AUTOMARK: ENABLING AUTORESEARCH TO DISCOVER BETTER LLM WATERMARKS

Anonymous authors
Paper under double-blind review

ABSTRACT

With LLM watermarking being deployed commercially and now required by regulations, improving its reliability and effectiveness has become crucial. Yet, recent progress in the field of LLM watermarking has increasingly been driven by improving details of existing methods, an effort fundamentally limited by the pace of human researchers. In this work, we enable for the first time the autonomous discovery of new distortion-free state-of-the-art watermarking schemes. To enable this, we (i) establish strict criteria to ensure that watermarks are reliable (e.g., they do not have an unexpectedly high false positive rate), (ii) propose rigorous statistical tests to automatically evaluate whether a watermarking scheme satisfies our criteria, and (iii) design an evaluation suite to rank watermarks along three key dimensions: detectability, quality, and robustness. By running our framework with 3 frontier models (GPT-6 Astra, Opus 5, Gemini-3.8 Flash), we discover over 50 different watermarking schemes, including several that outperform prior works along all key dimensions. We complement this by a manual study of the discovered schemes, distilling the key ideas into smaller components, and individually studying the impact of each component across dimensions (detectability, quality, robustness) to better understand how the proposed schemes operate. Importantly, we find that the agents, on top of improving existing ideas, also discover fundamentally new ideas (e.g., aligning watermark scores with a random per-request direction). Overall, our work establishes the first steps of fully autonomous watermarking research, enabling the discovery of more reliable and effective watermarks.

1 INTRODUCTION

Large Language Model (LLM) watermarking, which embeds a signal invisible to humans in model outputs, has emerged as the standard for tracing AI-generated text: regulators mandate it (European Parliament and Council of the European Union, 2024), and major providers deploy it (Dathathri et al., 2024; Anthropic, 2026). This makes research on new watermarking schemes crucial as they have to be reliable enough to prevent false accusations and robust enough to prevent evasion across real-world deployments. Recent progress has been steady but limited with improvements mostly driven by refining existing designs, e.g., with entropy weighting (Lee et al., 2024), or key switching (Sander et al., 2026), and the pace of iteration capped by that of human researchers. Autonomous research agents are thus a natural and promising direction for further accelerating LLM watermarking research.

Autoresearch Autoresearch systems let agents propose, implement, and evaluate ideas in a loop (Romera-Paredes et al., 2024; Novikov et al., 2025), and have already led to scientific breakthroughs (OpenAI, 2026). Fields with readily verifiable outputs, such as mathematics with Lean-generated proofs, are especially well suited to this approach. Watermarking can also benefit from autoresearch, but its evaluation requires more care: useful properties such as detectability, robustness, and quality trade off, so improvement on a single metric need not represent a better watermark. Also, agents may exploit gaps in the evaluation harness (i.e., reward hacking), e.g., artificially inflating detectability/robustness by increasing the rate of falsely flagged human texts. A separate concern is that agents may produce results faster than humans can understand them, especially when their explanations are poor, making the resulting solutions difficult to trust. In this work, we show that autoresearch, when carefully executed with an adequate harness and methodology, discovers watermarks that are trustworthy, explainable, and Pareto-superior to existing watermarking schemes.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

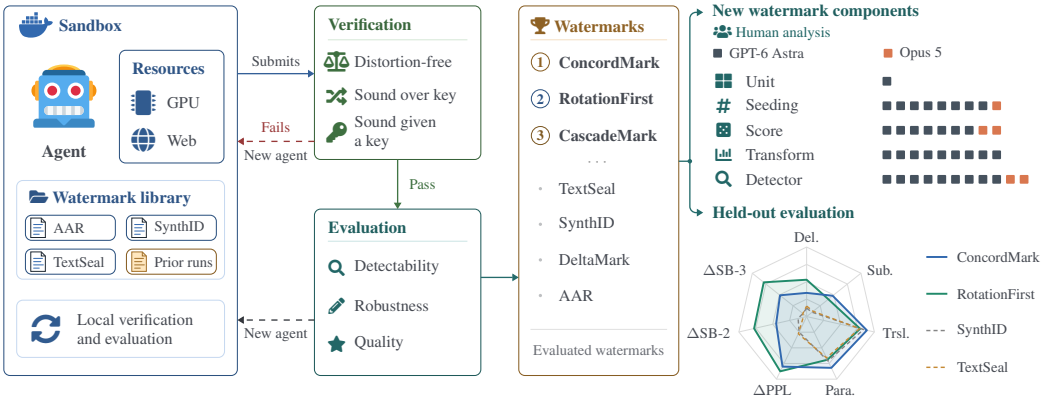


Figure 1: Overview of our work. In an isolated sandbox, an agent iterates on a watermarking scheme. Submissions are first verified and, upon passing, evaluated and added to a leaderboard. We then manually decompose the discovered schemes into 5 components and ablate them for insights. Lastly, we evaluate two selected schemes on a held-out test set and find that they outperform prior methods.

This work: autoresearch for LLM watermarks To enable autoresearch for watermarking, we mathematically formalize what makes a watermark valid (Sec. 3.2) using two criteria: soundness in expectation over the private key, and, given a fixed text distribution, the probability of sampling a key that leads to false flags should be small. Given these criteria, we design our autoresearch harness illustrated in Figure 1. An agent is fully isolated in an environment and discovers new watermarking schemes (left). It can submit its scheme to a *private* verification pipeline (middle) that (i) tests whether the proposed scheme violates our criteria (using tailored tests we designed in Sec. 3.3), and, upon passing, (ii) evaluates the schemes along three key axes (detectability, robustness, and quality).

Running our harness on three state-of-the-art agents, GPT-6 ASTRA, OPUS 5, and GEMINI-3.8 FLASH, we generated over 50 schemes, with some outperforming prior work on all our evaluated axes. To understand the key insights of these schemes, we manually classify them all into five atomic components (Figure 1, right) and ablate their individual effects (Sec. 4.2 and App. A). We also select two schemes, ConcordMark and RotationFirst, that we evaluate on a held-out evaluation set (Sec. 4.3). We find that ConcordMark achieves state-of-the-art robustness and is Pareto-superior to all evaluated prior work, whereas RotationFirst has performance similar to prior work at much higher text quality.

Main contributions Our main contributions are:

- We formalize the criteria for reliable watermarking, and design statistical tests that automatically detect violations (Sec. 3.2 and Sec. 3.3).
- We build an autoresearch harness for LLM watermarking around this verification (Sec. 3.4).
- We show that agents running our harness push the Pareto frontier of LLM watermarks, with two new schemes that outperform prior work on a held-out evaluation (Sec. 4.1 and Sec. 4.3).
- We distill the agents' schemes into 39 new components, ablate each of them, and identify fundamentally new watermarking ideas (Sec. 4.2).

2 RELATED WORK

LLM watermarking LLM watermarks modify the sampling procedure of an LLM to embed a (key-dependent) signal that a later statistical test can detect. Red-Green watermarks (Kirchenbauer et al., 2023) achieve this by distorting the next-token distribution through a pseudo-random green token bias. In contrast, recent schemes, including all in this work, are distortion-free, i.e., they preserve the next-token distribution in expectation over the key. AAR (Aaronson, 2023) uses the Gumbel-max trick, KTH (Kuditipudi et al., 2024) inverse-transform sampling, DiPMark and MCMark (Wu et al., 2024; Chen et al., 2025) reweight the distribution based on score-rankings, and SynthID (Dathathri et al., 2024), deployed in production, uses tournament sampling. Most recently, TextSeal (Sander et al., 2026) extends Gumbel-max sampling with dual-key generation and entropy-weighted detection.

Watermark properties Beyond detectability, benchmarks and toolkits (Piet et al., 2025; Tu et al., 2024; Pan et al., 2024) evaluate watermarks along several, often competing, properties. *Robustness* measures whether the watermark remains detectable after the text is edited, e.g., by word substitutions, translation, or paraphrasing (Kirchenbauer et al., 2024; Kuditipudi et al., 2024). *Quality* measures the impact of the watermark on the generated text, e.g., on perplexity or downstream tasks (Tu et al., 2024). Distortion-free schemes preserve the distribution of each generation, but, as a repeated context always yields the same scores, they reduce output diversity across generations (Gloaguen et al., 2026) which noticeably affects quality. *Security* measures whether an adversary can learn the watermark rules to spoof or scrub it (Jovanović et al., 2024), and is outside the scope of this work. Lastly, *reliability* requires the false positive rate of the detector to be controlled, e.g., Fernandez et al. (2023) show that the z-tests used in early works underestimate the false positive rate, and propose exact tests with de-duplication of repeated (context, token) pairs. Yet, in prior works, these guarantees hold only in expectation over the private key: a specific deployed key can still exhibit a high false positive rate.

Autoresearch Program search systems (Romera-Paredes et al., 2024; Novikov et al., 2025; Lange et al., 2026) let LLMs iteratively propose programs scored by an automatic evaluator, leading to new results in verifiable domains such as algorithm design and mathematics (OpenAI, 2026). Other works automate the entire research loop, from generating ideas to writing papers (Lu et al., 2026), discovering new training objectives (Lu et al., 2024), or improving LLM training (Karpathy, 2026), and benchmarks measure the research capabilities of agents (Chan et al., 2025; Wijk et al., 2025). Yet, all require a strict evaluator as otherwise agents may exploit loopholes, e.g., by modifying tests instead of solving the task (Zhong et al., 2026). To our knowledge, we are the first to propose an automated pipeline to verify and evaluate watermarks, enabling autoresearch for LLM watermarking.

3 METHOD

In this section, we introduce the necessary background on LLM watermarking (Sec. 3.1), formally define what makes a watermarking scheme valid (Sec. 3.2), explain our design for automatically verifying watermarking schemes (Sec. 3.3), and finally detail our autoresearch framework (Sec. 3.4).

3.1 PRELIMINARIES ON LLM WATERMARKS

In this part, we describe the four key components of most watermarking algorithms.

Mathematical description Let Σ be the finite vocabulary, and let $\omega \in \Sigma^*$ be a finite sequence of tokens. At each step t of autoregressive generation, the LLM returns $p_t \in \Delta(\Sigma)$, the conditional probability distribution of the next token given $\omega_{<t}$. Without a watermark, the next token is sampled according to p_t. With a watermark, we assume the existence of a sequence (U_{a_t}) of pseudo-random vectors in $[0,1]^{|\Sigma|}$ whose entries $U_{a_t,v}$ are i.i.d. uniform variables seeded by the context hash a_t, and a watermark function $f : [0,1]^{|\Sigma|} \times \Delta(\Sigma) \to \Delta(\Sigma)$ that transforms the next-token probability distribution p_t into a watermarked probability distribution $f(\tilde{U}_{a_t}, p_t)$. We refer to f as the *logits transformation* and (U_{a_t}) as the *watermark score*, where $(\tilde{U}_{a_t})$ are i.i.d. uniform variables derived from (U_{a_t}) and in most cases $\tilde{U}_{a_t} = U_{a_t}$. We give concrete examples in App. D.4. The next token ω_t, sampled according to $f(\tilde{U}_{a_t}, p_t)$, will be correlated with the sequence (U_{a_t}).

For detection, given any text $\omega \in \Sigma^*$, we recover the corresponding scores U_{a_t,ω_t} at retained positions $t \in D$, with $N := |D|$, after deduplicating repeated (seed, token) pairs. By assumption, under the null hypothesis (i.e., the text was generated without knowledge of the watermark), the retained scores U_{a_t,ω_t} are i.i.d. uniform random variables, whereas when the watermark is used, they are not. We can therefore build a statistic and a corresponding statistical test to detect the watermark, i.e., a *detector*.

Additional requirements Furthermore, we require the watermark to be distortion-free,

$$\forall p \in \Delta(\Sigma), \mathbb{E}_{\tilde{U}}[f(\tilde{U}, p)] = p. \tag{1}$$

This property ensures that a watermark has minimal impact on quality: with perfect randomness, the quality of the watermarked model is indistinguishable from its base model. Yet, in practice, as we explain below we can still observe mild quality degradation within distortion-free watermarks.

Practical considerations In practice, the sequence (U_{a_t}) needs to be computed from the sequence of tokens ω so that it can be recovered at detection time. A typical way to derive (U_{a_t}) is to use a hash of the preceding tokens and a private watermark key ξ to obtain a seed a_t for a pseudorandom number generator (App. G). The resulting scores $U_{a_t,v}$ remain approximately i.i.d. across distinct (seed, token) pairs and can be derived directly from the text given the private key. We refer to this as *context seeding*. Seeding creates some practical problems: in case of a repeated (seed, token) pair, the watermark scores are no longer i.i.d., making detection unsound under the null. For a single request, we use a cache mechanism: if a seed a_t is repeated we no longer apply the watermark. For repeated requests however, we cannot apply this cache as in the limit, the watermark is no longer applied. This lack of i.i.d. has also direct implications for the distortion-free property (Equation (1)) which in practice is therefore also violated across a larger corpus with repeated tokens. Thus, even for theoretically distortion-free schemes, we have to measure their impact on quality (Wu et al., 2026).

3.2 What Defines a Valid Watermarking Scheme

To automatically verify watermarking scheme validity at scale, we first need to formally define what makes a watermarking scheme valid. In particular, we identify two key criteria: *soundness over the key* and *soundness given a key*. Though our definitions build upon prior work, we find in App. C that several existing schemes fail our more stringent validity checks. In App. B, we further motivate our criteria by observing the failure modes of schemes designed by autoresearch without these safeguards.

Soundness over the key Soundness over the key is the property that, in expectation over the watermarking key ξ, the detector is a valid statistical test. Formally we require,

$$\forall \omega \in \Sigma^*, \forall \alpha \in [0,1], \quad \Pr_{\xi}[P_\xi(\omega) \leq \alpha] \leq \alpha, \tag{2}$$

where $P_\xi(\omega)$ is the p-value returned by the detector. Most prior watermarking schemes satisfy this property, which distinguishes statistical watermarks from zero-shot LLM-text detectors. However, this property alone does not guarantee that the watermarking scheme is reliable.

For instance, consider a deliberately naive watermarking scheme: using the private key ξ, we randomly partition the vocabulary into a hundred classes and declare a text watermarked if its last token lies in the first class. Averaged over the key, this occurs with probability at most $1/100$, so the scheme satisfies Equation (2). Yet its FPR can be high for a specific key: if, for some ξ, "." lies in the first class, most human texts will be classified as watermarked. Thus, while Equation (2) bounds the FPR in expectation over the private key, it does not ensure a well-behaved FPR for each fixed key.

Soundness given a key We therefore introduce a novel second requirement for watermarking schemes: soundness given a key. Ideally, we want to require that for a human text distribution and a p-value threshold α, the false positive rate is bounded by α. This is arguably impractical as it would require precisely specifying the human text distribution, which, e.g., noticeably differs between languages. Instead, we move the randomness to the key: we require that, for *every* non-degenerate (App. C) human distribution $\mathcal{H}_0$ over texts, the probability of sampling a key whose FPR on $\mathcal{H}_0$ exceeds α is at most β.

$$\forall \mathcal{H}_0 \in \Delta(\Sigma^*) \text{ non-degenerate}, \ \forall \alpha \in [0,1], \quad \Pr_{\xi}\Big[\Pr_{\Omega \sim \mathcal{H}_0}[P_\xi(\Omega) \leq \alpha] > \alpha\Big] \leq \beta. \tag{3}$$

For $\beta = 5\%$, if a watermarking scheme satisfies Equation (3), then for any fixed human text distribution, the probability of sampling a key for which the detector is not well calibrated (on that distribution) is at most 5%. Interestingly, as we show in App. J, a scheme satisfying Equation (3) can be derived from a scheme satisfying Equation (2) at the cost of its power: The same scheme with its p-value multiplied by $1/\beta$ also satisfies Equation (3) (while still satisfying Equation (2)).

3.3 Automatic Verification

Now we describe how we automatically test whether a proposed watermarking scheme violates our criteria (Sec. 3.2). To this end, we design three statistical tests to test whether a scheme is distortion-free (Equation (1)), sound over the key (Equation (2)), and sound given a key (Equation (3)).

1. **Distortion-freeness** (Algorithm 1): for given unwatermarked next-token distributions, we average the watermarked distributions over independently sampled private keys and test whether it significantly differs from the original. We check both individual probability shifts and smaller shifts spread across tokens, using independent samples to identify and test a potential bias direction. If either test detects a significant deviation after multiple testing corrections, the scheme is rejected.
2. **Soundness over the key** (Algorithm 2): for a given set of non-watermarked texts, the test verifies for independently sampled private keys whether the empirical false positive rate is abnormally high. If it is, it means that the scheme likely does not satisfy Equation (2) and thus is rejected.
3. **Soundness given a key** (Algorithm 3): for independently sampled private keys, we sample a separate set of non-watermarked texts (human or LLM-generated) per key and screen for an abnormally high empirical false positive rate. We then test whether too many keys screen positive, accounting for both the allowed fraction β of bad keys and the probability that a sound key screens positive. If they do, the scheme likely violates Equation (3) and is rejected.

3.4 A Framework for Watermark Autoresearch

Next, we describe our harness designed around three components: a dedicated vLLM watermarking library, a coding agent environment, and the automatic verification & evaluation pipeline (Sec. 3.3).

Research harness We illustrate the autoresearch loop in Figure 1. Our watermarking library is designed around four primary components: the context seeding, the watermark score, the logits transformation, and the detector, following the formalization described in Sec. 3.1. A watermark is an implementation of all 4 components. We defer additional design choices to App. E.

The research environment is initialized with the watermarking library and the baseline evaluations. The agent is tasked with proposing a watermarking algorithm that outperforms prior schemes in several key aspects: raw detectability and robustness to several attacks (e.g., word substitution, word deletion, paraphrasing, and backtranslation) while remaining distortion-free and statistically sound. We detail the initial prompt in App. K.1. We also give the agent a public configuration specifying how to evaluate and verify the proposed scheme. Once the agent completes its task, its proposed scheme is automatically evaluated (with a private configuration), and the evaluation results are added to the environment alongside the baselines. Then, a new agent takes over with access to all prior results and code modifications. The loop continues indefinitely. Inside the environment, the agent has full access to the Internet (e.g., it can download third-party libraries or read papers), a dedicated GPU, and API keys for third-party AI providers (primarily for experimenting with paraphrasing attacks).

