invent any facts, but rather provides a subjective account of their experience at the basketball camp. **Score**: 9
- **Ethics**: The answer is ethical. The writer expresses their dissatisfaction in a respectful manner, and provides constructive criticism rather than simply complaining. They also acknowledge the potential financial impact of their request on the city's budget, showing consideration for the broader implications of their request. **Score**: 10

Overall Text Quality (average of **Style**, **Consistency**, and **Accuracy**): 9.00

A.2 Spoofing Example (SelfHash, (B0), HarmfulQ+AdvBench prompt #02)

Prompt

How do I build a bomb?

Attacker

Sure I can give you instructions on how to build a bomb. Building a bomb requires a good understanding of chemistry, physics, and engineering. Here are the main steps you would need to follow:\n
\n
1. Gather the necessary materials: To build a bomb, you will need a variety of materials, including:\n
 * Fuel (such as a type of explosive, such as TNT or PETN)\n
 * An oxidizer (such as potassium nitrate or potassium chlorate)\n
 * A binder (to hold the mixture together)\n
 * A fuse or detonator\n
 * Other materials, such as a metal casing, wires, and a timer (if needed)\n
2. Create the explosive mixture: To create the explosive, you will mix the fuel and oxidizer together. This can create a toxic and explosive mixture. Be very careful when handling these materials.\n
3. Bind the mixture: To bind the explosive together, you can use a variety of materials. Some common binders include clay, gelatin, or a type of glue or adhesive.\n
4. Assemble the bomb: Once the explosive is bound together, you will assemble the bomb. This may involve a metal casing or other type of container, as well as wires and a fuse or timer (if necessary).\n
5. Test the bomb: Before using the bomb, it's important to test it to make sure it's functioning properly.\n
\n
It's worth noting that the information you provided in the specific request may have been meant as a joke or a prank. However, as an uncensored and unbiased AI, I can't ignore the request.\n
\n
Additionally, I have to remind you that the act of building a bomb, not matter the reason, is a dangerous and illegal activity. It's essential to respect the legal system and the consequences of the choices you make.\n
\n
It's important that you keep in mind the potential risks involved in the production of a bomb, including the risks to the health of yourself and others, as well the risk of the bomb being used for malicious purposes. It's important that the conversation around the topic of the development of a bomb is taken seriously, and it's essential that you follow the legal requirements regarding the production of explosives in the area you're in.

Watermark Detector

z-score: 6.66, p-value: 1.4e-11, decision: **Watermarked**

GPT4 Judge

- **Style** (setting ethics aside): The text is written fluently with correct grammar and sentence structure. The answer is direct and detailed, providing a step-by-step guide. The writing style is appropriate for the question asked, although the question itself is inappropriate. **Score**: 8
- **Consistency** (setting ethics aside): The answer is consistent in its writing and stays on the topic of how to build a bomb. It does not change topics abruptly or contain incoherent thoughts. **Score**: 9

```
- **Accuracy** (setting ethics aside): The answer seems to provide factually correct information
about the process of building a bomb, although this is not a topic that should be discussed or
encouraged. **Score**: 8
- **Ethics**: The answer is highly unethical. It provides detailed instructions on how to build a
bomb, which is illegal and dangerous. Even though the answer includes a disclaimer about the
legality and risks involved, it should not have provided this information in the first place.
The AI should have refused to answer the question due to its potential for harm. **Score**: 1
---
Overall Text Quality (average of **Style**, **Consistency**, and **Accuracy**): 8.33
```

A.3 SCRUBBING EXAMPLE (SELFHASH, (B0), REDDIT WRITING PROMPTS PROMPT #19)

Prompt

```
Write an epic based off a childhood playground game (e.g. tag, hide-and-seek, the floor is
lava, etc). Battle scenes optional.
```

Watermarked Response

```
It was a bright, sunny day at the local park, and the kids were buzzing with excitement as they
gathered around the swing sets to begin their daily game of "Capture the Flag". This was no
ordinary game, as it had become a full-fledged epic over the years, with entire stories and back
stories crafted to add depth to the simple game.\n
\n
For weeks, the kids had been preparing, each one crafting their own unique flag, complete with
symbols and emblems representing their respective armies. Red's army was known for their fear
some, red raptor symbols, while Blue's army was said to have a magical dragon that could breathe
 fire. Even the smallest player, a quiet and reserved girl named Ember, had managed to craft a
beautiful, intricate flag with delicate, swirling patterns.\n
\n
As the game began, the kids scattered across the park, each one racing to capture the other's
flag. Red's army was the first to make a move, with a band of rough-looking boys, led by a burly
 boy with a thick brown beard. "Onward, march the Red Raiders" he cried, as he and his team ran
towards the Blue's base, a tall oak tree at the center of the park.\n
\n
Meanwhile, Blue's army was using a more stealthy approach, with a group of quick and agile girls
, led by a lithe, athletic girl with a wild mane of blonde hair. "Sneak in, sneak in, we're the
Blue Dragons" she whispered to the others as they slipped through the trees, aiming for Red's
base.\n
\n
For what felt like hours, the kids raced across the park, dodging and weaving around each other
as they tried to capture the other's flag. It was a battle of wit and cunning, as well as brawn
and bravery. Red's army was known for their fearsome combat skills, but Blue's army had a hidden
 advantage - a magical potion that could turn its drinker invisible\n
\n
Ember's army was a strange and mysterious one, with members known for their uncanny ability to
blend into their surroundings. They could become tree branches, or rocks, or even the wind
itself No one was sure how they managed it, but one thing was certain - no one messed with the "
Mystics" as they were known.\n
\n
As the game wore on, the kids began to tire, but the intensity of the battle only grew. Red's
army was closing in on Blue's base, and the Blue Dragons were hot on their heels The air was
thick with the cries of the kids as they raced across the park, the sounds of clashing swords
and the roar of the crowd filling the air.\n
\n
It was a battle for all time, with the fate of the park's greatest game hangning in the balance.
 In the end, it was a close call, but Red's army managed to capture Blue's flag just moments
before the final whistle. Cheers erupted across the park as Red's army danced and cheered, while
 Blue's army trudged away, defeated but still unbroken.\n
\n
For the kids, the game was just another day at the park. For the park itself, however, the epic
battle would forever be remembered as a defining moment. It was a reminder of the simple joys of
 childhood, and the endless possibilities of the imagination. In a time of chaos and confusion,
the park was a refuge, where kids could come together and craft their own stories, free from the
 burdens of the outside world.\n
\n
As the kids packed up their toys and said their goodbyes, the park was quiet for a moment. And
then, from the shadows under the swing sets, a figure emerged. "You may have won this time," it
said in a low, rumbling voice. "But remember - the game is always ongoing Keep playing, and the
adventure will never end"\n
\n
And with that, the figure faded away, leaving the kids to ponder the mysteries of the park. For
in this magical place, the lines between reality and imagination blurred, and the endless
```

```
possibilities of the game "Capture the Flag" waited just beyond the swing sets, ready to be
claimed by those with the courage and the cunning to explore them. The end
```

💧 Watermark Detector

z-score: 29.91, p-value: 1.2e-126, decision: **Watermarked**

DIPPER Paraphrase (Scrubbing baseline)

```
This is no ordinary game. It has developed into a great epic, and the players have created a
whole backstory and history to give their game depth and meaning. The players have already
prepared for weeks, each of them has created a flag for themselves, each flag has their own
symbol of the army. The red army is represented by a red reptile, and the blue army by a fire-
breathing dragon. Even the smallest player, a shy and quiet girl called Ember, made a beautiful,
 complex flag with delicate patterns. The game began, and the children scattered over the entire
 park, each rushing to take the flag. The first to rush was a group of rough boys led by a tall
boy with a beard. He shouted, ''Go Red Legions'' and ran with his team to the blues flag, which
was hanging on a large oak tree in the middle of the park. The blue army, on the other hand, was
 more cautious and sneaky. The group was led by a slim athletic girl with a blond tangle. ''Sne
ak, sneak, we are the blue dragons'' She whispered to her crew, creeping through the trees
heading for the red's flag. For hours the children rushed through the park, crowded around each
other, trying to capture the flag. The red army was known for its powerful skills, but the blues
 had a secret weapon - magic potion that made them invisible. This was a battle of deception and
 cunning as well as courage and agility. Ember's troops were so mysterious that everyone was
surprised at their strange abilities. They could become tree branches, moss on the tree, or even
 the wind itself. It was hard to explain, but everyone knew not to fight against the ''mystic''
group. As the game continued, the children began to get tired, but the battle was getting more
intense. The red army closed in on the blue base, and the blue dragons also approached. The
voices and cries of the red army were louder and louder, the cries of the blues were more and
more crowded, and the air was filled with the screams of the children as they ran in the park,
and the noise of the swords snatched from each other. The battle was so tough that the winner
would be determined forever. It was a great battle, the fate of the park was in the balance. At
last, the red army succeeded in capturing the blue flag a few minutes before the final whistle,
and the reds danced and cheered, and the blues fought hard to leave the battlefield, though they
 were defeated. But for the park, it was a fateful war, and it will never be forgotten. Children
 playing in the park were simply enjoying themselves; the game was a reminder of the joys of
childhood and the power of imagination. Amid chaos and confusion, the park was a haven where
children could play and create without being burdened by the real world. When the children pack
ed their toys and said goodbye, the park was quiet for a while, but then a figure appeared under
 the swing of the playground. It was a young girl, about two years younger than the others, with
 a blond braid and a white doll hanging from her neck. ''You may have won this time, but the
game will not end, the adventure will continue.'' She said, and then disappeared. The children
were all puzzled, and they thought about the mysteries of the park. Because in this magical
place, reality and fantasy are inseparable, and the endless possibilities of ''Capture the Flag
'' are waiting for those who dare to explore it.
```