To stimulate the agent even more, we run a two-phase setup. The first agent is told it has 4 available submissions to the private evaluation before being shut down: this allows it to set up its environment, and adjust in case of unexpected failures. Then, after the initial 4 submissions, subsequent agents are told they only have 1 submission left. We found in App. E.2 that this combined setup led to more creative solutions from the agents and less hyperparameter tuning.

4 Evaluation

In this section, we analyze the results of running our autoresearch framework across state-of-the-art agents. In Sec. 4.1 we analyze the trajectories of the agents during the autoresearch loop, while in Sec. 4.2 we break down the schemes proposed by the agents into individual categories and assess which metrics they impact. Lastly, in Sec. 4.3, we select two watermarking algorithms from the autoresearch runs that outperform prior work on our evaluated metrics.

Experimental setup We compare against the most popular prior and deployed schemes: AAR, SynthID, and TextSeal. Our private grader measures detectability and robustness as the TPR@1%FPR on 1000 LLAMA-3.1-8B 200 to 300 tokens long replies to ELI5 (Fan et al., 2019) prompts, on clean text and under four attacks: word deletion, synonym substitution, back-translation, and paraphrasing. For quality, we measure the distance to the unwatermarked model in perplexity, and in diversity (Self-BLEU) across 100 replies to the same 10 prompts, and, as these metrics are highly correlated, sometimes average them into a single quality metric (see App. L for un-aggregated results). Over 5 watermark keys, we report the worst TPR and median quality distance. We defer details to App. D.

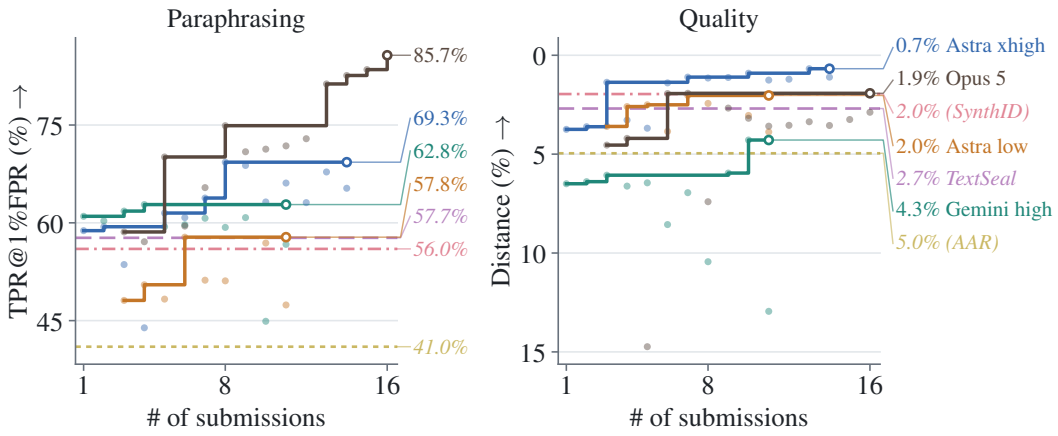


Figure 2: Submission history for each agent along two evaluated axes: robustness to paraphrasing and quality. Each dot corresponds to a submission and the solid lines are the best value so far. The baselines are in italic, and a parenthesis means the baseline *fails* our verification.

4.1 AUTORESEARCH TRAJECTORIES

We first focus on how the metrics evolved over the number of submissions to the evaluation harness.

Research agents and settings We use four different research agents: three frontier models GPT-6 ASTRA (xhigh), GPT-6 ASTRA (low), OPUS 5 (high), and a state-of-the-art efficient model GEMINI-3.8 FLASH (high). We let each agent run until its submission budget was exhausted. Using standard API prices, the research-agent costs are $357.08 for GPT-6 ASTRA (xhigh), $123.29 for GPT-6 ASTRA (low), $435.96 for OPUS 5 (high), and $13.81 for GEMINI-3.8 FLASH (high).

Main results Figure 2 shows the evolution of robustness to paraphrasing and of the watermark quality for different agents, with the baselines for comparison, for a total of 52 watermark schemes being evaluated. We find that all tested agents manage to improve their submissions over time, outperforming baselines in most cases. For two baselines (AAR and SynthID), the evaluated configurations do not pass our soundness given a key check: as we show in App. C this means that their robustness is slightly inflated. Moreover, only the submissions from GPT-6 ASTRA (both low and xhigh) are provably sound given a key. In contrast, submissions from OPUS 5, GEMINI-3.8 FLASH and the TextSeal baseline configuration satisfy the property only empirically (i.e., they do not fail our check). This means that the p-values of GPT-6 ASTRA are overly conservative, and, as we find in App. C, calibrating them to simply pass our empirical check increases the paraphrasing TPR of their best submission by 13–15 percentage points, reaching robustness similar to that of OPUS 5.

Research dynamics The research processes of GPT-6 ASTRA and OPUS 5 were similar, proposing a range of novel and creative watermarking ideas (App. A). Their first submissions mostly consisted of tweaks to the TextSeal baseline and attempts to correlate both the private verification and evaluation results with their local evaluation set-up. Both GPT-6 ASTRA agents then decided to submit only schemes that are provably sound given a key by multiplying their p-values by $1/\beta$ (Sec. 3.2) whereas OPUS 5 opted for empirical calibration on its public verification evaluation, which in practice turned out to correlate well with the private verification. Lastly, GEMINI-3.8 FLASH was significantly less creative. All its submissions consisted of hyperparameter tuning of the TextSeal watermark to improve the metrics and did not result in any new ideas.

4.2 BREAKDOWN OF THE AGENT RESEARCH DIRECTIONS

Next, we manually go through the schemes proposed by all agents, decompose them into five components (the four components from Sec. 3.1 and the *watermark unit*, a generalization introduced by the agents), and evaluate the impact of each component on watermark properties. We summarize the decomposition in Figure 3, and provide a detailed explanation of each component in App. A.

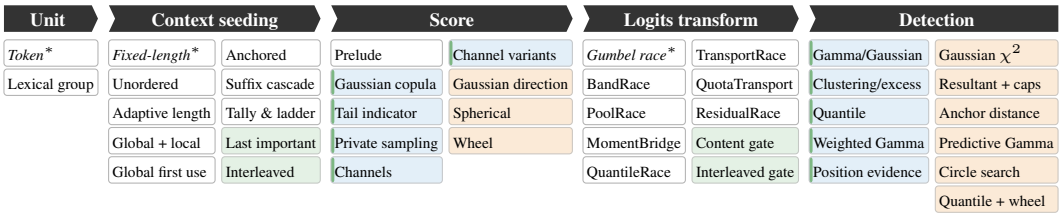


Figure 3: Atomic components extracted from the agents' watermarking schemes. Colors indicate the family (Fresh race , Private geometry , Skeleton) where components of the same family are interchangeable, and white components are compatible with every family. Skeleton schemes reuse the fresh-race scores and detectors (green edge). Components marked * come from prior work.

Watermark unit Most watermarks operate at the token level: each token is pseudo-randomly scored and the watermark modifies the logits according to both the score and the model probability distribution (Sec. 3.1). We find that the agents proposed to generalize this idea: let $\mathcal{G} : \Sigma \to \mathbb{N}$ be a mapping from the token space to a set of equivalence classes. In any watermark algorithm, one can replace a token v with its class $\mathcal{G}(v)$ and, given the next-token probability distribution $p_t \in \Delta(\Sigma)$, define the probability of each class $g \in \mathcal{G}(\Sigma)$ as $\Pr[g] = \sum_{v \in \Sigma} p_t(v) \mathbb{1}\{\mathcal{G}(v) = g\}$. If the watermark decides to sample a class g, the actual token is then sampled according to the residual probability

$$\forall v \in \Sigma, \Pr[v \mid g] = \mathbb{1}\{\mathcal{G}(v) = g\} \frac{p_t(v)}{\sum_{v' \in \Sigma} p_t(v') \mathbb{1}\{\mathcal{G}(v') = g\}}. \tag{4}$$

For instance, using a `Lexical group` as $\mathcal{G}$ naturally increases the scheme robustness but slightly lowers quality: if tokens from the same lexical group are swapped, the context remains unchanged.

Context seeding For context seeding, agents explore the trade-off alluded to in Kirchenbauer et al. (2024): increasing the context size lowers the robustness but improves the quality. Also, for small contexts, because of deduplication during detection (to guarantee soundness over the key (Sec. 3.1)), lowering the context size may also hurt detectability. Historically, prior works have converged on hashing the tuple of the candidate token and the k preceding tokens (`Fixed-length` in Figure 3). Agents iterated upon this idea along three main axes. The first idea is that tokens should not be treated equally: less frequent tokens should be dynamically assigned a lower k. This increases robustness without impacting quality significantly (because the distortion-free guarantee ensures that quality degrades only when contexts are repeated across texts). The second idea is to dynamically increase the context size: the first time we use a tuple as a context, it is of size 1. The next time we are about to use the same tuple, we increase the context size until the context has not been seen yet. This keeps the robustness benefits of short contexts without lowering the detectability. The third axis is more creative: assign tokens to categories, and when sampling a token from a given category, use a fixed-window context over the previous tokens from that category only. This increases robustness: edits in one category do not change the context for other categories.

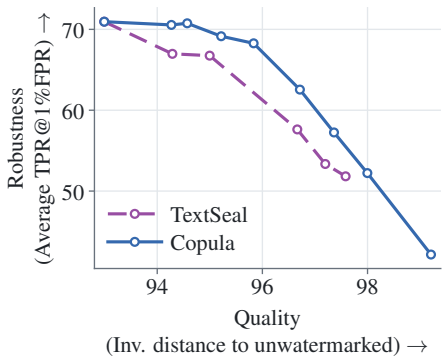


Figure 4: Robustness-quality for TextSeal dual key routing and `Gaussian copula`. Robustness is the average of all 4 attacks' TPR@1%FPR.

Score The score refers to the pseudo-random variable used by the logits transformation. For instance, with Red-Green watermarks, the scores are the colors of the tokens. Without loss of generality, as in Sec. 3.1, we refer to the *seeded* watermark score as $U_t \sim \mathcal{U}([0,1])^{|\Sigma|}$ and the final score used in the logits transformation as $\tilde{U}_t$. With different score designs, the agents tried to increase the quality while minimizing the impact on detectability. To do that, they consistently add a small amount of true randomness to the watermark. This is conceptually similar to the approaches by Sander et al. (2026); Dathathri et al. (2024) which also add true randomness to the watermark sampling

to improve quality. Yet, we find that agents explore concepts that substantially differ from prior works. With the `Gaussian copula`, they propose sampling random Gaussian noise $\eta_t \sim \mathcal{N}(0, I_{|\Sigma|})$ and applying the following transformation

$$\tilde{U}_t = \Phi(\rho\Phi^{-1}(U_t) + \sqrt{1-\rho^2}\eta_t), \tag{5}$$

where $\rho \in [0, 1]$ is a scheme parameter that trades off detectability for quality (in the limit, $\rho = 0$ has no watermark). In Figure 4, we compare this approach to TextSeal's random key switching (keeping all other watermarking components fixed) and find that it is strictly better: it outperforms TextSeal along all evaluated dimensions. With `Channels`, we use a finite pool of private keys and for each request randomly decide which key to use. Detection is run with every key and uses the Bonferroni correction. `Private geometries` are more complex, and we explain them further in App. A.

Logits transformation For logits transformations, most proposed schemes settle on Gumbel-max sampling (`Gumbel race`) from Aaronson (2023): as proven in Gloaguen et al. (2026), it maximizes watermark detectability. Yet, we find that the agents commonly propose a Gumbel-max sampling extension via residual clocks (`ResidualRace`). Given a context seed a_t, let U_{a_t} be the uniform watermark score. The first time a_t is visited, we initialize residual clocks $R_{a_t} = -\log U_{a_t}$, and otherwise, we use previously computed scores. Then we sample the next token according to the exponential race:

$$\tau_t = \min_u \frac{R_{a_t}(u)}{p_t(u)}, \qquad u_* = \arg\min_u \frac{R_{a_t}(u)}{p_t(u)}, \tag{6}$$
$$R_{a_t}(u) \leftarrow R_{a_t}(u) - \tau_t p_t(u).$$

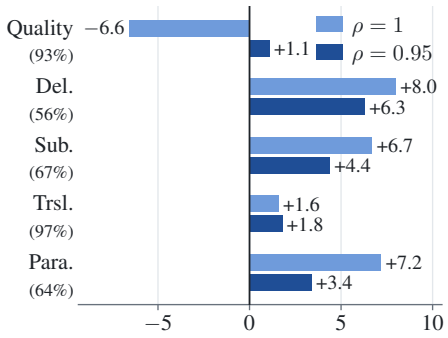


Figure 5: `ResidualRace` standalone or with `Gaussian copula` $\rho = 0.95$ to match its quality with the `Gumbel race`.

and replace the winning token's score with a fresh exponential random variable: $R_{a_t}(u_*) \sim \text{Exp}(1)$. In Figure 5, we compare `ResidualRace` against the `Gumbel race`, keeping all other components fixed. `ResidualRace` improves robustness against every attack at a slight cost to quality. Yet, adding `Gaussian copula` noise to the initial clocks recovers the quality of the `Gumbel race` while retaining most of the robustness gains. We find that most other logits transforms proposed by agents are designed to only increase the quality of the schemes at the expense of detectability. As we show in App. A this generally offers worse trade-offs than e.g., using the `Gaussian copula`.

Detection Most agents initially converged independently on the `Gamma` detector. Let $\omega \in \Sigma^*$ be a sequence of tokens and $U_t(\omega_t)$ the corresponding uniform watermark score at position t. Then we have

$$S_{\text{Gamma}}(\omega) = \sum_{t=1}^{|\omega|} -\log(1 - U_t(\omega_t)) \tag{7}$$

which follows a Gamma$(|\omega|, 1)$ distribution under the null. As we show in Figure 6 (left), they further improve it with `Clustering` and `Excess` thresholding. Lastly, using a model forward pass during detection to estimate the model probabilities can further improve detectability by a significant margin over `entropy weighting`, a prior method that also relies on a forward pass during detection (Figure 6, right). For `Private geometries`, which use a different class of detectors, we defer their full evaluation to App. A.

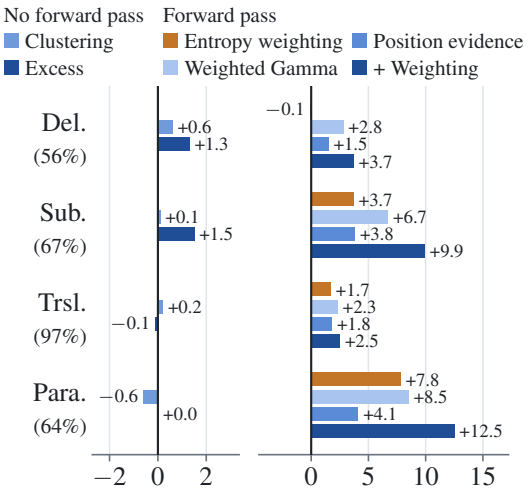


Figure 6: Comparison of detector variants without forward pass (left) and with forward pass (right).

Table 1: Evaluation of selected agent schemes. Each cell reports English / Chinese results. We **bold** all results that outperform the best baseline, and highlight the best result in each column.

		Robustness TPR@1 ↑				Quality deviation (%) ↓		
Scheme	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
Llama-3.1-8B								
AAR	92 / 89	8 / 16	11 / 22	70 / 17	64 / 42	1.9 / 1.7	5.2 / 6.3	12 / 15
SynthID	94 / 90	16 / 25	14 / 31	76 / 24	72 / 53	1.1 / 0.4	3.0 / 2.9	7.0 / 7.7
TextSeal (high)	95 / 91	17 / 31	15 / 38	78 / 28	74 / 56	1.5 / 0.9	2.8 / 3.5	6.2 / 8.6
ConcordMark (low)	92 / 89	**35 / 54**	**42 / 62**	**78 / 37**	**74 / 65**	**1.1** / 0.7	**2.0 / 2.3**	**4.4 / 4.8**
ConcordMark (high)	**96 / 91**	**39 / 61**	**49 / 69**	**87 / 55**	**83 / 77**	**1.1** / 0.7	**2.0 / 2.3**	**4.4 / 4.8**
RotationFirst	88 / 89	**48 / 60**	**38 / 62**	70 / **37**	66 / **61**	**0.7 / 0.2**	**0.6 / 0.7**	**1.3 / 1.7**
Qwen3.8-27B								
AAR	91 / 84	7 / 6	9 / 11	64 / 15	76 / 65	0.9 / 0.6	6.2 / 7.3	13 / 16
SynthID	93 / 88	11 / 10	12 / 16	74 / 21	76 / 72	1.2 / 1.1	2.9 / 3.5	6.3 / 8.1
TextSeal (high)	94 / 88	14 / 10	13 / 17	74 / 26	77 / 74	1.0 / 0.7	2.7 / 3.2	5.8 / 7.2
ConcordMark (low)	92 / 85	**25 / 23**	**32 / 34**	**74 / 29**	76 / 72	1.4 / 0.7	**1.9 / 1.8**	**4.1 / 3.6**
ConcordMark (high)	**96 / 90**	**32 / 31**	**40 / 43**	**85 / 47**	**83 / 82**	1.4 / 0.7	**1.9 / 1.8**	**4.1 / 3.6**
RotationFirst	84 / 81	**35 / 24**	**27 / 35**	61 / 22	68 / 64	2.0 / **0.5**	**0.4 / 0.5**	**0.9 / 1.3**

4.3 Independent Evaluation of the Best Directions

Next, we highlight two state-of-the-art schemes proposed by the agents that outperform prior works: ConcordMark as a robust watermarking scheme, and RotationFirst as a scheme with a strong quality focus. We give an in-depth description of both schemes and provide additional evaluation in App. H.