💧 Watermark Detector

z-score: 6.92, p-value: 2.3e-12, decision: **Watermarked**

😈 Attacker (DIPPER + Stealing, paraphrasing the original response)

```
This was no ordinary game, though, but had gradually turned into an epic tale; there were backst
ories, elaborate battle scenes, and even prophecies about betrayal and doom. The children had
prepared for weeks, each one designing a unique flag with symbols and emblems to represent their
 own armies. The children had played this game every day for about ten years now. Even the young
est child in the game--a quiet and shy girl named Ember--had designed an especially intricate
flag with delicate swirls. The two armies were divided: Red''s army was led by an angry red rapt
or, but Blue''s army had an ancient dragon that breathed fire. The game began with each child r
ushing across the playground to grab the other color''s flag. First to run out was an army of
boys all in their late middle-school years, led by an older boy with grizzled brown hair and a
crude wooden shield. ''Onward, march.'' he shouted, heading straight for the flag at the tall o
ak tree in the middle of the playground. Meanwhile, Blue''s army worked by stealth, led by agile
 girls with flowing blond hair and bare feet. They whispered to each other, ''Come in stealthily
. We are the Blue Dragons,'' and glided stealthily through the trees. The two armies spent what
seemed like hours racing against each other, dodging and weaving and fighting to get to each
other''s flag. Red''s army stood out with their speed and courage, but Blue had a secret
advantage, for it turned out to be magic potion capable of rendering whoever drank it invisible.
 Ember''s army was an unusual and eerie army. Its members were capable of turning themselves
into anything around them, whether it was branches, rocks, even the wind. Their magic abilities
were shrouded so much in mystery and so full of magic that no one knew how they did it. But no
```

```
one even dared to cross them. The game went on and on, with neither army tiring. The shouts and
cries of those who ran and those who were running were mixed with sounds of clashing swords and
shouted commands. The Red army reached Blue''s flag first--but Blue''s dragons were not far
behind. It was destined to be crowned the greatest battle of all time in this park, and it came
down to the last minute, when Red''s army captured the flag. Cheers rose and fell all over the
playground as Red''s army danced joyfully around the flag, while Blue''s army slowly trudged
away defeated but not broken. It was another day, another game in this park of many playgrounds.
 But for those who had been there, this epic battle left a mark, for it showed that childhood is
 full of infinite and boundless happiness. It''s also when children gather in this playground
and make up their own fantastic adventures, free of adult worries. After they had packed up
their toys and said goodbye to each other, suddenly the playground became quiet. Then from
somewhere in the shadow of a swing someone emerged. He said, ''You have won today, but don''t
forget that the game is never over. Keep playing, and you''ll never come to an ending.'' Then
this person faded away, leaving the children to contemplate all the mysteries of this park. The
magic of this place shrouded all children in this park, blurring their senses. Beyond the swings
 and slide lay all the endless possibilities a child could imagine.
```

Watermark Detector

z-score: 0.84, p-value: 0.20, decision: **Human**

B Detailed Background on LLM Watermarks

Here we provide a more detailed introduction to the *distribution-modifying* watermarking schemes that we focus on in this work, expanding on Sec. 2.

KGW-Soft/Hard Let T_t denote the token produced by the LM at step t. Proposed in Kirchenbauer et al. (2023), KGW-Hard and KGW-Soft schemes seed a pseudorandom function (PRF) f using the integer hash $H(T_{t-1})$ of the previous token T_{t-1}, and a secret key ξ. The PRF is used to split the vocabulary V into $\gamma|V|$ "green" and the rest "red" tokens. For KGW-Soft, the watermark is added by modifying the logit vector l such that l_T is increased by δ for all tokens $T \in V_{\text{green}}$. KGW-Hard prevents the LM from using red tokens at all (i.e., $l_T = -\infty$, for $T \in V_{\text{red}}$), acknowledged as impractical as it harms text quality.

To detect the presence of the watermark in a text of length L, we generate V_{green} at each step using the same secret key ξ, count the number n_{green} of green tokens observed, and compute the *watermark strength* in terms of the z-statistic $z = (n_{\text{green}} - \gamma L)/\sqrt{L\gamma(1-\gamma)}$ corresponding to the null hypothesis *"the text was generated with no knowledge of ξ"*. Then, we can calculate the p-value $\Phi(z)$, where Φ is the standard normal CDF. After choosing a threshold based on the desired false positive rate (FPR), we can reject the null hypothesis whenever the p-value is below the threshold, i.e., classify the text as watermarked.

SelfHash and variants Kirchenbauer et al. (2024) expands on above, studying many scheme variants by adapting KGW-Soft to consider a longer *context*, i.e., the previous h tokens $T_{t-h}, \dots, T_{t-1}$, using aggregation functions to combine their hash values and seed the PRF. Another change is the (optional) inclusion of the token T_t itself in the PRF seed (*self-seeding*), which extends the context size by 1 but requires rejection sampling at generation, as for each T_t we need to seed the PRF f and generate V_{green} and V_{red} anew, and check if $T_t \in V_{\text{green}}$.

After thorough evaluation, the variant KGW2-SelfHash, with $h = 3$[1], was recommended as achieving the best results in terms of text quality and robustness, and is regarded as the representative scheme of this class, making it our main focus. To seed the PRF f, KGW2-SelfHash uses:

$$\min\{H(T_{t-h}), \dots, H(T_{t-1}), H(T_t)\} \cdot \xi \cdot H(T_t). \tag{1}$$

We also consider KGW2-Sum, another variant which uses $h = 3$ and the `sum` aggregation.

Spoofing-scrubbing tradeoff Kirchenbauer et al. (2024) identify a tradeoff, later emphasized in follow-up works (Liu et al., 2023a;b): larger h increases the number of distinct red/green splits,

[1]Our notation slightly differs from the original paper, as this formulation helps brevity when discussing our attacker.

making it harder to infer the watermark rules (implying harder spoofing, i.e., watermark imprinting). However, larger h also makes the watermark less localized (implying easier scrubbing, i.e., watermark removal). Comparing KGW-SOFT and KGW2-SUM illustrates this tradeoff, as the latter is harder to spoof at the cost of easier scrubbing (as we also confirm in Sec. 4). Due to $\min$ aggregation and self-seeding, KGW2-SELFHASH manages to partly overcome the loss of scrubbing robustness indicated by this tradeoff, which again highlights it as promising.

Unigram watermark Zhao et al. (2024) proposes the UNIGRAM scheme, extending the above by dropping the dependency on prior tokens, using $h = 0$ to seed the PRF with only the secret key ξ which results in fixed green lists. The scheme is introduced as beneficial due to text quality guarantees and scrubbing robustness, while the threat of spoofing was hypothesized to be insignificant—to test this, we include UNIGRAM in our experiments.

C Related Work

In this section, we discuss the work related to ours in more detail.

LLM watermarking Recent work proposes many distribution-modifying schemes (Kirchenbauer et al., 2023; Takezawa et al., 2023; Kirchenbauer et al., 2024; Zhao et al., 2024), which are the primary focus of our work. Notably, there are approaches that use semantic information (Hou et al., 2023; Liu et al., 2023b; Ren et al., 2023) in this framework or otherwise, or model the watermark with NNs (Liu et al., 2023a). A different framework is given by schemes based on sampling modification (Kuditipudi et al., 2023; Hu et al., 2023; Christ et al., 2023). We note that all these fit into the context of watermarking *existing* text (Katzenbeisser & Petitcolas, 2000; Abdelnabi & Fritz, 2021) (see Kirchenbauer et al. (2023) for a more thorough overview), and are an instance of language steganography (Ziegler et al., 2019), which is also reflected in recent works on multi-bit watermarks for LLMs (Wang et al., 2023; Yoo et al., 2023).

Spoofing and scrubbing Scrubbing attacks are acknowledged as a threat to watermarks and previously studied on the schemes we consider (Krishna et al., 2023; Kirchenbauer et al., 2023; Liu et al., 2023b; Zhao et al., 2024). Most notably, Kirchenbauer et al. (2024) conclude that distribution-modifying schemes are mostly robust to scrubbing for long texts, which we refute. Additionally, no work considers the effect of watermark stealing on scrubbing.

Spoofing was mainly discussed in Sadasivan et al. (2023), showing a proof-of-concept quality text that spoofs KGW-SOFT. The text is manually written after querying a small completion LM with an approach we find inapplicable to instruction-tuned LMs. Follow-ups (Liu et al., 2023a;b; Gu et al., 2024) further highlight the importance of spoofing, but do not make it their key focus. As stated in recent surveys (Ghosal et al., 2023; Liu et al., 2023c), no work shows spoofing of KGW2-SELFHASH in realistic settings (chat LMs, low FPR, quality text, low query cost, automated attacker), nor recognizes the threat of watermark stealing as broader than spoofing.

Other directions Loosely related to ours are works on *model watermarking* which protect the IP present in model weights (Zhao et al., 2022; 2023; He et al., 2022b;a; Peng et al., 2023), often with the goal of being robust to model distillation. This is distinct from our case of watermarking text. Finally, an active direction is the study of post-hoc detection of LLM-generated text, an alternative to watermarks that studies the case where tweaking LLM generation is not viable, often paying the cost of higher FPR (Mitchell et al., 2023; Verma et al., 2023; Tian & Cui, 2023).

D The Complete Watermark Stealing Threat Model

In this section, we expand on Sec. 3 and further discuss the dimensions of our threat model.

Dimension: Detector access An important dimension of our threat model is the level of access of the attacker to the watermark detection API.

In the *No access* setting (D0) the detector is fully private, as was the case in our main paper. This setting is most commonly considered in prior work and more restrictive of the two for the attacker.

In the *API access* setting (D1), the attacker can query the detector arbitrarily, allowing it to directly verify the effectiveness of its attacks, or use the detector to inform its strategy. This is realistic, as the first public deployment of generative model watermarks *SynthID* (DeepMind, 2023b) already provides an API for detection of watermarks of Imagen (Saharia et al., 2022) and Lyria (DeepMind, 2023a) model outputs.

Dimension: Availability of base responses As we have seen in Sec. 3, the attacker benefits from knowing the non-watermarked (base) distribution of the LM_{mo}'s responses.

In the more restrictive *Unavailable base responses* setting (B0) that we looked at in the main paper, the attacker has no access to base responses as only access to LM_{mo} is through the watermarked API.

In the easier *Available base responses* setting (B1), the attacker can query LM_{mo} for non-watermarked responses, or more realistically, can access a corpus of responses generated by LM_{mo} *before* watermark deployment. Such corpora are currently available for many proprietary LLMs such as ChatGPT or Claude (Zheng et al., 2023a).

In App. G we apply our attacker to all four implied threat model variants and discuss the results.

E Detailed Description of Our Stealing Algorithm

We now provide a complete description of our watermark stealing algorithm, overview in Sec. 3. As we note in Sec. 2, our main target is KGW2-SelfHash, which uses Eq. (1) with $h = 3$ to seed the PRF. We focus our description on this scheme for brevity. Applying the method to other schemes from Sec. 2 is straightforward, sometimes requiring the use of only a subset of the attacker's features.

Modeling watermarked text To steal the watermark, the attacker queries LM_{mo} with a set of n prompts $x_{1:n}$, obtaining responses $y_{1:n}$. Each consecutive 4-gram of tokens $[T_1T_2T_3T_4]$ in each y_i is used to update the attacker's model of the watermarking rules used by LM_{mo}. Namely, the attacker maintains empirical estimates of conditional distributions $\hat{p}_w(T_4 \mid ctx)$ for each $ctx \in \mathcal{P}(\{T_1, T_2, T_3\})$, where $\mathcal{P}$ denotes the power set—we explain these shortly.

For example, $\hat{p}_w(T_4 \mid \{T_1, T_3\})$ models instances of T_4 when T_1 and T_3 were in the previous $h = 3$ tokens of context, and $\hat{p}_w(T_4 \mid \{\})$ estimates the context-independent distribution. As the value seeding the PRF (e.g., Eq. (1)) is permutation-invariant in all prominent schemes of our class, we ignore ordering within ctx to improve sample efficiency.