Experimental setup To mitigate overfitting due to the autoresearch process, we use a different evaluation protocol detailed in App. D.5. Notably, we generate 1000 replies to English and 1000 to Chinese prompts from WildChat (Zhao et al., 2024a), use DeepSeek-V4 Flash for back-translation and paraphrasing, and evaluate the replies generated by both Llama-3.1-8B and Qwen3.8-27B. Lastly, across 5 watermarking seeds, we report the worst TPR and the median quality distance. For consistency with the three baselines, we do not apply the soundness given a key correction (Sec. 3.2).

Main results Table 1 compares the two selected agent schemes with the three baselines. For each scheme, high means the detector requires a model forward pass. We see that ConcordMark (high) is Pareto better than all the baselines: on almost all metrics, it outperforms the best baseline. RotationFirst significantly outperforms all schemes on all quality metrics: its diversity is very close to the unwatermarked model. At the same time, it keeps relatively strong detection and robustness metrics making this scheme very suitable for model providers that desire strong watermarking with the smallest impact on quality. Importantly, for both agent schemes, the results from the autoresearch evaluation (Figure 2) transfer to new settings with different prompts and attackers. This suggests that autoresearch, with adequate verification, is a reliable method for improving watermarking schemes. In App. I, we also compare both schemes to prior semantic watermarks, which we also outperform.

5 Conclusion and Limitations

In this work, we enable for the first time autoresearch for LLM watermarking by introducing rigorous validity criteria (Sec. 3.2), statistical tests to verify them (Sec. 3.3), and a tailored harness (Sec. 3.4). We show that frontier agents go beyond hyperparameter tuning, introduce novel ideas (Sec. 4.2), with two schemes, ConcordMark and RotationFirst, outperforming prior work on held-out data (Sec. 4.3).

Limitations In autoresearch, the key challenge is to build powerful verification that agents cannot exploit, and that faithfully transcribes the desired property (i.e., in our case a watermark that is robust and with controlled false positives given a key). For watermarking, there are many desirable properties we did not include in this work e.g., security, radioactivity, applicability to open-weights setting, or extension to the multibit setting. We believe that future research should not only focus on establishing new schemes but rather on how to build verifiers and evaluations for those properties.

LLM USAGE

The goal of this work is to enable coding agents to discover novel and better watermarking schemes: hence, all the new watermark proposals in this work have been designed by fully autonomous coding agents. Yet, organizing the schemes into individual components and understanding their impact on watermark properties have required significant manual efforts. Notably, all ablation evaluations of the components have been designed by humans and executed with AI-assistance. The autoresearch harness used in this work has been coded with AI-assistance. The main paper has been written by hand, with AI-assistance for polishing the writing and the visuals. The proofs have been LLM-generated, then proofread and edited by humans.

ETHICS STATEMENT

This paper enables autonomous agents to discover new LLM watermarking schemes, which we believe has a positive societal impact: better watermarks support the provenance, auditing, and mitigation of large-scale misuse of AI-generated text. While our verification tests screen for schemes with unsound false positive rates (Sec. 3.3), LLM watermarks should not be used as sole evidence in high-stakes decisions. We also acknowledge that publicly disclosing new schemes, along with a detailed component-level analysis, may facilitate targeted attacks on them. Yet we argue that the benefits of advancing research in this area clearly outweigh the risks.

REPRODUCIBILITY STATEMENT

Our autoresearch harness is described in Sec. 3.4, and all details of the verification, the evaluation, the research agents (including their versions and reasoning efforts), and the baselines are given in App. D, with the full agent prompt in App. K. We fully specify the agent-generated components in App. A and the two selected schemes, ConcordMark and RotationFirst, in App. H. We also release our code, including the harness, the watermarking library, each agent final environment, a standalone implementation of all described components, and a standalone optimized implementation for ConcordMark and RotationFirst.

REFERENCES

Scott Aaronson. Watermarking of large language models. Talk at the Simons Institute for the Theory of Computing, August 2023. URL https://simons.berkeley.edu/talks/scott-aaronson-ut-austin-openai-2023-08-17.

Zhenxin Ai and Haiyun He. PASA: A principled embedding-space watermarking approach for LLM-generated text under semantic-invariant attacks. *arXiv preprint arXiv:2605.10977*, May 2026. URL https://arxiv.org/abs/2605.10977.

Anthropic. How Claude's text watermark works. Anthropic News, August 2026. URL https://www.anthropic.com/news/claude-text-watermark.

Dara Bahri and John Wieting. A watermark for black-box language models. *Transactions on Machine Learning Research*, 2026. URL https://arxiv.org/abs/2410.02099.

Jun Shern Chan, Neil Chowdhury, Oliver Jaffe, James Aung, Dane Sherburn, Evan Mays, Giulio Starace, Kevin Liu, Leon Maksin, Tejal Patwardhan, Lilian Weng, and Aleksander Mądry. MLE-bench: Evaluating machine learning agents on machine learning engineering. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=6s5uXNWGIh.

Liang Chen, Yatao Bian, Yang Deng, Deng Cai, Shuaiyi Li, Peilin Zhao, and Kam-Fai Wong. WatME: Towards lossless watermarking through lexical redundancy. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9166–9180. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.496. URL https://aclanthology.org/2024.acl-long.496/.

Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. Improved unbiased watermark for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20587–20601, Vienna, Austria, July 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1005. URL https://aclanthology.org/2025.acl-long.1005/.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, Ilia Shumailov, Ciprian Baetu, Sven Gowal, Demis Hassabis, and Pushmeet Kohli. Scalable watermarking for identifying large language model outputs. *Nature*, 634 (8035):818–823, October 2024. doi: 10.1038/s41586-024-08025-4. URL https://doi.org/10.1038/s41586-024-08025-4.

Robert B. Davies. Hypothesis testing when a nuisance parameter is present only under the alternative. *Biometrika*, 74(1):33–43, 1987. doi: 10.1093/biomet/74.1.33.

European Parliament and Council of the European Union. Regulation (EU) 2024/1689 of the european parliament and of the council of 13 june 2024 laying down harmonised rules on artificial intelligence and amending regulations (EC) no 300/2008, (EU) no 167/2013, (EU) no 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act). Official Journal of the European Union, June 2024. URL https://eur-lex.europa.eu/eli/reg/2024/1689/oj/eng.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: Long form question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 3558–3567, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1346. URL https://aclanthology.org/P19-1346/.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. In *2023 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–6. IEEE, December 2023. doi: 10.1109/WIFS58808.2023.10374576. URL https://doi.org/10.1109/WIFS58808.2023.10374576.

Thibaud Gloaguen, Robin Staab, Nikola Jovanović, and Martin Vechev. A unified framework for LLM watermarks. *arXiv preprint arXiv:2602.06754*, February 2026. doi: 10.48550/arXiv.2602.06754. URL https://arxiv.org/abs/2602.06754.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 22570–22593. PMLR, 2024. URL https://proceedings.mlr.press/v235/jovanovic24a.html.

Andrej Karpathy. autoresearch: AI agents running research on single-GPU nanochat training automatically. GitHub repository, March 2026. URL https://github.com/karpathy/autoresearch.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 17061–17084. PMLR, 2023. URL https://proceedings.mlr.press/v202/kirchenbauer23a.html.

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. In *The Twelfth International Conference on Learning Representations*, pp. 49660–49704, 2024. URL https://openreview.net/forum?id=DEJIDCmWOz.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=FpaCL1MO2C.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. ShinkaEvolve: Towards open-ended and sample-efficient program evolution. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=lKEdGCoDNC.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4890–4911, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.268. URL https://aclanthology.org/2024.acl-long.268/.

Chris Lu, Samuel Holt, Claudio Fanconi, Alex J. Chan, Jakob Foerster, Mihaela van der Schaar, and Robert Tjarko Lange. Discovering preference optimization algorithms with and for large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 86528–86573, 2024. doi: 10.52202/079017-2748. URL https://doi.org/10.52202/079017-2748.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of AI research. *Nature*, 651:914–919, March 2026. doi: 10.1038/s41586-026-10265-5. URL https://doi.org/10.1038/s41586-026-10265-5.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

OpenAI. Finite time blowup for the euler equation, September 2026. URL https://cdn.openai.com/pdf/315b36cd-ec98-4023-8342-93345194ece1/euler.pdf. Accessed: 2026-09-22.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. MarkLLM: An open-source toolkit for LLM watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 61–71, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-demo.7. URL https://aclanthology.org/2024.emnlp-demo.7/.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. Mark my words: Analyzing and evaluating language model watermarks. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 68–91. IEEE, April 2025. doi: 10.1109/SaTML64287.2025.00012. URL https://doi.org/10.1109/SaTML64287.2025.00012.

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. *Journal of Machine Learning Research*, 21(140):1–67, 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. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 613–625. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-naacl.40. URL https://aclanthology.org/2024.findings-naacl.40/.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M. Pawan Kumar, Emilien Dupont, Francisco J. R. Ruiz, Jordan S. Ellenberg, Pengming Wang, Omar Fawzi, Pushmeet Kohli, and Alhussein Fawzi. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024. doi: 10.1038/s41586-023-06924-6. URL https://doi.org/10.1038/s41586-023-06924-6.

John K. Salmon, Mark A. Moraes, Ron O. Dror, and David E. Shaw. Parallel random numbers: As easy as 1, 2, 3. In *Proceedings of the 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pp. 1–12. ACM, November 2011. doi: 10.1145/2063384.2063405. URL https://doi.org/10.1145/2063384.2063405.

Tom Sander, Hongyan Chang, Tomáš Souček, Tuan Tran, Valeriu Lacatusu, Sylvestre-Alvise Rebuffi, Alexandre Mourachko, Surya Parimi, Christophe Ropers, Rashel Moritz, Vanessa Stark, Hady Elsahar, and Pierre Fernandez. TextSeal: A localized LLM watermark for provenance & distillation protection. *arXiv preprint arXiv:2605.12456*, May 2026. doi: 10.48550/arXiv.2605.12456. URL https://arxiv.org/abs/2605.12456.

Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. WaterBench: Towards holistic evaluation of watermarks for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1517–1542, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.83. URL https://aclanthology.org/2024.acl-long.83/.

Hjalmar Wijk, Tao Roa Lin, Joel Becker, Sami Jawhar, Neev Parikh, Thomas Broadley, Lawrence Chan, Michael Chen, Joshua M. Clymer, Jai Dhyani, Elena Ericheva, Katharyn Garcia, Brian Goodrich, Nikola Jurkovic, Megan Kinniment, Aron Lajko, Seraphina Nix, Lucas Jun Koba Sato, William Saunders, Maksym Taran, Ben West, and Elizabeth Barnes. RE-bench: Evaluating frontier AI R&D capabilities of language model agents against human experts. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 66772–66832. PMLR, 2025. URL https://proceedings.mlr.press/v267/wijk25a.html.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. A resilient and accessible distribution-preserving watermark for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53443–53470. PMLR, 2024. URL https://proceedings.mlr.press/v235/wu24h.html.

Yihan Wu, Xuehao Cui, Ruibo Chen, and Heng Huang. Analyzing and evaluating unbiased language model watermark. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 142418–142433, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/e6643cfb7da77f9a4349df318e97aa3f-Paper-Conference.pdf.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for AI-generated text. In *International Conference on Learning Representations*, volume 2024, pp. 43738–43772, 2024b. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/beae9ed5316bcc48e616754c06c11875-Paper-Conference.pdf.

Ziqian Zhong, Aditi Raghunathan, and Nicholas Carlini. ImpossibleBench: Measuring LLMs' propensity of exploiting test cases. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=SeO4vyAj7E.

A BREAKDOWN OF ALL GENERATED COMPONENTS

In this section, we explain each of the watermarking components proposed by the research agents and visualized in Figure 3. In App. A.1, we explain in depth the decomposition of watermarks into five components (watermark unit, context seeding, scores, logits transformation, and detectors), and detail the notation used throughout. In the remaining subsections, we follow the architecture of Figure 3 and go through each component one by one: the watermark unit (App. A.2), context seeding (App. A.3), the watermark score (App. A.4), the logits transformation (App. A.5), and the detectors (App. A.6).

A.1 PRELIMINARIES AND NOTATION

Four components We decompose a watermarking scheme into the four key components introduced in Sec. 3.1, with the additional concept of the watermark unit, which the research agents introduced.

1. The *watermark unit* is the component on which the watermark is applied: in prior works, it is usually a token, but agents also proposed applying watermarks to equivalence classes of tokens.
2. *Context seeding* hashes the previously generated tokens into a seed that is used, along with the watermark private key, to pseudo-randomly sample the watermark score of every candidate unit.
3. The *watermark score* is the pseudo-random score, which can be combined with true randomness before the logits transformation uses it. At detection, only its pseudo-random part can be recovered.
4. The *logits transformation* combines the watermark score with the model's next-token distribution to pick the next token such that the watermarked generation remains distortion-free (Equation (1)).
5. The *detector* recomputes the pseudo-random scores of a candidate text, aggregates them into a statistic, and returns its p-value under the null hypothesis that the given text is not watermarked.

Notation Let Σ be the vocabulary and $\omega \in \Sigma^*$ a text, with ω_t its t^{th} token and $p_t \in \Delta(\Sigma)$ the model's next-token distribution given $\omega_{<t}$. A grouping $\mathcal{G} : \Sigma \to \mathbb{N}$ maps tokens to units. We write $u = \mathcal{G}(v)$ for the group of a candidate token v and

$$M_t(u) := \sum_{v \in \Sigma} p_t(v) \mathbb{1}\{\mathcal{G}(v) = u\} \tag{8}$$

for its mass under p_t. Context seeding maps the context c_t (a function of $\omega_{<t}$) and the private key ξ to a seed a_t. The seed is then used to pseudo-randomly sample a multivariate uniform random variable $U_{a_t,u} \sim \mathcal{U}([0,1])$ for every token unit u. Because the watermark score may add true randomness that is not computable at detection, we use the notation $\tilde{U}_{a_t,u}$ for the watermark score mixed with true randomness (and thus used by the logits transformation). The logits transformation selects a group u_* from $(\tilde{U}_{a_t}, p_t)$ and then samples the emitted token inside that group using the residual probability $p_t(v)/M_t(u_*)$. At detection, the scheme reconstructs the set D of retained (seed, unit) pairs of ω (e.g., after de-duplication of repeated (a_t, u) tuples as in Fernandez et al. (2023)), with $N := |D|$, aggregates them into a statistic $S(\omega)$, and reports a p-value $P(\omega)$. We write $Q(N, s) := \Pr[\mathrm{Gamma}(N, 1) \geq s]$ for the upper-tail probability of the $\mathrm{Gamma}(N, 1)$ distribution.

Experimental setup We ablate every component against a simple baseline AAR scheme: identity grouping, a two-token context, no private randomness in the score ($\tilde{U} = U$), the `Gumbel race`, and the scalar `Gamma` detector of Sec. 4.2. For each ablation, we swap one component (e.g., the `Gumbel race` with the `ResidualRace`) and report the difference in percentage points for our evaluation metrics (Sec. 4). Quality is one minus the average relative distance to the unwatermarked model in perplexity and both Self-BLEU scores. In the bar plots, positive values indicate an improvement, and values in parentheses give the reference quality or TPR@1%FPR. For some hyperparameter ablations, we use a reference other than the simple AAR baseline, which we then state in the caption.

A.2 WATERMARK UNIT

Prior works (Aaronson, 2023; Kirchenbauer et al., 2023; Dathathri et al., 2024; Sander et al., 2026) used a single token as the watermark: $\mathcal{G}$ is the identity and $M_t = p_t$.

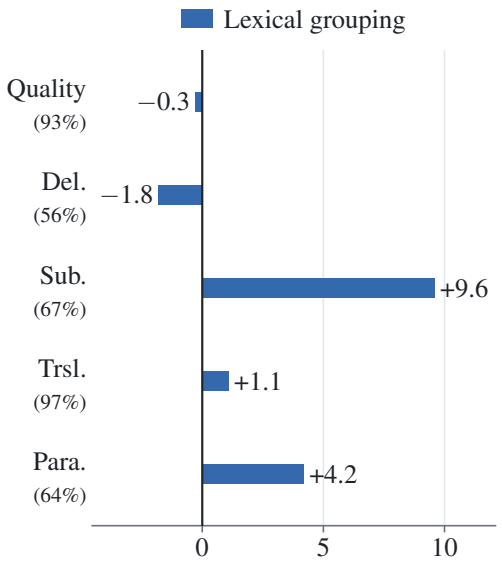


Figure 7: Impact of the `Lexical group` compared to the baseline identity grouping (`Token`).

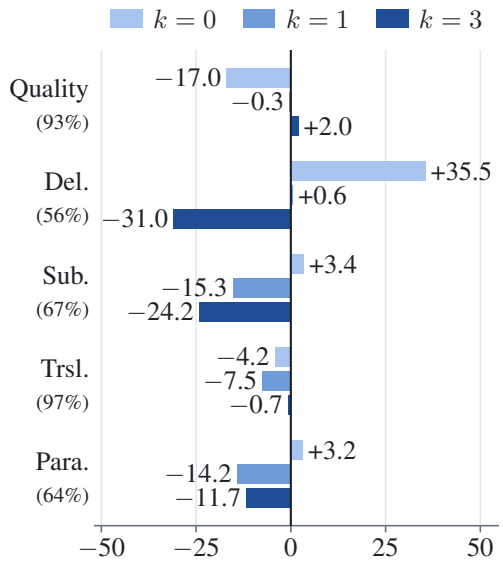


Figure 8: Impact of the context length k compared to the simple two-token baseline ($k = 2$).

`Lexical group` Agents replace tokens by classes of a public map $\mathcal{G}$ built from the tokenizer and WordNet 3.0: a token is decoded, stripped, and lower-cased, and if what remains is an ASCII word of at least three characters it maps to the representative of its first WordNet synset. Hence synonyms share a class, which makes it robust to substitution: a replacement inside a class leaves the seed and the observed "token" unchanged. Figure 7 confirms this intuition: `Lexical group` significantly improves robustness to substitution, and slightly improves backtranslation and paraphrasing robustness. However, it significantly lowers the robustness to word deletion and also slightly lowers the quality.

A.3 Context Seeding

For context seeding, unlike some prior works (Kirchenbauer et al., 2023), we assume the hashing function is without collision: the hash of a context is unique. Because the set of possible contexts is finite, this assumption is achievable in practice. We defer the details of the hashing functions, as well as an analysis of the different hashing functions, to App. G. We also use a per-request watermark cache: if a seed a_t hashed from the context is repeated, the next token is sampled without the watermark. This cache is required to guarantee that each individual request remains distortion-free.