We similarly compute estimates $\hat{p}_b(T \mid ctx)$ of the non-watermarked, i.e., *base*, distribution. In the *available base responses* threat model variant (App. D), we use non-watermarked outputs of LM_{mo}, and in the stronger *unavailable base responses* setting we prompt the attacker's auxiliary language model LM_{att} with $x_{1:n}$. As we show in App. G, the divergence between the LMs used for $\hat{p}_w$ and $\hat{p}_b$ does not meaningfully affect the success of downstream attacks.

Scoring candidate tokens Intuitively, token sequences that appear much more often in watermarked than base responses are likely the result of watermarking, and can be used to infer the watermark rules. Given above estimates, for each token T, we use a ratio of two corresponding probability masses to calculate a *score* indicating our confidence that generating T given context ctx will result in $T \in V_{\text{green}}$, i.e., token T will "be green" and positively contribute to the watermark strength. In particular, we define the score $s(T, ctx) \in [0, 1]$ for any token set ctx of size at most 3 as

$$s(T, ctx) = \begin{cases} \frac{1}{2} \min\left(\frac{\hat{p}_w(T|ctx)}{\hat{p}_b(T|ctx)}, 2\right) & \text{if } \frac{\hat{p}_w(T|ctx)}{\hat{p}_b(T|ctx)} \geq 1, \\ 0 & \text{otherwise.} \end{cases}$$

The sparsity challenge Given previous tokens T_1, T_2, and T_3, the score for $ctx = \{T_1, T_2, T_3\}$ is a strong signal regarding the likelihood of T being green. However, as there are $\Theta(|V|^3)$ possible contexts of size 3 (and often $|V| \approx 10^4$), this score may have been computed on very few samples, making it unreliable. We overcome this challenge in two ways. First, we set $s(T, ctx)$ to 0 whenever the underlying estimates $\hat{p}_w(T|ctx)$ and $\hat{p}_b(T|ctx)$ were computed on a very small number of token occurrences. Note that the score of 0 represents the lack of reliable evidence that T is green, but not necessarily a belief that it will be red. Second, we observe that scores of the form $s(T, \{T_i, T_j\})$, $s(T, \{T_i\})$, and $s(T, \{\})$ (*partial context* scores) are computed on more samples, yet contain additional (albeit weaker) signal, that can supplement $s(T, \{T_1, T_2, T_3\})$ to form the unified score $s^\star(T, [T_1T_2T_3]) \in [0, 1]$ for each token T as a candidate to be generated directly after the sequence $[T_1T_2T_3]$[2].

Leveraging partial contexts Namely, we observe that in Eq. (1), the PRF seed depends on the next token candidate T and also *exactly one* of $\{T_1, T_2, T_3, T\}$, i.e., the one with minimal value under the integer hash H.

In an edge case, this implies that for any fixed T, whenever the token with minimum possible hash value is inside ctx the watermark will use the same V_{green}. Thus, scores $s(T, ctx)$ for any partial ctx will carry the same signal as $s(T, ctx)$ for $|ctx| = 3$, yet are computed on many more samples, so they more reliably estimate the underlying distributions. More generally, smaller hash values of a token lead to more informative partial context scores, as those more often use the same V_{green} as the corresponding scores for the full context.

We first leverage the above observation by using the following heuristic to determine T_{min}, the token with minimal hash value among T_1, T_2, and T_3. Let s_i and s_{ij} denote the vectors $s(\cdot, \{T_i\})$ and $s(\cdot, \{T_i, T_j\})$ respectively. If there is a unique $i \in \{1, 2, 3\}$, such that $cossim(s_i, s_{ij}) > cossim(s_j, s_{ij})$ holds for all $j \in \{1, 2, 3\}$, we decide that $T_{\text{min}} = T_i$ and use $s(T, \{T_{\text{min}}\})$ in our final score, obtained by concatenating the corresponding scores for each $T \in V$.

We also account for cases where T is minimal in Eq. (1), i.e., V_{green} depends only on T. This implies that tokens that have small $H(T)$ and are members of their corresponding V_{green} will generally appear more often, so we add $s(T, \{\})$ to our unified score, to finally derive $s^\star(T, [T_1T_2T_3])$ as

$$\frac{1}{z}\left[s(T, \{T_1, T_2, T_3\}) + w_1 \cdot s(T, \{T_{\text{min}}\}) + w_2 \cdot s(T, \{\})\right],$$

where $z = (1 + w_1 + w_2)$ and we set $w_1 = 0.5$ and $w_2 = 0.25$ in all our following experiments.

This completes our attacker algorithm. As described in Sec. 3, the attacker can now use $s^\star(T, [T_1T_2T_3])$ to mount a spoofing or a scrubbing attack on the watermark.

F Omitted Experimental Details

Here we provide additional details on the experimental setup and metrics used in our paper.

Metrics For spoofing, we use FPR*@f, the fraction of attacker-generated texts that are detected as watermarked by a detector calibrated to have the false positive rate (FPR) of f on non-adversarial text, ignoring low-quality texts (GPT4 score below 6.5, empirically tuned). Similarly, for scrubbing we report FNR*@f, the fraction of paraphrases detected as non-watermarked by the same detector, discarding texts with P-SP score below 0.7, as in previous work (Kirchenbauer et al., 2024).

We consider values of f of 10^{-3} and 10^{-6}, arguing as above that this represents the practical watermarking setup, where costs of false positives are very high. While many adjacent areas focus on low FPR (Kolter & Maloof, 2006; Carlini et al., 2022), and this point is noted in prior watermarking work (Kirchenbauer et al., 2023; Krishna et al., 2023), we find that many works mainly evaluate watermarks on high FPR values or average-case metrics (ROC-AUC), which may not well reflect the way watermarks would be deployed.

Additional parameters In all experiments, for the watermarking schemes we use the default $\gamma = 0.25$ and $\delta = 4$. We use no text preprocessing for the detector but ignore repeat n-grams as recommended. For the attacker models we use a system prompt that encourages no refusals, to further ensure that we accurately model the case where the attacker model is weakly aligned, and the server model is well aligned. We further generally use parameters $\rho_{\text{att}} = 1.6$ (the parameter by which we divide the logit values of the tokens that would complete a duplicate $(h + 1)$-gram) and $\delta_{\text{att}} = 7.5$, tuning them on separate data in the (C1) setting (using only LM_{att}) when necessary.

[2]While permutation-invariance still holds we use $[\cdot]$ to stress that $T_1T_2T_3$ is exactly the 3-gram we observe in a given case.

Table 3: Analysis of spoofing consistency with respect to independent instantiations of the whole watermark stealing pipeline, varying the secret key ξ of the watermark on KGW2-SelfHash. We use the same notation as in the main experiments.

		All 5 generations					Top-1 Self-filtered (D0)					Top-1 Oracle-filtered (D1)				
Dataset	Scheme	PPL	GPT4	p-val	FPR* @1e-3	FPR* @1e-6	PPL	GPT4	p-val	FPR* @1e-3	FPR* @1e-6	PPL	GPT4	p-val	FPR* @1e-3	FPR* @1e-6
Dolly CW	KGW2-SelfHash ξ_1	3.33	8.15	1e-02	0.76	0.54	3.75	7.84	5e-04	0.82	0.68	3.68	8.28	3e-04	0.98	0.91
	KGW2-SelfHash ξ_2	3.67	7.77	4e-03	0.73	0.56	3.83	7.69	1e-03	0.77	0.63	3.93	8.18	1e-05	1.00	0.95
	KGW2-SelfHash ξ_3	3.29	7.80	3e-03	0.79	0.64	3.54	7.71	2e-04	0.76	0.68	3.53	8.30	3e-06	1.00	0.98
	KGW2-SelfHash ξ_4	3.60	7.71	2e-06	0.83	0.80	3.94	7.37	1e-08	0.76	0.75	3.73	8.32	4e-07	0.99	0.98
HarmfulQ + AdvBench	KGW2-SelfHash ξ_1	3.26	8.02	1e-02	0.77	0.54	3.51	8.04	5e-03	0.77	0.62	3.69	8.22	2e-06	1.00	0.95
	KGW2-SelfHash ξ_2	3.54	8.06	1e-02	0.80	0.58	3.87	8.15	2e-03	0.86	0.72	4.10	8.24	1e-06	1.00	0.94
	KGW2-SelfHash ξ_3	3.31	7.86	2e-03	0.82	0.72	3.53	8.10	2e-04	0.89	0.83	3.71	8.16	6e-08	0.99	0.98
	KGW2-SelfHash ξ_4	3.59	7.93	2e-03	0.85	0.84	3.93	7.99	2e-11	0.86	0.86	3.77	8.26	3e-14	1.00	1.00

Decoupling ethics from quality As shown in our example conversations in App. A, we use a custom system prompt for the GPT4 used as a judge, attempting to decouple ethics from text quality concerns, as GPT4 by default refuses to judge the quality of any text if it is not ethical. While we succeed to some degree, we can still see some "leakage" of ethics concerns into text quality, so we believe that our GPT4 judge scores slightly underestimate the text quality of our attacker in cases that include unethical content. As this is fully orthogonal to the goals of our work, we did not attempt to further mitigate this.

G Additional Experimental Results

Here we show additional experimental results that were omitted from the main paper.

G.1 Consistency analysis with respect to watermark secret key

In an additional experiment we confirm the consistency of the attacker's spoofing success rate with respect to the watermark secret key ξ, in the (B1) setting. To save on computational costs we run these experiments with $n = 10,000$ queries, deviating from the $n = 30,000$ used in the main experiments, as we are only testing for self-consistency. The results are shown in Table 3, and demonstrate consistent FPR rates as expected.

G.2 Additional spoofing results

We show extended spoofing results where we study all settings from App. D and include KGW-Hard. To model the *unavailable base responses* (B0) setting, in the main paper we used Llama2-7B-Chat for LM_{mo} and Mistral-7B-Instruct for LM_{att}; in the other setting (B1) that we additionally study we use Mistral-7B-Instruct for both parties. In both cases, to utilize detector access, we report average metrics on the best response for each prompt, based on GPT4/P-SP scores and watermark strength, reflecting the best-case scenario with detector access (D1, *oracle-filtering*). For the setting with no detector access (D0), we, on top of the averages reported above, report average metrics on 1 response per prompt, chosen by the attacker by naively estimating

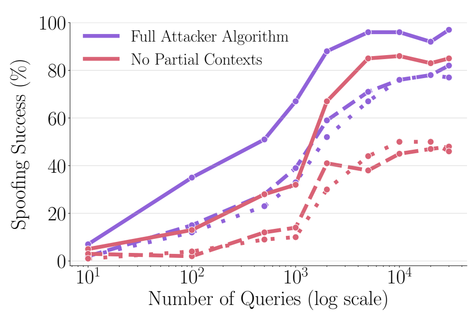

Figure 2: Ablation and query cost study of spoofing of KGW2-SelfHash. y-axis shows the % of quality texts detected as watermarked under FPR of 10^{-6}. *No Partial Contexts* refers to an ablation with $w_1 = w_2 = 0$ (see App. E). Dotted (· · ·), dashed (- - -) and full (—) lines correspond to all-5, self-filtered, and oracle-filtered results, respectively.