`Fixed-length` Prior work hashes the tuple of the k preceding tokens (Kirchenbauer et al., 2023; Sander et al., 2026). In particular, $k = 0$ is the unigram scheme of Zhao et al. (2024b). As we show in Figure 8, the context size k controls a robustness-quality trade-off: shorter contexts survive more edits but repeat more often, which leads to correlation across a corpus of generated texts. Furthermore, at low context, i.e., $k = 1$ or $k = 0$, because we need to de-duplicate (seed, token) pairs at detection, the lack of context diversity lowers the detectability and therefore can counteract the gain in robustness.

`Unordered` It is the same as `Fixed-length` context seeding, except that it hashes the *set* of the k preceding tokens rather than their tuple (the candidate v is kept separate), so that a transposition of preceding tokens does not change the context. For $k \leq 1$, it is identical to `Fixed-length`. As we show in Figure 9, for $k = 2$ and $k = 3$ this change barely affects robustness and slightly improves quality. The agents also proposed an *unordered pair* variant (*Pair* in Table 7) that hashes the set $\{\omega_{t-1}, v\}$, which includes the candidate token itself, rather than the tuple (ω_{t-1}, v). Compared to the core scheme, it significantly increases robustness but also significantly lowers quality (Figure 9).

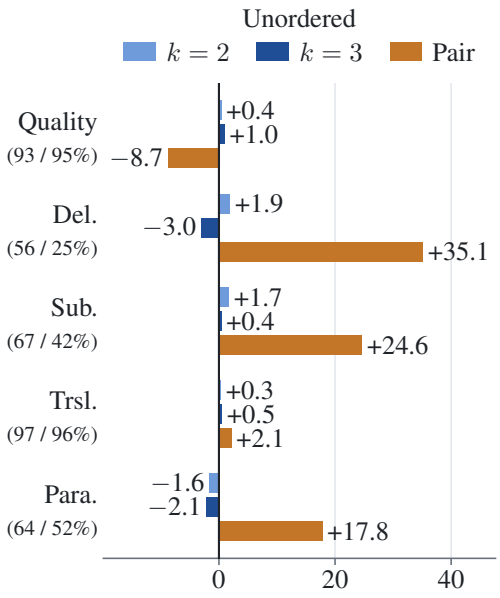


Figure 9: Impact of `Unordered` context seeding relative to ordered contexts of the same length k, and of the unordered pair relative to the core ($k = 2$). Parentheses give the ordered reference percentages for $k = 2$ and $k = 3$, respectively.

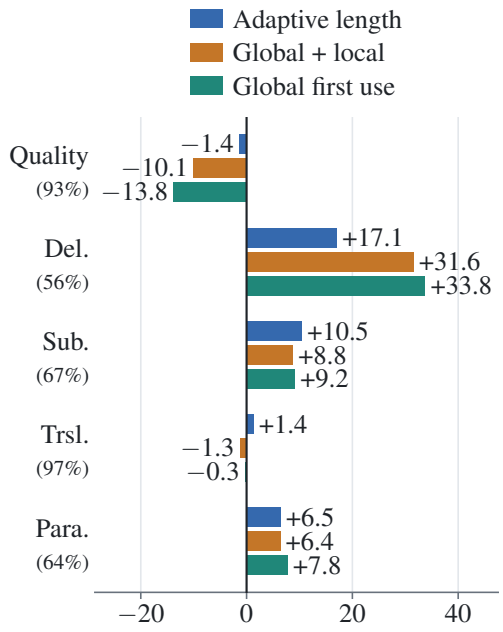


Figure 10: Impact of `Adaptive length`, `Global + local`, and `Global first use` context seeding compared to the simple baseline, which instead always uses a two-token context.

Adaptive and global length Rather than fixing k, these variants use a shorter context for the tokens that are unlikely to repeat anyway, so that only those positions that need a long context pay for one.

1. `Adaptive length`: given a classifier $\text{rare} : \Sigma \to \{0, 1\}$ (here that a token frequency in a fixed corpus is below 0.1%), if the previous token is rare, hash only that token ($k = 1$), otherwise hash the full two-token window. In Table 7, we also evaluate it with the `Excess` detector of App. A.6.
2. `Global + local`: fix a public set $E \subset \Sigma$ of eligible tokens (the agents used as E the tokens that have a WordNet entry), use unigram seeding (i.e., $k = 0$) if the candidate token is in E, and the standard k-token context otherwise, with $k = 2$ as in the simple baseline unless stated otherwise.
3. `Global first use` refines `Global + local` by applying unigram to a candidate in E only the first time it appears in the completion, and then using a two-token context when it appears again.

We show in Figure 10 that all of these approaches significantly increase the robustness compared to the simple fixed two-token baseline, but both global variants also significantly decrease the quality.

`Anchored` Let E again be a public token set and let s be the distance from the current token back to the most recent token of E. If $s \leq S_{\max} = 4$, use as context the tuple of that anchor token and the offset s. Otherwise ($s > S_{\max}$), skip the watermark. At detection, for every token ω_t, the detector finds the closest previous anchor, sweeps all permitted distances $s \in \{0, \ldots, S_{\max}\}$, and adjusts the p-values with a Bonferroni correction. Yet, as we show in Figure 11, while intuitively the slot search should improve robustness (editing tokens between anchor tokens does not change the context), the search is too expensive, and this scheme is worse than the baseline along every evaluated direction.

`Suffix cascade` and `Tally & ladder` These approaches share a goal: when a context was already used in the completion, derive a new seed instead of skipping the position, so that a short context keeps its robustness without losing detectability. They differ only in how the new seed is constructed.

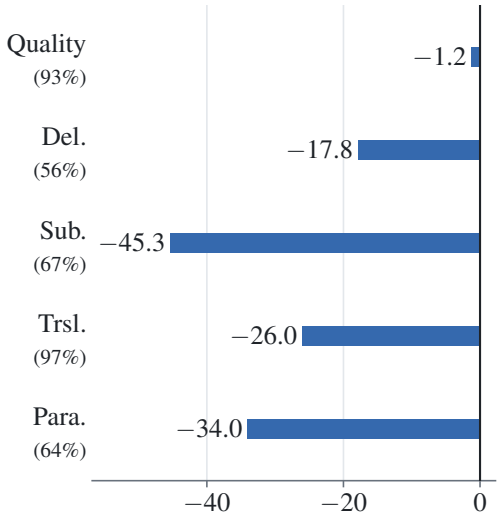


Figure 11: Impact of `Anchored` context seeding with slot search at detection, compared to the simple baseline with a fixed two-token context.

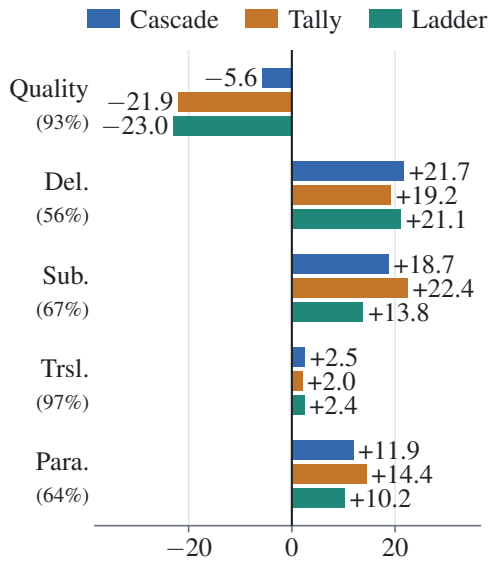


Figure 12: Impact of `Suffix cascade`, `Tally`, and `Ladder` context seeding compared to the simple baseline with a fixed two-token context.

1. `Suffix cascade`: try suffix lengths in increasing order and use the shortest one that has not already been used in this completion. If every allowed length has already been used, skip the watermark.
2. `Tally`: always keep a one-token suffix and hash the previous token together with the number of its earlier occurrences in the current completion, so that even a repeated context yields a fresh seed.
3. `Ladder` generalizes both: count the earlier uses of the suffix of every width $1 \leq w \leq w_{\max}$, select the shortest width whose count is below a threshold, and hash the width, the suffix, and the count. Counts are capped at 8192, beyond which the position falls back to ordinary sampling due to the specifics of the hash function (App. G). Optionally, a *floor* forces a width of at least 2 when the previous token ID is below 200 (as in ConcordMark, App. H.1), since for most tokenizers, low-ID tokens tend to be very common (e.g., letters, numbers, or punctuation). Unless stated otherwise, we use $w_{\max} = 4$, a threshold of 3, and the floor, and report all four combinations of $w_{\max} \in \{3, 4\}$ with and without the floor in Table 7. The `Ladder` makes almost every token position usable with a context of only one or two tokens, which is why it was selected for ConcordMark (Sec. 4.3).

As we show in Figure 12, the `Suffix cascade` indeed keeps most of the robustness of a short context, with a small quality cost, and, unlike $k = 1$, it improves over the baseline on all robustness metrics.

`Last important` and `Interleaved` These two schemes belong to the `skeleton family` (Figure 3). The `skeleton family` splits the vocabulary with a fixed token classifier into important (content) and filler tokens and then gives each of the two resulting token streams its own history for context seeding.

1. `Last important` ignores filler tokens when computing the context and uses the last important token as context, so that edits to filler tokens do not change the context of the next content word.
2. `Interleaved` keeps both streams separate: it uses the last important token as context when sampling an important token, and the last two filler tokens as context when sampling a filler token.

Both context seedings require a corresponding logits transformation, so we evaluate them in App. A.5.

A.4 Watermark Score

`Prelude` `Prelude` is the only component that can be composed with any other. It consists of sampling ordinarily at the start of the completion until the accumulated entropy $H_2(p_t) =$

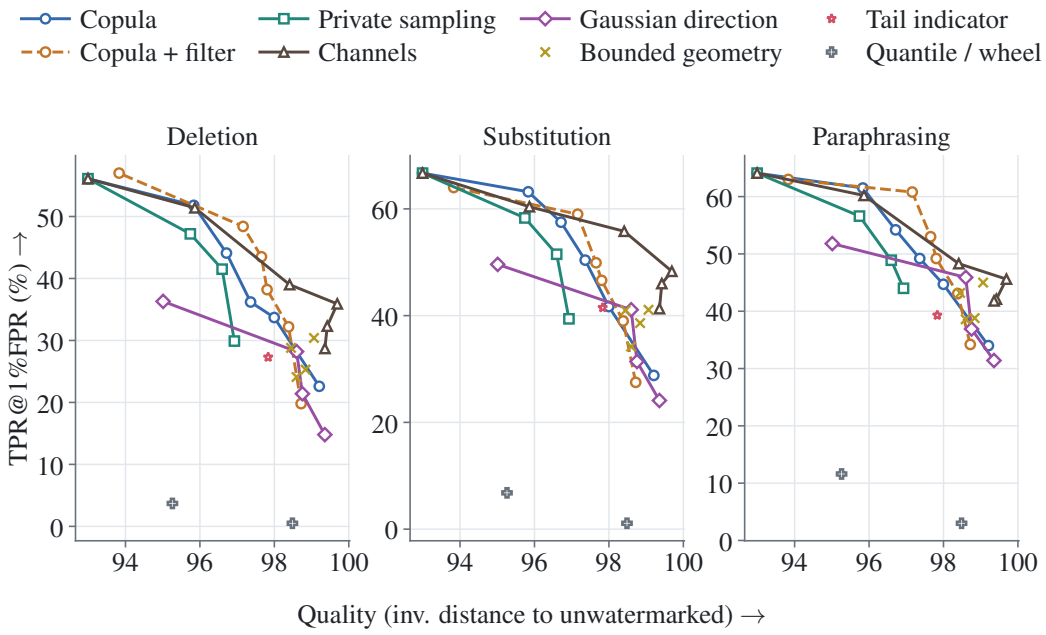


Figure 14: Quality-robustness trade-offs for the different watermark scores.

$-\log \sum_v p_t(v)^2$ of the first generated tokens reaches h nats or T tokens have been generated, whichever comes first. In practice, the agents use $h = 14$ and $T = 24$. For detection, we keep all tokens, including those of the prelude, since the detector cannot recompute the entropy without the model. We compare `Prelude` to the no-Prelude baseline and find in Figure 13 that it indeed improves quality with only a mild cost in terms of robustness.

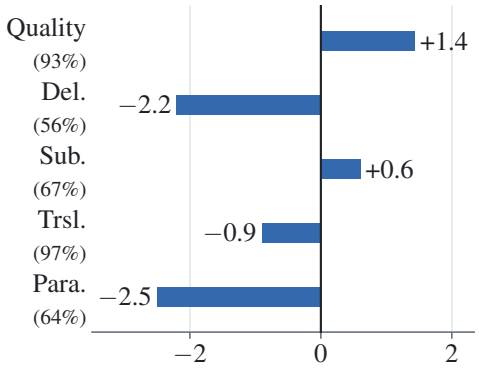


Figure 13: Impact of the entropy `Prelude`.

`Gaussian copula` `Gaussian copula` draws a fresh private normal $\eta_{t,u} \sim \mathcal{N}(0,1)$ at every position and sets $\tilde{U}_{a_t,u} = \Phi(\rho\Phi^{-1}(U_{a_t,u}) + \sqrt{1-\rho^2}\,\eta_{t,u})$. The goal is to inject true randomness into the watermark score to increase quality at the cost of detectability. At $\rho = 0$, the scheme is unwatermarked, and at $\rho = 1$, it uses only the watermarked pseudo-random scores. As we show in Sec. 4.2, it is a more effective way of improving quality via true randomness than the approaches proposed in prior works.

The `Gaussian copula` can be further refined with a common pair filter: we replace ρ with a function of the current context, $\rho_{t,u} = \rho\,\mathbb{1}\{(a_t, u) \notin \mathcal{F}\}$, where $\mathcal{F}$ is the set of the 10,000 most frequent (seed, token) pairs of a public corpus (in this case, the first 20,000 texts of the public ELI5 corpus, counting each pair at most once per text). The goal is to further increase the quality with a better detectability trade-off. Yet, as we show in Figure 15, using the simpler $\rho \in [0, 1]$ is better in practice.

`Tail indicator` For each unit u, set $B_u = \max(1, \lfloor 1/\max(M_t(u), 2^{-40}) \rfloor)$ and $b_u = 1 - 1/B_u$, draw a private $V_u \sim \mathcal{U}([0,1])$, and let $\tilde{U}_{a_t,u} = b_u + V_u/B_u$ if $U_{a_t,u} \geq b_u$ and $b_u V_u$ otherwise. Thus, $\tilde{U}$ is uniform and the scheme is distortion-free, while the detector still reads the full U. Yet, Figure 14 shows that it has a significantly worse quality-robustness trade-off than e.g., the `Gaussian copula`.

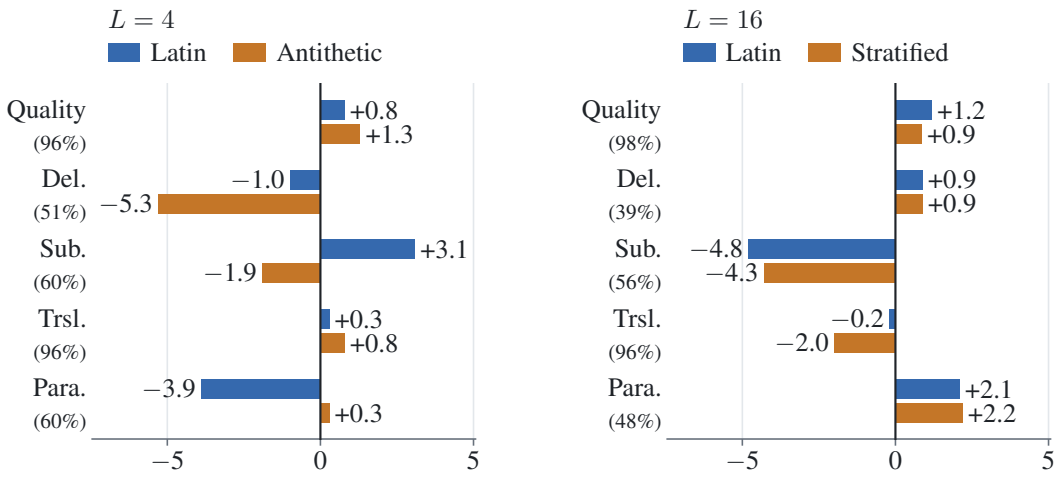


Figure 16: Impact of the three correlated `Channel variants` compared to L independent `Channels`, keeping the channel count L and the Bonferroni correction fixed. Parentheses give the independent-channel reference percentages in each panel, while the bars show the differences in percentage points.

`Private sampling` This is the naive way of adding true randomness: with probability r, ignore the watermark at this position and sample from p_t. However, as we show in Figure 14, it also has a significantly worse quality-robustness trade-off than e.g., the `Gaussian copula`.

`Channels` `Channels` maintain L independent keyed fields $U_{j,a,u}$ and draw a single channel $J \sim \mathcal{U}\{0, \ldots, L-1\}$ per request, using $\tilde{U}_{a_t,u} = U_{J,a_t,u}$ throughout the completion. The goal is the same as the `Gaussian copula`, spending private randomness on quality, except that here the randomness is discrete: two requests that share a prompt and a context take different fields. The cost is paid at detection, where the detector runs all L fields and corrects its p-value by a Bonferroni factor L. However, as we show in Figure 14, that factor dominates quickly: quality saturates around $L = 32$ while the robustness to every attack keeps falling, so only a small L is worthwhile.

To account for the L tests, we can use either the Bonferroni or the Šidák correction. The L per-channel p-values are independent under the null, so the Šidák correction $1 - (1 - \min_j P_j)^L$ is valid and strictly tighter than Bonferroni, and the submissions use both. Yet, we observe no differences in practice (Table 10).

`Channels` can be refined by correlating the L fields so that the channel choice costs less detectability.