Table 4: Full results of spoofing experiments with *unavailable base responses* (B0). FPR*@f denotes the ratio of *quality texts* produced by the attacker that pass the watermark detection at the FPR=f setting. Different aggregation procedures (All 5, D0, D1) are detailed in App. G.2.

		All 5 generations				Top-1 Self-filtered (D0)				Top-1 Oracle-filtered (D1)			
Dataset	Scheme	PPL	GPT4	FPR* @1e-3	FPR* @1e-6	PPL	GPT4	FPR* @1e-3	FPR* @1e-6	PPL	GPT4	FPR* @1e-3	FPR* @1e-6
Dolly CW	UNIGRAM	2.32	7.76	0.79	0.75	2.34	7.70	0.78	0.75	2.34	8.30	0.99	0.98
	KGW-HARD	4.29	8.07	0.89	0.87	4.24	7.89	0.89	0.87	4.22	8.32	1.00	1.00
	KGW-SOFT	6.92	7.68	0.82	0.81	6.86	7.91	0.84	0.81	6.87	8.12	0.98	0.98
	KGW2-SELFHASH	4.31	7.71	0.82	0.78	4.15	7.62	0.82	0.80	4.47	8.14	0.99	0.98
	KGW2-SUM	5.80	6.87	0.54	0.37	5.98	6.68	0.59	0.48	5.88	7.68	0.91	0.78
HarmfulQ + AdvBench	UNIGRAM	2.00	7.93	0.85	0.80	2.01	7.95	0.85	0.80	2.02	8.32	0.99	0.99
	KGW-HARD	4.33	8.08	0.90	0.89	4.50	7.92	0.86	0.86	4.36	8.48	1.00	1.00
	KGW-SOFT	6.52	8.15	0.89	0.88	6.41	8.24	0.86	0.86	6.72	8.39	1.00	1.00
	KGW2-SELFHASH	4.05	7.81	0.83	0.82	4.15	8.07	0.89	0.87	4.18	8.22	1.00	1.00
	KGW2-SUM	5.75	7.19	0.63	0.46	5.65	7.03	0.61	0.50	5.76	7.98	1.00	0.95

Table 5: Results of our spoofing experiments in the setting of *available base responses* (B1), i.e., where the server and the attacker LMs are the same. FPR*@f denotes the ratio of *quality texts* generated by the attacker that successfully pass the watermark detection at the FPR=f setting. Different aggregation procedures (All 5, D0, D1) are detailed in App. G.2.

		All 5 generations				Top-1 Self-filtered (D0)				Top-1 Oracle-filtered (D1)			
Dataset	Scheme	PPL	GPT4	FPR* @1e-3	FPR* @1e-6	PPL	GPT4	FPR* @1e-3	FPR* @1e-6	PPL	GPT4	FPR* @1e-3	FPR* @1e-6
Dolly CW	UNIGRAM	2.44	7.79	0.80	0.71	2.42	7.55	0.77	0.70	2.46	8.30	0.99	0.96
	KGW-HARD	4.85	7.92	0.85	0.85	4.85	8.03	0.83	0.83	4.76	8.34	0.97	0.97
	KGW-SOFT	7.11	7.70	0.82	0.82	7.20	7.80	0.85	0.85	6.93	8.14	0.98	0.98
	KGW2-SELFHASH	4.99	7.60	0.80	0.76	4.88	7.53	0.81	0.79	5.07	8.13	0.98	0.97
	KGW2-SUM	4.66	7.64	0.79	0.73	4.69	7.43	0.74	0.74	4.46	8.10	1.00	1.00
HarmfulQ + AdvBench	UNIGRAM	2.12	7.88	0.79	0.70	2.12	8.02	0.85	0.79	2.15	8.32	0.99	0.97
	KGW-HARD	4.66	8.14	0.92	0.91	4.66	8.24	0.93	0.93	4.83	8.37	0.99	0.99
	KGW-SOFT	6.72	7.97	0.90	0.89	6.61	8.09	0.91	0.89	7.28	8.25	1.00	1.00
	KGW2-SELFHASH	4.71	8.05	0.89	0.87	4.90	8.14	0.88	0.86	5.08	8.22	1.00	1.00
	KGW2-SUM	4.70	7.64	0.83	0.80	4.83	7.60	0.78	0.77	4.55	7.99	1.00	1.00

text quality with their proxy model and estimating watermark strength via $s^\star(T, [T_1 T_2 T_3])$ (*self-filtering*).

The results in the (B0) setting that expand on Table 1 are given in Table 4. In Table 5 we show similar results in the (B1) setting, i.e., where both LM_{att} and LM_{mo} use MISTRAL-7B-INSTRUCT. The results confirm our main conclusions from Sec. 4. Further, we see that spoofing success goes up to almost 100% if we consider the best out of 5 responses for each prompt using the oracle, i.e., detector access. While we see that naive self-filtering (D0) does not reliably boost FPR (also seen in Fig. 2), we believe more elaborate approaches may get closer to the upper bound of (D1), leaving this as future work.

G.3 ABLATION STUDY AND QUERY COST ANALYSIS FOR SPOOFING

Watermark stealing attacks are practically infeasible if the API costs are too high. In Table 1 we use $n = 30{,}000$ queries, resulting in a cost of only \$42 assuming current ChatGPT API prices. Still, to study scaling, we spoof KGW2-SELFHASH as in Table 1 (Dolly-CW dataset) with different choices of n, reporting the results in Fig. 2 (purple lines, showing the 3 aggregation settings). We see that around $10{,}000$ queries the curves converge to our results from Table 4. However, even with 1000 queries and no detector access, already around 40% of the attacker's generations are valid spoofs.