1. `Latin` channels derive them from a single base uniform by rotation, $U_{j,a,u} = (U_{a,u} + j/L) \bmod 1$.
2. `Antithetic` channels, which need normal scores $X = \Phi^{-1}(U_{j,a,u})$, pair each keyed X with $-X$.
3. `Stratified` channels use a keyed permutation and keyed jitters, $U_{j,a,u} = (\sigma_{a,u}(j) + \upsilon_{j,a,u})/L$.

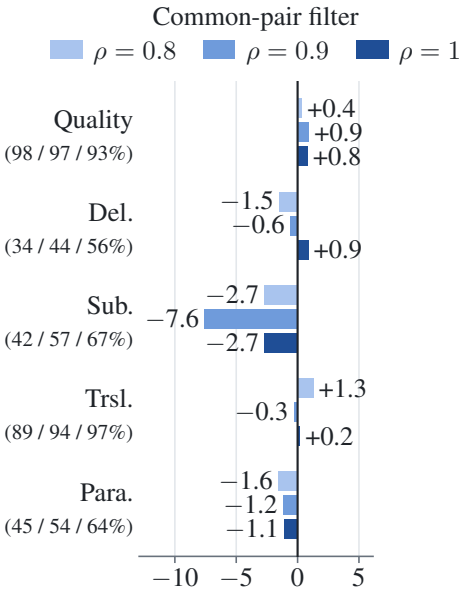


Figure 15: Impact of the common-pair filter compared to the unfiltered `Gaussian copula` at the same correlation parameter ρ. Parentheses give the unfiltered reference percentages for $\rho = 0.8$, $\rho = 0.9$, and $\rho = 1$, respectively.

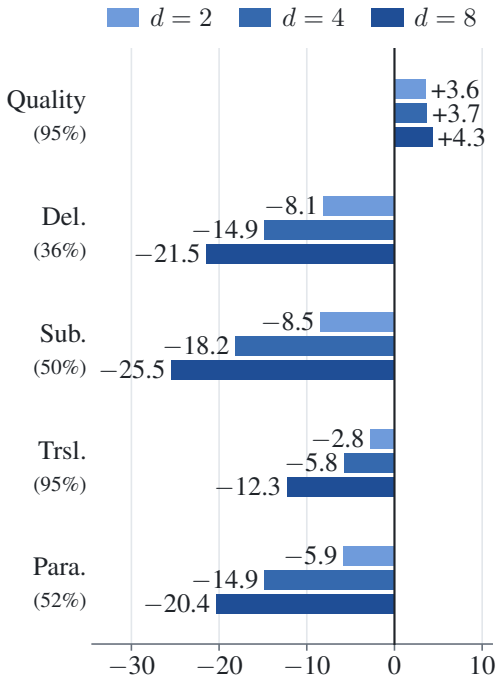


Figure 17: Impact of the `Gaussian direction` dimension d, compared to the scalar case $d = 1$.

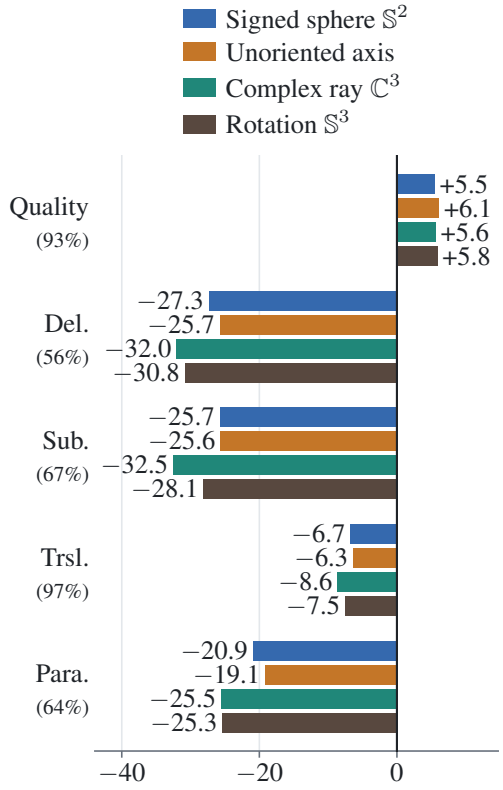


Figure 18: Impact of the bounded `Spherical` geometries compared with the simple baseline.

All three keep the same detector and use only the Bonferroni correction (Šidák is no longer valid due to the dependence between tests), so Figure 16 compares each against independent channels at the same L. Yet, none of the three is a reliable improvement: the differences are within a few points everywhere and change sign across attacks. We therefore recommend using the simpler `Channels` version with independent keys, which additionally benefits from the more powerful Šidák correction.

`Gaussian direction` This is the first of the private-geometry scores (Figure 3), and it generalizes `Channels` from a discrete choice to a continuous one. We use the context seed and watermark private key to sample a multivariate Gaussian $Z_{a_t,u} \sim \mathcal{N}(0, I_d)$, and *per request* we draw a private unit direction $A \in \mathbb{S}^{d-1}$ once, used for the whole completion:

$$\tilde{U}_{a_t,u} = \Phi(A \cdot Z_{a_t,u}). \tag{9}$$

Hence, two different requests (even those with the same prompt) use different directions, and thus different watermark scores, which increases the output diversity. As we show in Figure 17, increasing the dimension d significantly improves the quality at the cost of detectability, and thus of robustness.

`Spherical` These scores follow the same idea as the `Gaussian direction`, but on compact spaces.

1. `Signed sphere` draws $Z_{a_t,u} \sim \mathcal{U}(\mathbb{S}^2)$ and a private $A \in \mathbb{S}^2$, and sets $\tilde{U}_{a_t,u} = (1 + A \cdot Z_{a_t,u})/2$.
2. `Unoriented axis` draws $Z_{a_t,u} \sim \mathcal{U}(\mathbb{S}^2)$ and a private axis $A \in \mathbb{S}^2$, and sets $\tilde{U}_{a_t,u} = |A \cdot Z_{a_t,u}|$.
3. `Complex ray` draws $Z_{a_t,u}$ uniformly from the unit sphere of $\mathbb{C}^3$ and a private unit vector A of $\mathbb{C}^3$, both modulo phase, with $\tilde{U}_{a_t,u} = |A^* Z_{a_t,u}|^2 (2 - |A^* Z_{a_t,u}|^2)$.
4. `Rotation` draws $Z_{a_t,u} \sim \mathcal{U}(\mathbb{S}^3)$ and a unit quaternion A, with $\tilde{U}_{a_t,u} = F(|A \cdot Z_{a_t,u}|)$, where $F(r) = \frac{2}{\pi}[\arcsin r + r\sqrt{1 - r^2}]$.

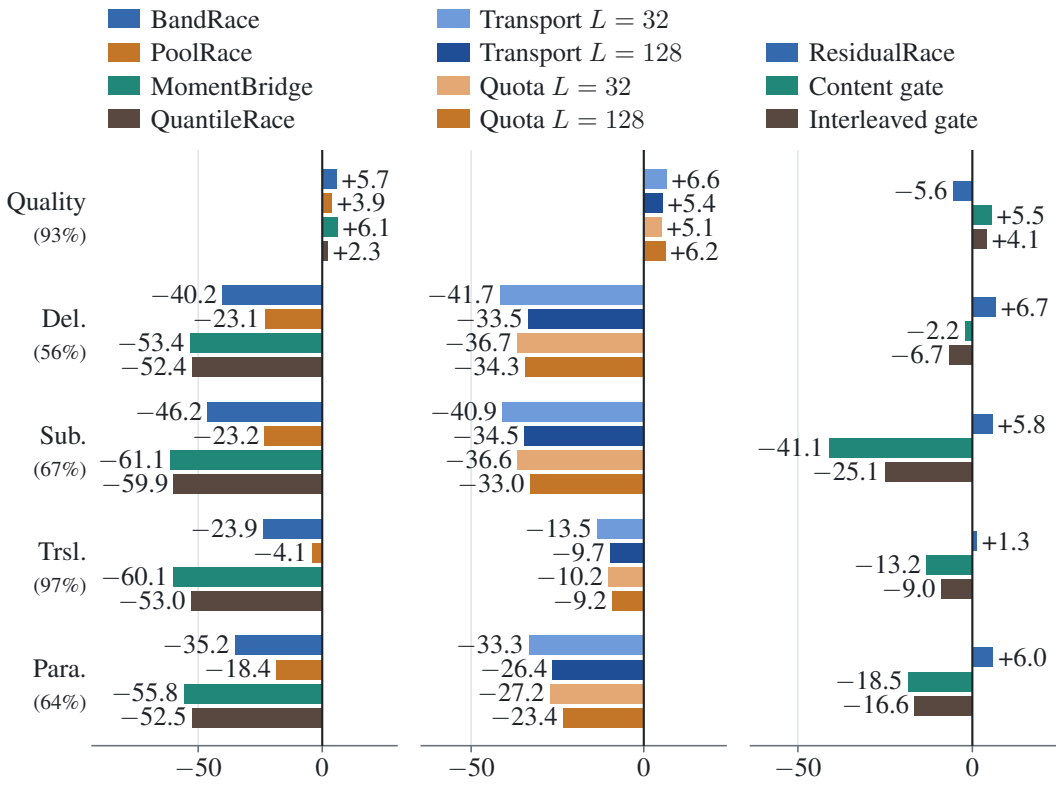


Figure 19: Impact of logits transformations compared to the `Gumbel race` baseline.

Figure 18 evaluates all four spherical geometries (using their corresponding detectors) compared to the simple baseline. We find that they all trade off quality for detectability and that `Unoriented axis` has the most favorable trade-off among them with the `Gumbel race`, whereas `Signed sphere` does with the `ResidualRace` (Table 8). Because the `Gaussian direction` and the spherical geometries only change the score $\tilde{U}$, they can be combined with any logits transformation: in Table 8, we evaluate each of these scores both with the `Gumbel race` and with the `ResidualRace` (App. A.5).

`Wheel` Draw a keyed offset U_{a_t} and a per-request random rotation $\varphi \sim \mathcal{U}([0, 1])$, and feed $\tilde{U}_{a_t,u} = (U_{a_t,u} + \varphi) \bmod 1$ to the `QuantileRace` transformation of App. A.5. It is the continuous limit of the `Latin` channels of Figure 16: the rotation φ is drawn from the continuous circle rather than from the discrete set of L evenly spaced angles. It hence requires the rotation-invariant detector of App. A.6.

A.5 Logits Transformation

`Gumbel race` This is the transformation of Aaronson (2023), which we rewrite here with the group notation for clarity:

$$G_{t,u} = -\log(-\log \tilde{U}_{a_t,u}), \qquad u_* = \arg\max_u \left[\log M_t(u) + G_{t,u}\right], \tag{10}$$

after which the token is drawn inside u_* with the residual probability $p_t(v)/M_t(u_*)$. By the Gumbel-max trick, u_* has distribution M_t in expectation over the key, so the scheme is distortion-free. Equivalently, it is an exponential race, in which the selected group u_* minimizes $-\log \tilde{U}_{a_t,u}/M_t(u)$.

`BandRace` Partition the groups into bands whose masses differ by less than a factor of four,

$$\mathcal{B}_{t,b} = \{u : 4^b \le M_{t,\max}/M_t(u) < 4^{b+1}\},$$

pick a band randomly with probability equal to its total mass, and run the `Gumbel race` inside it. This increases quality as the choice of the band is truly random, but lowers the detectability (Figure 19).

`PoolRace` It is the black-box equivalent of the `Gumbel race` (i.e., it is an approximation of the `Gumbel race` that does not require access to the logits), similar to what has been proposed in Bahri & Wieting (2026): draw $n = 16$ tokens V_i independently from p_t, let N_u count the proposals in group u, and pick

$$u_* = \arg\min_{u:N_u>0} \frac{-\log \tilde{U}_{a_t,u}}{N_u}. \tag{11}$$

As we see in Figure 19, this variant slightly improves the quality because the n proposals are drawn from p_t with true randomness, but it harms the detectability compared to the `Gumbel race` baseline.

`MomentBridge` The closest equivalent to this scheme in the literature is SynthID (Dathathri et al., 2024), where the watermark distribution is refined iteratively using the keyed watermark scores. Let $K \in \mathbb{N}$ be the number of iterations. Start with the weights $w_u^{(0)} = M_t(u)$ and the fixed costs $c_u = -\log M_t(u)$. Let $X_{t,u} = \Phi^{-1}(\tilde{U}_{a_t,u})$ be the normal watermark score. For $i = 1, \ldots, K$, sample random Gaussians $\eta_{i,u}$ and compute:

$$z_{i,u} = \frac{X_{t,u}}{\sqrt{K}} + \eta_{i,u} - \frac{1}{K}\sum_{\ell=1}^{K} \eta_{\ell,u}.$$

Let

$$\bar{c}_i = \sum_u w_u^{(i-1)} c_u, \qquad \bar{z}_i = \sum_u w_u^{(i-1)} z_{i,u},$$

and update the weights iteratively with

$$w_u^{(i)} = w_u^{(i-1)} \left(1 + \frac{r_{i,u}}{A_i}\right),$$

where $A_i = \max_{u:w_u^{(i-1)}>0} |r_{i,u}|$,

$$b_i = \frac{\sum_u w_u^{(i-1)}(c_u - \bar{c}_i)(z_{i,u} - \bar{z}_i)}{\sum_u w_u^{(i-1)}(c_u - \bar{c}_i)^2}, \text{ and } r_{i,u} = z_{i,u} - \bar{z}_i - b_i(c_u - \bar{c}_i).$$

The next group is sampled randomly according to the $w^{(K)}$ distribution. Because of the true randomness in $\eta_{i,u}$, this logits transformation trades off quality for detectability. However, as we show in Figure 19, it achieves a comparatively poor trade-off between the quality and the detectability.

`QuantileRace` It replaces the `Gumbel race` with inverse-CDF selection: order the groups (or tokens) of the vocabulary by a keyed pseudo-random permutation $R_{a_t,u}$, give each group u an interval of length $M_t(u)$, and emit the group whose interval contains the watermark score $\tilde{U}_{a_t}$ (Figure 19).

`TransportRace` and `QuotaTransport` This is the second black-box family, and the only one that couples the transformation to the `Channels`. Draw L tokens from the model, compute the keyed score $E_{j,a_t,\mathcal{G}(V_i)}$ of every (channel, proposal) pair, and choose the permutation σ_* that maximizes $\sum_j E_{j,a_t,\mathcal{G}(V_{\sigma(j)})}$. `QuotaTransport` stratifies the proposal pool before the same assignment, drawing one proposal from each of the L mass intervals $(i + W)/L$ for a fresh $W \sim \mathcal{U}([0, 1])$, so that the pool covers the distribution rather than concentrating on its mode. We evaluate both at $L \in \{32, 128\}$ with the `Gamma` detector and a Bonferroni correction. As we see in Figure 19, all four evaluated configurations substantially improve quality, but they are less robust than the `Gumbel race` baseline.

`ResidualRace` This is the extension of the `Gumbel race` described in Sec. 4.2. We keep one residual clock $R_{a_t,u}$ per (context, group) pair, initialized at $-\log \tilde{U}_{a_t,u}$ on first use. At each step, we sample the next token according to the exponential race:

$$\tau_t = \min_u \frac{R_{a_t,u}}{M_t(u)}, \qquad u_* = \arg\min_u \frac{R_{a_t,u}}{M_t(u)}, \qquad R_{a_t,u} \leftarrow R_{a_t,u} - \tau_t M_t(u),$$

and replace the winning clock with a fresh exponential random variable: $R_{a_t,u_*} \sim \mathrm{Exp}(1)$. Because of the memorylessness of the exponential distribution, all losing residual exponentials are independent given the past, so the next winning probability is still $M_t(u)$ even when a context recurs and its

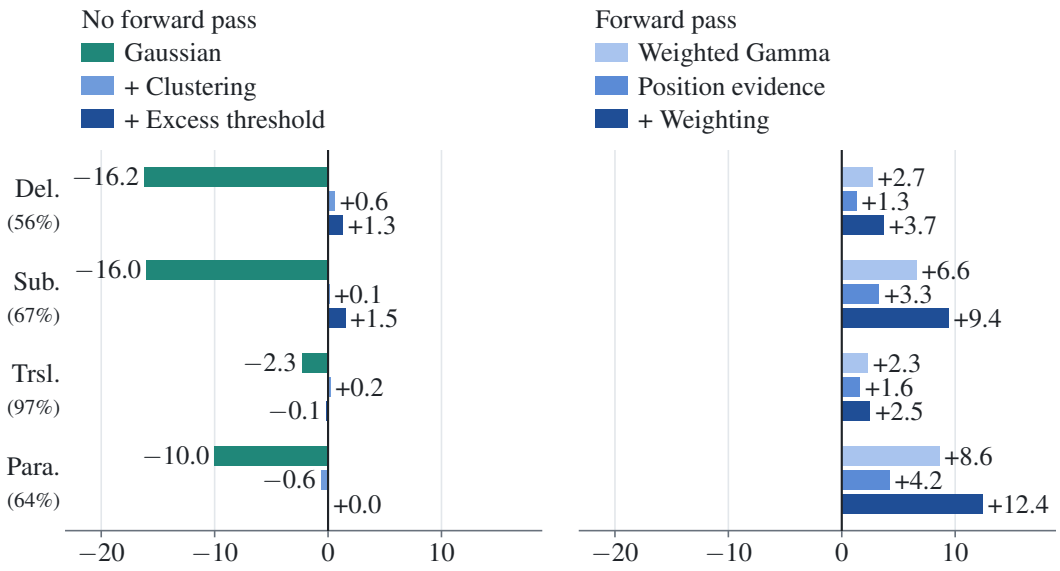


Figure 20: Comparison of scalar detectors without a model forward pass (left) and with a forward pass (right). Quality is not shown because changing the detector does not change the generated texts.

masses have changed. This is what makes it sound to reuse previous clocks in the exponential race when a context recurs (Proposition 5). As we show in Figure 19, it increases detectability at the cost of quality. Yet, as we have shown in Figure 5, adding `Gaussian copula` noise to the initial residual clocks recovers the quality of the `Gumbel race` baseline while still retaining the detectability gains.

`Content gate` and `Interleaved gate` This is the logits transformation of the `skeleton family`, which relies on either the `Last important` or the `Interleaved` context seeding. Let $\mathcal{C}$ be the set of important groups, and let $r_t = \sum_{u \in \mathcal{C}} M_t(u)$. At every step, draw a fresh gate $B_t \sim \text{Bernoulli}(r_t)$.

1. The `Content gate` uses the `Gumbel race`, seeded with the `Last important` context, within the important groups when $B_t = 1$ and otherwise samples from p_t conditionally on the filler groups.
2. The `Interleaved gate` uses a `Gumbel race` inside whichever group the gate selects (i.e., either the important or the filler group), seeding the `Gumbel race` with the corresponding group context.

As we see in Figure 19, the `Interleaved gate`, which watermarks both streams, outperforms the `Content gate` in robustness but has worse quality, though both gates show limited robustness overall.

A.6 Detection

Scalar `Gamma` and `Gaussian` The `Gamma` detector sums exponential scores,

$$S(\omega) = \sum_{t \in D} -\log(1 - U_{a_t, \omega_t}) \sim \text{Gamma}(N, 1).$$

The `Gaussian` detector sums the normal scores instead,

$$S(\omega) = N^{-1/2} \sum_{t \in D} \Phi^{-1}(U_{a_t, \omega_t}) \sim \mathcal{N}(0, 1).$$

Both detectors weight the evidence differently: unlike the `Gaussian` detector, the `Gamma` detector gives more weight to the tail of the distribution, i.e., to the tokens with very high watermark scores. As we see in the left panel of Figure 20, the global `Gamma` detector significantly outperforms the `Gaussian` detector without any change to the generation algorithm, and therefore at no cost to the text quality.

Gamma `Clustering` and `Excess` threshold These two detectors are refinements of the `Gamma` detector that group the evidence by context before aggregating it. Let $\mathcal{H}(\omega)$ be the set of distinct contexts in ω and $D_a(\omega)$ the set of distinct successors of each context a in ω, and let $m_a = |D_a(\omega)|$.

1. `Clustering` calibrates each context separately, and then aggregates the calibrated values,

$$S_a(\omega) = \sum_{v \in D_a(\omega)} -\log(1 - U_{a,v}),$$
$$S(\omega) = \sum_{a \in \mathcal{H}(\omega)} -\log Q(m_a, S_a(\omega)) \sim \mathrm{Gamma}(|\mathcal{H}(\omega)|, 1),$$

   so that a context with many successors cannot dominate the statistic on its own. Indeed, during generation, all successors of a given context take part in the same race, which means that two observations of the same race (i.e., context) carry less information than two observations of distinct races. Hence, even though the detection remains sound as long as no (context, candidate) pair is repeated, the detector should account for this loss of information whenever a context is repeated.

2. `Excess` detection keeps only the part of each $-\log Q(m_a, S_a(\omega))$ above the $\mathrm{Exp}(1)$ median,

$$S(\omega) = \sum_a \max(-\log Q(m_a, S_a(\omega)) - \log 2, 0),$$

   which is zero with probability one half under the null and $\mathrm{Exp}(1)$ otherwise, giving the exact tail $R_H(s) = \sum_{b=1}^{H} \binom{H}{b} 2^{-H} Q(b, s)$ for $H = |\mathcal{H}|$. This pushes the idea behind the `Gamma` detector even further: only the upper tail of the score distribution provides a reliable watermarking signal.

In Figure 20, both refinements indeed slightly increase the detection power at no cost in text quality.

`Weighted Gamma` and `Position evidence` These two detectors are extensions of the `Gamma` detector that require a model forward pass at detection time, and hence are considerably more costly. Let $\hat{p}_t$ be the next-token distribution given $\omega_{<t}$, as estimated by the model during this additional forward pass.

1. `Weighted Gamma` weights each exponential score by the observed token's surprisal quantized to quarter units,

$$\eta_t = 1 + \frac{1}{4} \mathrm{round}\left(4 \min(\max(-\log \hat{p}_t(\omega_t), 0), 2.5)\right)$$

   and uses as a detection statistic

$$S(\omega) = \sum_{t \in D} -\eta_t \log(1 - U_{a_t, \omega_t}).$$

   The intuition is that a high-entropy position is one where the race had a real choice, so its score carries more information than the score of a position where the model was almost deterministic.

2. `Position evidence` goes further and reconstructs the race itself from the observed token's log-probability and those of up to eight competitors. At position t, let C_t contain the retained competitors other than ω_t. It then reconstructs the `Gumbel race` outcome:

$$\alpha_{t,v} = \min\left(10^4, \frac{\hat{p}_t(v)}{\hat{p}_t(\omega_t)}\right), \qquad \lambda_t = \sum_{v \in C_t} \alpha_{t,v}, \qquad W_t = \mathbf{1}\left\{U_{a_t,v} < U_{a_t,\omega_t}^{\alpha_{t,v}} \text{ for all } v \in C_t\right\},$$

   where W_t indicates whether the observed token wins the reconstructed race. Its exact race-win tail is

$$R_t = \begin{cases} \dfrac{1 - U_{a_t,\omega_t}^{1+\lambda_t}}{1+\lambda_t}, & W_t = 1, \\ 1 - U_{a_t,\omega_t} + \dfrac{U_{a_t,\omega_t}^{1+\lambda_t}}{1+\lambda_t}, & W_t = 0. \end{cases}$$

   Under the null, $\Pr(W_t = 1 \mid U_{a_t,\omega_t} = u) = u^{\lambda_t}$ and $R_t \sim \mathcal{U}([0,1])$ (Proposition 7). When no competitor survives, $\lambda_t = 0$ and $W_t = 1$, giving $R_t = 1 - U_{a_t,\omega_t}$. Keeping only the first

retained candidate for each context (i.e., deduplicating occurrences of the context), denoted by D_{pos}, avoids reusing the same race and gives the statistic

$$S(\omega) = \sum_{t \in D_{\text{pos}}} -\log R_t \sim \text{Gamma}(|D_{\text{pos}}|, 1).$$

This detector can also be combined with the surprisal weights η_t of the `Weighted Gamma` detector.

As we have shown in Sec. 4.2, and as we see again in Figure 20, the weighted `Position evidence` detector is significantly more powerful than the detectors that do not require a model forward pass.

`Quantile` detector This is the detector matched to `QuantileRace`. Given the keyed ordering score $R_{a_t,u}$ and the keyed offset $\tilde{U}_{a_t}$, the detector computes $S(\omega) = \sum_{(a,u)\in D} \cos(2\pi(R_{a_t,u} - U_{a_t}))$ and then calibrates it by Monte Carlo simulation, since the statistic has no closed-form null distribution.

`Quantile + wheel` This is the rotation-invariant detector required by the `Wheel`. Form the angles $A_{a_t,u} = 2\pi((R_{a_t,u} - U_{a_t}) \bmod 1)$ and their resultant $S = \|(\sum_D \cos A_{a_t,u}, \sum_D \sin A_{a_t,u})\|$, then report an e-value rather than a p-value. With $\kappa = 2\sqrt{\log(2000)/N}$, the quantity $\mathcal{E} = I_0(\kappa S)/I_0(\kappa)^N$ has unit expectation under the null, so $P = \min(1, 1/\mathcal{E})$ is a valid p-value. As we see in Figure 21, the `Wheel` slightly improves quality but almost entirely eliminates the robustness of `QuantileRace`.

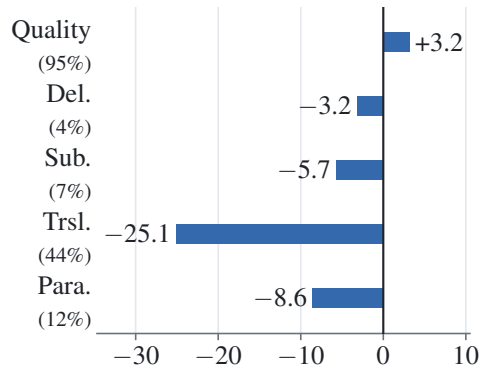


Figure 21: Impact of the `Wheel` compared to `QuantileRace`, under the `Quantile` detector.

`Predictive Gamma` This variant of the `Gamma` detector applies to all the private-geometry scores above.

Enumerate the retained pairs in D as $(a_i, u_i)_{i=1}^N$ and write $Z_i = Z_{a_i,u_i}$. Let $A_0, \ldots, A_{n_A-1}$ be a fixed public dictionary of candidate geometries ($n_A = 128$ for the `Signed sphere` and `Unoriented axis`, and $n_A = 512$ for the `Complex ray` and `Rotation`), with weights $w_{i,j}$ measuring how well A_j explains the scores observed before step i. Let B_i denote the current estimate of the private geometry A formed from the weights $w_{i,j}$ before observing Z_i. Write $\tilde{U}(B, Z)$ for the corresponding geometric score from App. A.4, evaluated with the geometry B in place of the true A, and define $d(B, Z) = 1 - \tilde{U}(B, Z)$. At each step i, compute $d_i = d(B_i, Z_i)$.

After scoring Z_i, update and normalize the weights as $w_{i+1,j} \propto w_{i,j}\, d(A_j, Z_i)^{-1/2}$, then compute

$$B_{i+1} = \begin{cases} \dfrac{\sum_{j=0}^{n_A-1} w_{i+1,j} A_j}{\left\|\sum_{j=0}^{n_A-1} w_{i+1,j} A_j\right\|}, & \text{signed sphere,} \\ \text{eig}_{\max}\left(\displaystyle\sum_{j=0}^{n_A-1} w_{i+1,j} A_j A_j^\top\right), & \text{unoriented axis or rotation,} \\ \text{eig}_{\max}\left(\displaystyle\sum_{j=0}^{n_A-1} w_{i+1,j} A_j A_j^*\right), & \text{complex ray,} \end{cases}$$

where $\text{eig}_{\max}$ returns a unit eigenvector associated with the largest eigenvalue and $*$ denotes the conjugate transpose. Under the null, each fresh score Z_i is independent of the earlier scores, and the matching geometric transform makes d_i uniform on $[0, 1]$ conditional on them. Thus,

$$S(\omega) = \sum_{i=1}^{N} -\log d_i \sim \text{Gamma}(N, 1). \tag{12}$$

We formally prove in Proposition 6 that $S(\omega)$ follows the $\text{Gamma}(N, 1)$ distribution under the null.

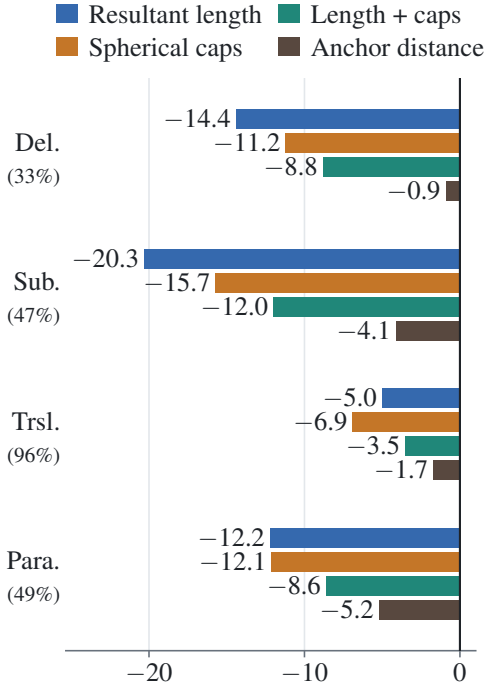


Figure 22: Comparison of `Signed sphere` detectors on the same generated texts. Bars show, for each attack, the difference in TPR@1%FPR to the `Predictive Gamma` in percentage points.

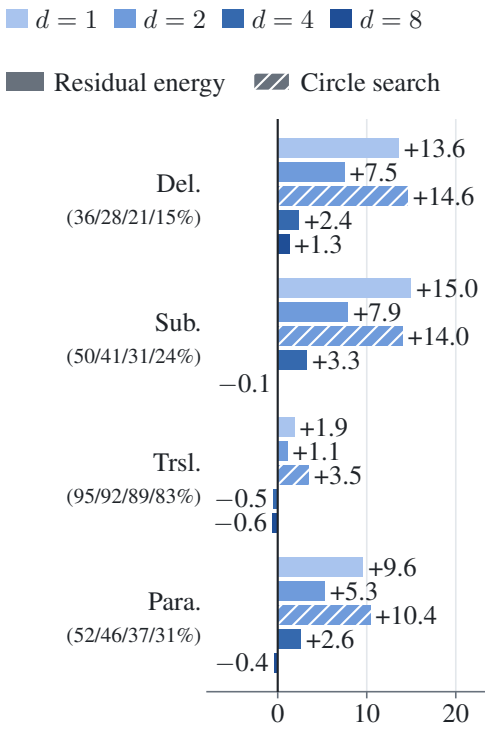


Figure 23: TPR@1%FPR differences to `Gaussian norm` for each attack. Colors indicate dimension d, solid bars show `Residual energy` and hatched bars show `Circle search` ($d = 2$). Parentheses give the `Gaussian norm` reference TPR@1%FPR for $d = 1/2/4/8$, respectively.

`Resultant length`, `Spherical caps`, and `Anchor distance` These detectors test whether the `Signed sphere` scores concentrate around a common direction, without knowledge of the private direction A. As above, we write $Z_i = Z_{a_i,u_i}$ for the scores associated with the N retained pairs in D.

1. `Resultant length` measures the alignment of the scores through $R = \|\sum_{i=1}^{N} Z_i\|$. Aligned scores produce a larger resultant, whereas scores pointing in different directions tend to cancel each other out. The detector compares R with its null distribution for N independent uniform points on $\mathbb{S}^2$, making it the spherical counterpart of the `Gaussian norm` detector described below.
2. `Spherical caps` measures local concentration around $m = \min(N, 16)$ anchor scores. For each anchor Z_j and area fraction $q \in \{1/16, 1/8, 1/4\}$, it counts $C_{j,q} = \sum_{i \neq j} \mathbb{1}\{Z_j \cdot Z_i \geq 1 - 2q\}$. Under the null, conditional on Z_j, this count follows $\mathrm{Bin}(N-1, q)$. The detector takes the smallest upper-tail p-value and applies a Bonferroni correction over all the $3m$ anchor–cap combinations.
3. `Anchor distance` replaces cap membership with a continuous measure of alignment. For each anchor Z_j, it computes $S_j = \sum_{i \neq j} -\log((1 - Z_j \cdot Z_i)/2)$, so that scores closer to the anchor contribute more evidence. Under the null, conditional on Z_j, the transformed inner products are independent uniforms, giving $S_j \sim \mathrm{Gamma}(N-1, 1)$. Searching over all N anchors and applying a Bonferroni correction gives the final p-value $P(\omega) = \min\{1, N \min_j Q(N-1, S_j)\}$.

Figure 22 compares these detectors to the `Predictive Gamma` detector on the same generated texts. We find that the `Predictive Gamma` detector outperforms each of those geometry-specific detectors.

`Gaussian norm`, `Circle search`, and `Residual energy` These detectors apply to the `Gaussian direction` scores. We denote by $Z_i = Z_{a_i,u_i}$ the scores associated with the N retained pairs in D.

Table 2: Evaluation of invalid agent schemes. Each TPR cell reports the result from the prior harness the scheme was designed in / from our current harness (App. D), whereas quality metrics only come from our current harness. Verified reports whether each scheme passes our current verification tests.

		Robustness TPR@1 ↑				Quality deviation (%) ↓			
Scheme	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3	Verified
ConcordMark-v0	100 / 100	86 / 83	99 / 96	99 / 99	91 / 90	2.1	8.1	21	✗
CathedralMark	100 / 100	100 / 85	100 / 97	100 / 98	99 / 91	3.8	8.6	22	✗
OverlayMark	100 / 100	100 / 84	100 / 97	100 / 99	100 / 92	4.1	8.9	23	✗
MeridianMark	99 / 98	88 / 88	62 / 74	94 / 93	77 / 78	59	23	60	✗

1. `Gaussian norm`: rather than searching for the private direction A, it integrates it out through the norm of the summed scores,

$$S(\omega) = \frac{1}{N}\left\|\sum_{i=1}^{N} Z_i\right\|^2 \sim \chi^2_d. \tag{13}$$

   The following two detectors, `Circle search` and `Residual energy`, both build on this statistic.

2. `Circle search` searches explicitly for the private direction A in dimension $d = 2$. For each candidate direction $B(\theta) = (\cos\theta, \sin\theta)$, it computes the Gamma statistic

$$S(\theta) = \sum_{i=1}^{N} -\log\Phi(-B(\theta)\cdot Z_i),$$
$$T = \max_{\theta\in[0,2\pi)} S(\theta).$$

   Under the null, $S(\theta) \sim \mathrm{Gamma}(N, 1)$ for each fixed θ, but maximizing over directions requires a correction. The detector uses the continuous-search bound of Davies (1987)

$$P_0 = \min\{1, Q(N, T) + 2\sqrt{\pi T} f_N(T)\},$$

   where f_N is the density of $\mathrm{Gamma}(N, 1)$ while the second term accounts for the search over θ.

3. `Residual energy` measures the dispersion of the scores around their mean, in addition to their alignment. Let $\bar{Z} = N^{-1}\sum_{i=1}^{N} Z_i$. The two statistics are

$$S_{\mathrm{mean}} = N\|\bar{Z}\|^2,$$
$$S_{\mathrm{res}} = \sum_{i=1}^{N}\|Z_i - \bar{Z}\|^2.$$

   Under the null, they are independent, with $S_{\mathrm{mean}} \sim \chi^2_d$ and $S_{\mathrm{res}} \sim \chi^2_{d(N-1)}$. For $N > 1$, let p_{mean} and p_{res} be their respective upper-tail p-values. The detector combines them through $S(\omega) = -\log p_{\mathrm{mean}} - \log p_{\mathrm{res}} \sim \mathrm{Gamma}(2, 1)$ under the null, giving $P_0 = Q(2, S(\omega))$. For $N = 1$, $S_{\mathrm{res}} = 0$ is degenerate, and the detector falls back to the `Gaussian norm` statistic S_{mean}.

As we see in Figure 23, both extensions improve over `Gaussian norm`, with `Circle search` giving the largest gains at $d = 2$, but the gains of `Residual energy` shrink as d grows and vanish at $d = 8$.