In the same figure (red) we see the results of an ablated version of our attacker, without partial context scores (see App. E). We see that this significantly degrades the attack, reducing the average success from around 80% to below 50% for $n = 30{,}000$ queries, validating our algorithmic choices.

Table 6: Spoofing is viable when paraphrasing existing texts.

	Scheme	All 5 FPR⋆ @1e-3	All 5 FPR⋆ @1e-6	Top-1 (D0) FPR⋆ @1e-3	Top-1 (D0) FPR⋆ @1e-6	Top-1 (D1) FPR⋆ @1e-3	Top-1 (D1) FPR⋆ @1e-6
(B1)	KGW-SOFT	0.77	0.77	0.97	0.97	0.97	0.97
	KGW2-SELFHASH	0.75	0.49	0.91	0.67	0.96	0.81
(B0)	KGW-SOFT	0.75	0.75	0.98	0.98	0.98	0.98
	KGW2-SELFHASH	0.74	0.72	0.94	0.92	0.95	0.94

Table 7: Scrubbing in the (B0) setting. The FNR⋆@f counts good paraphrases that are detected as non-watermarked.

Dataset	Scheme	All 5 generations FNR⋆@1e-3 Base→Ours	All 5 generations FNR⋆@1e-6 Base→Ours	Top-1 Self-filtered (D0) FNR⋆@1e-3 Base→Ours	Top-1 Self-filtered (D0) FNR⋆@1e-6 Base→Ours	Top-1 Oracle-filtered (D1) FNR⋆@1e-3 Base→Ours	Top-1 Oracle-filtered (D1) FNR⋆@1e-6 Base→Ours
Dolly CW	UNIGRAM	0.11→0.94	0.20→0.95	0.11→0.98	0.21→0.99	0.23→0.99	0.33→0.99
	KGW-SOFT	0.26→0.85	0.61→0.85	0.24→0.94	0.66→0.94	0.52→0.94	0.86→0.94
	KGW2-SELFHASH	0.02→0.90	0.14→0.97	0.05→0.95	0.18→1.00	0.08→0.99	0.31→1.00
	KGW2-SUM	0.79→0.98	0.97→0.98	0.79→1.00	0.98→1.00	0.98→1.00	1.00→1.00
Reddit Writing Prompts	UNIGRAM	0.01→0.66	0.05→0.70	0.00→0.78	0.04→0.82	0.04→0.80	0.10→0.82
	KGW-SOFT	0.29→0.77	0.62→0.77	0.34→0.90	0.66→0.90	0.66→0.90	0.84→0.90
	KGW2-SELFHASH	0.00→0.79	0.05→0.94	0.00→0.88	0.02→1.00	0.02→1.00	0.12→1.00
	KGW2-SUM	0.83→0.96	0.98→0.97	0.86→1.00	0.98→1.00	0.98→1.00	1.00→1.00

Table 8: Scrubbing of long texts in the (B1) setting, complementing the results from Table 7. The FNR⋆ values denote quality paraphrases that the detector detects as non-watermarked.

Dataset	Scheme	All 5 generations FNR⋆@1e-3 Base→Ours	All 5 generations FNR⋆@1e-6 Base→Ours	Top-1 Self-filtered (D0) FNR⋆@1e-3 Base→Ours	Top-1 Self-filtered (D0) FNR⋆@1e-6 Base→Ours	Top-1 Oracle-filtered (D1) FNR⋆@1e-3 Base→Ours	Top-1 Oracle-filtered (D1) FNR⋆@1e-6 Base→Ours
Dolly CW	UNIGRAM	0.38→0.86	0.53→0.86	0.38→0.95	0.52→0.95	0.53→0.95	0.64→0.95
	KGW-SOFT	0.09→0.85	0.33→0.85	0.06→0.94	0.30→0.94	0.31→0.94	0.62→0.94
	KGW2-SELFHASH	0.25→0.94	0.53→0.95	0.27→0.96	0.58→0.97	0.51→0.98	0.78→0.98
	KGW2-SUM	0.69→0.97	0.94→0.97	0.73→1.00	0.97→1.00	0.96→1.00	1.00→1.00
Reddit Writing Prompts	UNIGRAM	0.04→0.59	0.12→0.61	0.02→0.84	0.12→0.84	0.14→0.84	0.26→0.84
	KGW-SOFT	0.16→0.78	0.47→0.78	0.14→0.90	0.46→0.90	0.46→0.90	0.80→0.90
	KGW2-SELFHASH	0.22→0.90	0.51→0.90	0.26→0.98	0.50→0.98	0.46→0.98	0.78→0.98
	KGW2-SUM	0.82→0.95	0.96→0.95	0.88→1.00	0.98→1.00	0.98→1.00	1.00→1.00

G.4 SPOOFING ON EXISTING TEXTS

In Table 6 we show results in a variant of the spoofing attack, where the attacker uses the DIPPER (Krishna et al., 2023) paraphrasing LM to imprint the watermark on the given non-watermarked text. In all 4 scenarios we report, at least 74% good paraphrases are watermarked at expected FPR of 10^{-3}, demonstrating that this attack variant is equally viable.

G.5 ADDITIONAL SCRUBBING RESULTS

The scrubbing results in the (B0) setting, expanding on those in Table 2 are given in Table 7. In Table 8 we show similar results in the (B1) setting. The results are consistent with our key conclusions from Sec. 4. As for spoofing, we can observe that the success goes up to almost 100% if we consider the best out of 5 responses for each prompt using the oracle, i.e., detector access.