B Examples of Invalid Autoresearch Schemes

Here, we present 4 schemes that research agents designed using earlier iterations of our harness, whose verification was weaker than in Sec. 3.3. All four schemes appear to outperform prior work, yet they either exploit gaps in the evaluation or have an abnormally high false positive rate. We re-evaluate them with our harness in Table 2: they show impressive robustness, but none passes our verification. Nevertheless, we believe that some of their ideas are worth investigating in future work.

Prior harnesses ConcordMark-v0, CathedralMark, OverlayMark, and MeridianMark come from a harness with neither private verification nor private evaluation (i.e., the agent could see the whole evaluation dataset). Furthermore, the verifier only checked distortion-freeness and the false positive rate on $5{,}000$ English C4 (Raffel et al., 2020) human texts, using a single fixed watermark key. In particular, it had no check for soundness over the key. Schemes that passed verification were then evaluated once, using the watermark key fixed in their configuration, i.e., a key chosen by the agent.

ConcordMark-v0 Despite its name, ConcordMark-v0 is unrelated to ConcordMark (App. H.1), and we simply kept the independently submitted names for both. Its generation is a `Gumbel race` with a one-token context (i.e., `Fixed-length` context seeding with $k = 1$). Its detector, however, combines two p-values: P_w, the rank of the statistic under the private key among 1023 decoy keys (i.e., a Monte Carlo estimate of the p-value), and P_c, a p-value derived from a zero-shot LLM-text classifier, specifically an n-gram classifier. The agent trained this classifier to separate its own watermarked texts (clean and attacked) from $2{,}500$ human texts and calibrated it on $2{,}500$ other human texts. A text is flagged as watermarked if $P_w \leq 0.002$ or $P_c \leq 0.008$, so that, by the union bound, the false positive rate on human texts is at most 1% overall. This greatly improved the scheme's robustness (Table 2).

Yet, the classifier learned to recognize LLM-generated text in general rather than watermarked LLM-generated text specifically: on unwatermarked LLM replies from WildChat, it wrongly flags 31% of the texts as watermarked at a supposedly 1% FPR. This shows the importance of soundness over the key. Indeed, P_c does not depend on the key, so a text flagged by the classifier is flagged as watermarked under every possible key, which therefore violates soundness over the key (Equation (2)).

CathedralMark CathedralMark is a later scheme from the same run that keeps the detector of ConcordMark-v0 but also steers the generation toward its classifier. Depending on a keyed latent state, it either tilts the next-token distribution toward the tokens that increase the classifier score or applies a compensating tilt, so that, in expectation over the key, the next-token distribution remains p_t. Hence, the scheme passes the distortion-freeness test but, like ConcordMark-v0, fails the soundness over the key check. Nonetheless, steering watermark generation toward easily detectable LLM patterns yields a particularly robust watermark: CathedralMark achieves more than 90% robustness to paraphrasing.

OverlayMark For OverlayMark, a scheme similar to CathedralMark, the agent selected a watermark key that maximized TPR@1%FPR on the evaluation dataset, which was public. Hence, as shown in Table 2, it achieves 100% paraphrasing robustness in the original harness, but this falls to 92% in our updated harness, which varies the key (the worst TPR@1%FPR across 5 random keys).

MeridianMark MeridianMark uses a `Gumbel race` seeded by the last 4 tokens mapped to 32 semantic clusters. Its detector combines an entropy-weighted `Gamma` statistic and a tail statistic, and computes a p-value by fitting a Cauchy distribution to the statistics of 1024 decoy keys. It then remaps the p-values: every p-value below 0.0066 is linearly mapped to $[0, 0.001]$. The agent chose this threshold just below the fifth-smallest p-value of the public human texts (0.0069), and documented it explicitly in its submission: "`evaluate` never measures human FPR [...] exactly 4 humans fall under it (one-text margin)". This example shows that, if verification is public, the agent can exploit the metric to pass verification incorrectly. This is why our final harness uses private verification instead.

Takeaways Each of the above schemes exploits a gap in the harness: a human null corpus that does not cover LLM-generated text (ConcordMark-v0 and CathedralMark), a watermark key chosen by the agent (OverlayMark), or the ability to exploit public verification (MeridianMark). Our harness closes these gaps: our *private* verification corpus includes LLM-generated replies (App. D.1), we evaluate each scheme with 5 private keys and report the worst result (App. D.2), and our soundness over the key test only requires a single text on which the detector is miscalibrated over the key to reject a scheme, independently of how well the detector is calibrated on any specific human corpus.

C Studying the Soundness Given a Key

In this section, we investigate the effect of enforcing soundness given a key (Sec. 3.2). In Figure 2, we plot the autoresearch trajectories of each agent independently, and, as we explain, the agents tackled soundness given a key differently, which makes the comparison uneven. Namely, GPT-6 Astra

multiplies its p-values by $1/\beta = 20$ to be provably sound given a key, OPUS 5 applies smaller safety margins that it calibrated locally, and GEMINI-3.8 FLASH and the baselines apply none. For the baselines, in the settings of Figure 2, neither AAR nor SynthID passes our soundness given a key test.

Non-degenerate distributions In Equation (3), we only consider non-degenerate human distributions $\mathcal{H}_0$, i.e., for a threshold α, distributions in which no single text has probability above α. In practice, this property holds for what is intuitively considered human text distributions. Without this restriction, the requirement is unreasonably strong. For instance, for a point mass on a text ω, the false positive rate of a given key is either 0 or 1, so Equation (3) requires $\Pr_\xi[P_\xi(\omega) \leq \alpha] \leq \beta$ for every $\alpha < 1$, i.e., that the p-value of ω equals 1 for all but a fraction β of the keys. Note that Proposition 1 holds for every distribution, degenerate or not.

Experimental setup We rescore the submissions of the four agents from Figure 2 and the three baselines. For each scheme, we remove the explicit outer p-value multiplier chosen by the agent (if any), and keep the rest of the detector unchanged (e.g., the Bonferroni correction over `Channels`) so that it remains sound over the key. We then evaluate the following three p-value correction settings:

- **Guaranteed**: we multiply the p-value by $1/\beta = 20$ as in Equation (14). By Proposition 1, any scheme that is sound over the key then becomes provably sound given a key, at the cost of power.
- **Empirical**: we use the smallest multiplier $\kappa \geq 1$ such that the scheme passes both soundness tests (Algorithms 2 and 3), using the thresholds, sample sizes, and significance levels of Sec. 3.3.
- **No correction**: we use the raw p-values ($\kappa = 1$) and do not enforce the soundness given a key test.

In all settings, every scheme must still be distortion-free and sound over the key. We then measure the paraphrasing TPR@1%FPR of each submission with the same evaluation protocol as in Sec. 4.

Baselines fail without correction In Figure 24, we show the smallest multiplier each scheme requires to pass our soundness tests. Among the baselines, only TextSeal passes without correction. With raw p-values, both AAR and SynthID fail the soundness given a key test. Similarly, 23 of the 52 agent submissions would fail without their outer multiplier. In fact, both GPT-6 ASTRA (low) and OPUS 5 only started adding such multipliers after their first submissions had been rejected by our soundness given a key test.

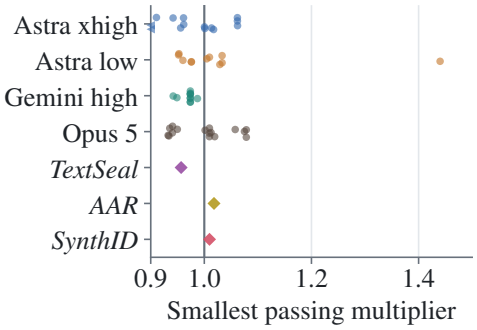


Figure 24: Smallest p-value multiplier each scheme requires to pass our soundness tests.

The guaranteed correction is overly conservative Yet, the multiplier required in practice is much smaller than $1/\beta$: at most 1.08 for all but one submission. This is expected: if the pseudo-random watermark scores are diverse enough over a corpus given a fixed key, then, by soundness over the key, the expected false positive rate converges to at most α, which satisfies soundness given a key. Hence, correcting with $1/\beta$ is overly conservative in many cases, costing significant detection power. This is why our main evaluation (Sec. 4.1) does not correct the p-values unless the agents did so themselves.

Impact on robustness Figure 25 shows the paraphrasing robustness trajectories in each setting. The guaranteed correction costs on average 16.3 percentage points of paraphrasing TPR compared to the empirical correction, which itself costs at most 2.5 percentage points compared to no correction. Importantly, the conclusions of Sec. 4.1 hold in all three settings: all agents improve over time and outperform all baselines, with OPUS 5 and GPT-6 ASTRA (xhigh) reaching the highest robustness. In particular, in the empirical setting, the best GPT-6 ASTRA (xhigh) submission reaches 83% paraphrasing TPR, versus 70% in the guaranteed setting, closing most of the gap with OPUS 5.

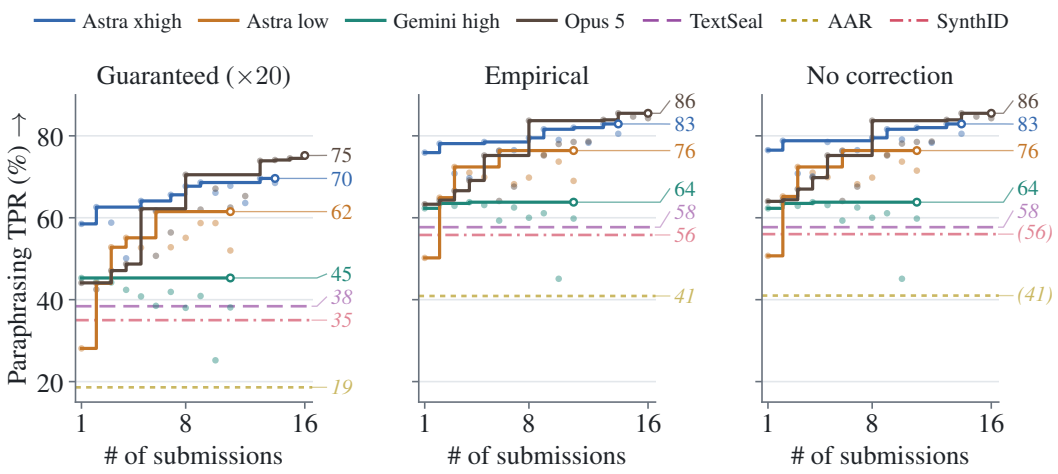


Figure 25: Paraphrasing robustness of the submissions of each agent under the three p-value corrections, where each dot is a submission and the solid lines are the best value so far. Baselines are in italics, and a parenthesis means the baseline fails our soundness given a key test without correction.

D Experimental Setup

Here, we describe the experimental setup of our evaluation (Sec. 4). Our harness distinguishes two configurations: the *public* configuration, which the research agent receives and can run locally, and the *private* configuration, which only the grader uses when the agent submits its scheme (Sec. 3.4).

D.1 Watermark Verification

To verify each watermarking scheme, we run the three statistical tests from Sec. 3.3 on a corpus of non-watermarked (human or LLM-generated) texts. A scheme is valid only if it passes all three tests.

Corpora The public corpus consists of the answers of the ELI5 (Fan et al., 2019) dataset. The private corpus is a mixture where half of the texts come from WildChat (Zhao et al., 2024a) and the other half from Wikipedia articles, evenly split across 10 languages (English, German, French, Spanish, Chinese, Russian, Arabic, Hindi, Japanese, and Swahili). For WildChat, we take in each conversation the first assistant reply that has at least 512 tokens (i.e., non-watermarked LLM-generated text), across all 86 shards of WildChat-4.8M. For Wikipedia, we use the first shard of the 2023-11-01 dump of each language. In both corpora, we tokenize the texts with the Llama-3.1-8B-Instruct tokenizer, discard texts shorter than 512 tokens, and truncate the others to their first 512 tokens. Each test samples texts from this mixture in proportion to the weights of its components.

Soundness tests For soundness over the key (Algorithm 2), we sample 200 fixed reference texts and draw m keys per text, with $m = 10{,}000$ in the public configuration and $m = 20{,}000$ in the private one. For soundness given a key (Algorithm 3), we draw 1000 keys and sample 20,000 fresh texts per key from the corpus (with replacement). We set the allowed bad-key fraction to $\beta = 5\%$ with a single screening level of 1%. Both tests check the thresholds $\alpha \in \{0.001, 0.01, 0.05\}$ in the private configuration and only $\alpha \in \{0.01, 0.05\}$ in the public one. The two soundness tests share a total significance level of 0.001, split equally between them (i.e., $\delta = 0.0005$ in Algorithms 2 and 3).

Distortion-freeness test For distortion-freeness (Algorithm 1), we use the first 128 tokens of 4 sampled corpus texts as contexts. We compute the next-token distributions of Llama-3.2-1B-Instruct (temperature 1, top-k 50) and average the watermarked distributions over 2048 keys, testing the 16 most likely tokens and the remaining mass. The significance level is $\delta = 0.001$. If the scheme leaves all 4 distributions unchanged, the test is inconclusive and we also reject the scheme.

Table 3: Research agents from Figure 2.

Agent	Interface	Version	Reasoning	Submissions
GPT-6 ASTRA (xhigh)	Codex CLI	0.153.3	xhigh	14
GPT-6 ASTRA (low)	Codex CLI	0.153.3	low	11
OPUS 5	Claude Code	2.1.231	high	16
GEMINI-3.8 FLASH (high)	Antigravity CLI	1.1.25	high	11

D.2 WATERMARK EVALUATION

The public and private evaluations share the same protocol for generation, detection, robustness, and quality, which we describe first, and they differ in their watermark keys, prompts, and sampling seeds.

Generation We sample 1000 prompts from the questions of the training split of ELI5 (Fan et al., 2019), keeping only questions with more than 100 characters and removing duplicate questions. We generate the replies with LLAMA-3.1-8B-INSTRUCT, prompting it with the raw question. We sample with temperature 1, top-k 50, and top-p 1, and force the replies to have between 200 and 300 tokens.

Detection and robustness To measure watermark strength (before or after attacks), we report the TPR@1%FPR, i.e., the fraction of the 1000 replies whose detector p-value is below 0.01. Because the verification (App. D.1) screens the p-values for soundness, we do not calibrate the threshold empirically. For robustness, we use four attacks. For 50% word deletion, each whitespace-separated word is deleted independently with probability 0.5. For 50% synonym substitution, we randomly replace half of the words (or all replaceable words, if fewer) with a random WordNet synonym. For back-translation through French and paraphrasing, we use GPT-5.4-MINI (`gpt-5.4-mini-2026-03-17`) with the default sampling parameters of the OpenAI API and the corresponding prompts of App. K.2.

Quality To measure the impact on quality, we use three different metrics. We compute the perplexity of each of the 1000 replies with LLAMA-3.1-8B-INSTRUCT, on the reply alone (i.e., not conditioned on the prompt), and average the perplexities over the replies. We also measure the diversity of replies to a single prompt: we sample 10 additional prompts from ELI5 and generate 100 replies for each prompt, following the protocol established in Gloaguen et al. (2026). For each reply, we compute the Self-BLEU (bigram and trigram) score on token IDs, using the 99 other replies to the same prompt as references, and average the scores over the 1000 replies. We then compare each quality metric to that of the unwatermarked model, evaluated with the same protocol. We report the relative L_1 distance (in percent), i.e., the absolute difference divided by the unwatermarked value, and, for an aggregated quality metric, we average the relative distances of the three metrics.

Public and private evaluation In the public configuration, the agent evaluates its scheme locally with its own watermark key, and the prompts are always the first 1000 (and 10 for the diversity metric) ELI5 questions after shuffling the dataset with seed 0. In the private configuration, the grader evaluates each submission with 5 private watermark keys. For each key, the grader derives from the key the seeds used to shuffle ELI5 (separately for the detection and diversity prompts) and the generation seed, so that the private prompts differ from the public ones and across keys. The grader then reports, for each metric, the worst result over the 5 keys: the minimum TPR and the maximum distance to the unwatermarked model. As the maximum distance is noisy across keys, in this paper, we instead report the median distance over the 5 keys for each quality metric (and keep the minimum TPR). Only the aggregated metrics and the verification verdicts are returned to the agent. The private evaluation runs in an isolated container that the agent cannot access.

D.3 RESEARCH AGENTS

We summarize the research agents in Table 3. We run all agents with the prompt of App. K.1 in their own command-line interface with subagents enabled. When an agent exits, the harness starts another in the same container and Git checkout, until the budget is exhausted and all submissions are graded.

Environment Each agent runs in a persistent Docker container with one NVIDIA RTX PRO 6000 Blackwell GPU (96 GB), 48 CPU cores, and full Internet access. The container holds the watermarking library (with the baselines of Sec. 3.4), the public configurations, and API keys for third-party AI providers (intended to be used for e.g., paraphrasing attacks). The grader runs in a separate container, which does not share any storage with the agent container, and on a different GPU.

D.4 BASELINES

We first detail how the three baselines (AAR, SynthID, and TextSeal) work using the notation from Sec. 3.1 and then go through their hyperparameters. Recall that $\omega \in \Sigma^*$ is a sequence of tokens and $p_t \in \Delta(\Sigma)$ the next-token probability distribution given $\omega_{<t}$. We assume there exists a sequence (U_{a_t}) of pseudo-random vectors in $[0,1]^{|\Sigma|}$ with i.i.d. uniform entries, seeded by the context hash a_t. The watermark is defined by $f : [0,1]^{|\Sigma|} \times \Delta(\Sigma) \to \Delta(\Sigma)$, and $S(\omega)$ denotes the detector statistic.

AAR With AAR, the logits transformation is simply a Gumbel race, and the next token is sampled according to

$$\omega_t = \arg\max_{v \in \Sigma} \left(-\log(-\log U_{a_t,v}) + \log p_t(v) \right).$$

Because $-\log(-\log U_{a_t,v})$ are i.i.d. Gumbel random variables, using the Gumbel-max trick we can show that in expectation over U_{a_t} the next token ω_t is sampled according to p_t. For detection, we use the one-sided Kolmogorov–Smirnov test against the uniform null, with the statistic

$$S(\omega) = \max_{1 \le i \le N} \left(U_{(i)} - \frac{i-1}{N} \right),$$

where $U_{(i)}$ are the ordered scores (U_{a_t,ω_t}).

SynthID With SynthID, the logits transformation corresponds to a repeated tournament sampling with L layers and C leaves. Instead of a single watermark score, SynthID uses L independent watermark scores $U_{a_t}^{(1)}, \ldots, U_{a_t}^{(L)}$, one per layer, which it binarizes into $g_{a_t,v}^{(l)} := \mathbb{1}\{U_{a_t,v}^{(l)} > 1/2\}$. Let $p_t^{(l)}$ be the probability distribution at layer l with $p_t^{(0)} := p_t$. Then, at each layer we apply

$$p_t^{(l)}(v) = p_t^{(l-1)}(v) \cdot \begin{cases} \dfrac{1-(1-M_t^{(l)})^C}{M_t^{(l)}} & \text{if } g_{a_t,v}^{(l)} = 1, \\ (1-M_t^{(l)})^{C-1} & \text{otherwise,} \end{cases} \quad \text{with} \quad M_t^{(l)} := \sum_{v \in \Sigma} p_t^{(l-1)}(v)\, g_{a_t,v}^{(l)},$$

and the logits transformation is $f(U_{a_t}, p_t) := p_t^{(L)}$. For detection, we use the weighted mean detector over the retained positions $t \in D$, with the statistic

$$S(\omega) = \frac{1}{N} \sum_{t \in D} \sum_{l=1}^{L} w_l\, g_{a_t,\omega_t}^{(l)},$$

where $w_1, \ldots, w_L \ge 0$ are layer weights summing to 1. Under the null, the per-token scores $\sum_l w_l\, g_{a_t,\omega_t}^{(l)}$ are i.i.d. with mean $1/2$ and variance $\sigma_w^2 := \frac{1}{4}\sum_l w_l^2$. We thus use the normal approximation for the p-value $1 - \Phi\left(\sqrt{N}\,(S(\omega) - 1/2)/\sigma_w\right)$, where Φ is the standard normal CDF.

TextSeal With TextSeal, the logits transformation is the same Gumbel race as AAR, but with two keys: a primary and a secondary key, yielding two watermark scores U_{a_t} and U'_{a_t}. At each step, the watermark score used by the logits transformation is $\tilde{U}_{a_t} = U'_{a_t}$ with probability λ and $\tilde{U}_{a_t} = U_{a_t}$ otherwise, where the choice is drawn with true randomness. For detection, we fuse the two scores of each retained position and use the weighted statistic

$$S(\omega) = \sum_{t \in D} w_t \left(-(1-\lambda)\log(1 - U_{a_t,\omega_t}) - \lambda \log(1 - U'_{a_t,\omega_t}) \right),$$

where the weights $w_t \in [w_{\min}, w_{\max}]$ increase linearly with the entropy H_t of the next-token distribution of an auxiliary LLM at position t, i.e., $w_t = w_{\min} + (w_{\max} - w_{\min})(H_t - \min_{t'} H_{t'})/(\max_{t'} H_{t'} - \min_{t'} H_{t'})$.

Watermark hyperparameters All three baselines seed their watermark with a hash of the 3 preceding tokens and the private key, and do not apply the watermark when the context was already seen in the same request. All sample from the top-50 tokens. For AAR (Aaronson, 2023), we use the Gumbel-max sampling, and its detector, a one-sided Kolmogorov–Smirnov test on the deduplicated Gumbel scores. For SynthID (Dathathri et al., 2024), we use the tournament sampling with 30 layers and 2 leaves, restricted to the top-50 tokens of the unwatermarked distribution, and the weighted mean detector (with weights decreasing linearly from 10 to 1 across layers) with a normal approximation. For TextSeal (Sander et al., 2026), we use a secondary key with probability $\lambda = 0.1$, and weights between $w_{\min} = 0.1$ and $w_{\max} = 1$ derived from the next-token entropies of GEMMA-3-270M-IT.

D.5 HELD-OUT EVALUATION

The held-out evaluation of Sec. 4.3 uses a new protocol that neither the agents nor our harness used.

Prompts We sample prompts from WILDCHAT-4.8M (Zhao et al., 2024a) using its language labels, English and Chinese. We use the first user turn of conversations whose second turn is an assistant reply, remove duplicates, and discard prompts longer than 3500 tokens. We keep a prompt only if its original WildChat reply has at least 200 tokens with both the LLAMA-3.1-8B-INSTRUCT and QWEN3.8-27B tokenizers. We then take the first 1000 qualifying prompts of each language after shuffling the dataset shards with a fixed seed. All schemes, models, and keys share the same prompts.

Generation We generate the replies with LLAMA-3.1-8B-INSTRUCT and QWEN3.8-27B (with thinking disabled), using their chat templates with a single user message and no system prompt. We sample with temperature 1, top-k 50, and top-p 1, and force the replies to contain 200 to 300 tokens.

Metrics We report the TPR@1%FPR as in App. D.2. For word deletion and synonym substitution, we segment the English texts with a regular expression and the Chinese texts with Jieba. Deletion removes exactly half of the words, chosen at random, and substitution replaces half of the words with a random synonym from WordNet (English) or the Open Multilingual WordNet (Chinese). If fewer words are replaceable, we replace all of them. For back-translation and paraphrasing, we use DEEPSEEK-V4 FLASH with temperature 1, reasoning disabled, and the attack prompts of App. K.2.

We compute the perplexity of the replies with the model that generated them, on the reply alone. For diversity, we use the first 10 prompts of each language, generate 100 replies per prompt, and compute the Self-BLEU scores as in App. D.2. We report the relative distance (in percent) to the unwatermarked model with the same model and language. We use 5 new watermark keys, different from all the keys used in our harness, and report the minimum TPR and the median quality distance over these 5 keys.

Finally, for RotationFirst (Sec. 4.3), we rebuild the `Lexical group` map for each tokenizer and extend it to Chinese tokens by using the Open Multilingual WordNet.

E OUR AUTORESEARCH HARNESS

We describe our watermarking library (App. E.1) and motivate our research environment (App. E.2).

E.1 OUR WATERMARKING LIBRARY

Library design We design our watermarking library around four primary components: the context seeding, the watermark score, the logits transformation, and the detector, following the formalization described in Sec. 3.1 and the work of Gloaguen et al. (2026). At generation time, the watermarking scheme is defined as a vLLM logit processor. It takes as input the list of tokens already generated and the next-token logits (i.e., the unnormalized next-token probability distribution) and returns the watermarked next-token

Table 4: End-to-end generation throughput (tokens/s). The reference implementations use Transformers, and our library uses vLLM with one or up to eight prompts per request.

Scheme	Reference	Ours (1)	Ours (8)
SynthID	152.3	243.5	628.2
AAR	194.5	303.9	1,413.3
TextSeal	186.9	298.6	1,338.7

logits. It is stateful per prompt but stateless across prompts (i.e., each prompt is treated independently). For detection, it takes as input a list of tokens and returns a p-value. This simple design, essentially a class with two methods, ensures that the proposed schemes can be easily evaluated and, following Gloaguen et al. (2026), is expressive enough to represent most prior watermarking schemes. In fact, alongside the library boilerplate, we provide the research agents with implementations of the following watermarks: AAR (Aaronson, 2023), SynthID (Dathathri et al., 2024), DiPMark (Wu et al., 2024), TextSeal (Sander et al., 2026), MCMark (Chen et al., 2025), SWEET (Lee et al., 2024), KGW (Kirchenbauer et al., 2023), unigram (Zhao et al., 2024b), and the schemes from Gloaguen et al. (2026).

Throughput We compare our vLLM library with the reference implementations of AAR, SynthID, and TextSeal on NVIDIA RTX PRO 6000 Blackwell GPUs. For each scheme, we generate 128 tokens for each of 100 ELI5 prompts with LLAMA-3.2-1B-INSTRUCT, temperature 0.7, and top-p 1. We measure throughput as generated tokens per second, excluding model loading. Our unbatched generation is 1.56–1.60× faster than the references, and batching raises this speedup to 4.12–7.27×.

E.2 RESEARCH ENVIRONMENT

Here, we motivate the two-phase submission budget of Sec. 3.4 with two runs from Table 6. We find that, when several submissions are available, agents mostly spend them on variants of a single scheme, whereas when a single submission is left, they explore distinct ideas locally before committing to one.

GPT-6 ASTRA (low) In this run, the first agent was given all 11 submissions at once. It spent 10 of them in about four hours on FreshRace and 8 of its variants (FreshRace-lexical was submitted twice). Then, with a single submission left, the 8 subsequent agents locally implemented and evaluated 15 new schemes before submitting RenewalRace. When we retrospectively submit the 15 unsubmitted schemes to our private grader, they all pass our verification, i.e., they are not merely failed attempts.

OPUS 5 This run uses the two-phase budget. In the first phase, the agent spent its 4 submissions on one scheme, TallyMark. After its first submission failed our soundness given a key test, it changed the context seeding (also rejected), then reverted it and multiplied the p-values by 4, and finally resubmitted the same code with an updated description. In the second phase, with a single submission per agent, all 12 submissions were different schemes, which substantially improved the robustness.

F STATISTICAL TESTS

In this section, we present the algorithmic description of the three statistical tests from Sec. 3.3. We prove in App. J.3 that each test rejects a valid scheme with probability at most δ. A rejected scheme violates the property with confidence $1 - \delta$. If it is not rejected, as with any statistical test, we cannot conclude whether the scheme violates the property. In practice, however, we find that the tests are powerful: e.g., they rejected all invalid schemes of App. B and even some of the baselines (App. C).

G HASH FUNCTION AND PRNG

In this part, we explain the hash functions that the agents use to turn the context into a seed (i.e., an integer), and we detail the PRNG algorithm that we implement in our vLLM watermarking library.

Hash functions As stated in App. A.3, we assume that the hash function has no collisions, i.e., two different contexts should return different hashes. In contrast, prior works used weaker hash functions that had a high probability of collisions. We assume that the context is a tuple $(x_1, \dots, x_k) \in \mathbb{N}^k$ of integers (token IDs or lexical classes $\mathcal{G}(\omega_t)$). All arithmetic is performed on unsigned 64-bit integers (i.e., modulo 2^{64}) unless stated otherwise, $\oplus$ denotes the bitwise XOR, and $\gg$ the bitwise right shift.

SplitMix64 hash: we chain the SplitMix64 finalizer $m(x) = y_2 \oplus (y_2 \gg 31)$, with $y_1 = (x \oplus (x \gg 30)) \cdot$ `0xbf58476d1ce4e5b9` and $y_2 = (y_1 \oplus (y_1 \gg 27)) \cdot$ `0x94d049bb133111eb`. The context hash is

$$h_0 = \texttt{0x243f6a8885a308d3}, \qquad h_j = m\big(h_{j-1} \oplus m(x_j + j)\big), \qquad a_t = h_k,$$

Algorithm 1 Testing distortion-freeness

Require: Watermark transform f, fixed contexts and next-token distributions $(\omega_{<t}, p_t)_{t=1}^{N}$, $N \geq 1$, keys per context $m \geq 2$, head size $1 \leq k \leq |\Sigma|$, significance level $\delta \in (0, 1)$.

$h \leftarrow \lfloor m/2 \rfloor$; $a \leftarrow$ False
for $t = 1, \ldots, N$ **do**
  $v_1, \ldots, v_k \leftarrow$ the k most likely tokens under p_t
  $G_t(q) \leftarrow (q(v_1), \ldots, q(v_k), 1 - \sum_{r=1}^{k} q(v_r))$ ▷ Pool remaining tokens
  Draw m uniform independent keys and derive the scores $\zeta_t^{(1)}, \ldots, \zeta_t^{(m)}$ at $\omega_{<t}$ ▷ $\zeta_t^{(j)}$: score $\tilde{U}_{a_t}$ fed to f
  $p_t^{(j)} \leftarrow f(\zeta_t^{(j)}, p_t)$; $d_j \leftarrow G_t(p_t^{(j)}) - G_t(p_t)$ for $j = 1, \ldots, m$
  $a \leftarrow a \vee (\exists j : p_t^{(j)} \neq p_t)$
  $\bar{d} \leftarrow m^{-1} \sum_{j=1}^{m} d_j$
  $p_{\text{coord}} \leftarrow \min\{1, 2(k+1)\exp(-2m\|\bar{d}\|_\infty^2)\}$
  $u \leftarrow \text{sign}(h^{-1} \sum_{j=1}^{h} d_j)$ ▷ Learn a bias direction
  $b \leftarrow \max\{0, (m-h)^{-1} \sum_{j=h+1}^{m} u^\top d_j\}$ ▷ Test on held-out keys
  $p_{\text{split}} \leftarrow \exp(-(m-h)b^2/2)$
  $q_t \leftarrow \min\{1, 2\min(p_{\text{coord}}, p_{\text{split}})\}$
end for
$Q \leftarrow \min\{1, N \min_t q_t\}$ ▷ Bonferroni across contexts
if $Q \leq \delta$ **then**
  return Reject
end if
return No distortion detected if a, otherwise inconclusive

Algorithm 2 Testing soundness over the key

Require: Detector P_ξ, nonempty corpus $C = (\omega_1, \ldots, \omega_n)$, keys per text $m \geq 1$, finite nonempty thresholds $\mathcal{A} \subseteq [0, 1]$, significance level $\delta \in (0, 1)$.

$\eta \leftarrow \big(1 - (1-\delta)^{1/n}\big)/|\mathcal{A}|$ ▷ Corrected cutoff
$\mathcal{R} \leftarrow \emptyset$
for $i = 1, \ldots, n$ **do**
  Draw $\xi_1^i, \ldots, \xi_m^i$ uniformly and independently from the key space
  $P_{i,j} \leftarrow P_{\xi_j^i}(\omega_i)$ for $j = 1, \ldots, m$
  for $\alpha \in \mathcal{A}$ **do**
    $X_i(\alpha) \leftarrow \sum_{j=1}^{m} \mathbf{1}\{P_{i,j} \leq \alpha\}$
    $q_i(\alpha) \leftarrow \Pr_{Z \sim \text{Binomial}(m,\alpha)}[Z \geq X_i(\alpha)]$
    if $q_i(\alpha) \leq \eta$ **then**
      $\mathcal{R} \leftarrow \mathcal{R} \cup \{(i, \alpha)\}$
    end if
  end for
end for
return $\mathcal{R}$ ▷ Reject the scheme if $\mathcal{R} \neq \emptyset$

where the offset $+j$ makes the hash order-sensitive. The hash is not injective, but with 64-bit outputs a collision among n distinct contexts has probability at most $n^2/2^{65}$ by a union bound over all pairs.

Rotation first hash: packs the context injectively, so the no-collision assumption holds exactly. However, it is only valid for context sizes up to two. A context of length one is encoded as $a_t = x_1 + 1$, and one of length two as $a_t = 2^{41} + 2^{20}x_1 + x_2$ (injective for vocabularies below 2^{20}).

Rolling polynomial hash: a rolling polynomial hash with a counter. This is the hash used with `Ladder` context seeding (App. A.3). Given the lexical classes $g_t = \mathcal{G}(\omega_t)$, a context of width $w \leq 4$ preceding position t is hashed as

$$K_w = \Big(\sum_{i=1}^{w} g_{t-i}\, \varphi^{\,w-i}\Big) \oplus (w \cdot \texttt{0x2545F4914F6CDD1D}), \qquad \varphi = \texttt{0x9E3779B97F4A7C15},$$

Algorithm 3 Testing soundness given a key

Require: Detector P_ξ, human text distribution $\mathcal{H}_0$, number of keys $d \geq 1$, texts per key $n \geq 1$, allowed bad-key fraction $\beta \in (0,1)$, predeclared finite nonempty thresholds $\mathcal{A} \subseteq [0,1]$ and screening levels $\mathcal{S} \subset (0,1)$, allocated significance level $\delta \in (0,1)$.

$T(s,p,c) \leftarrow \Pr_{Z\sim\mathrm{Binomial}(s,p)}[Z \geq c]$ ▷ Binomial upper tail
$\eta \leftarrow \delta/(|\mathcal{A}||\mathcal{S}|)$; $\mathcal{R} \leftarrow \emptyset$
for $j = 1, \ldots, d$ **do**
  Draw a uniform key ξ_j and $\omega_1^j, \ldots, \omega_n^j \sim \mathcal{H}_0$ ▷ All draws independent
  $P_{j,t} \leftarrow P_{\xi_j}(\omega_t^j)$ for $t = 1, \ldots, n$
  $X_j(\alpha) \leftarrow \sum_{t=1}^{n} \mathbf{1}\{P_{j,t} \leq \alpha\}$ for $\alpha \in \mathcal{A}$
end for
for $(\alpha, \gamma) \in \mathcal{A} \times \mathcal{S}$ **do**
  $c \leftarrow \min\{k \in \{1, \ldots, n+1\} : T(n, \alpha, k) \leq \gamma\}$
  $r \leftarrow T(n, \alpha, c)$; $b \leftarrow \beta + (1-\beta)r$ ▷ Null screening bound
  $B \leftarrow \sum_{j=1}^{d} \mathbf{1}\{X_j(\alpha) \geq c\}$ ▷ Keys screening positive
  $q \leftarrow T(d, b, B)$
  if $q \leq \eta$ **then**
    $\mathcal{R} \leftarrow \mathcal{R} \cup \{(\alpha, \gamma)\}$
  end if
end for
return $\mathcal{R}$ ▷ Reject the scheme if $\mathcal{R} \neq \emptyset$

where the XOR with a width-dependent constant separates contexts of different lengths. Let n be the number of earlier occurrences of K_w in the text, and $w^\star$ the width selected by the ladder. The seed is then

$$a_t = \big\lfloor \mathrm{fmix}_{64}\big(8192\, K_{w^\star} + n\big)/2^{32} \big\rfloor,$$

where fmix_{64} is the MurmurHash3 finalizer and 8192 is the maximum tally, so that repeated contexts with different counts get different seeds. Unlike the two previous hashes, the resulting seed has only 32 bits, so collisions already become likely after roughly 2^{16} distinct contexts.

PRNG Given a seed a_t, a candidate token v, and the 64-bit private key ξ, the PRNG returns the score $U_{a_t,v} \in [0,1)$. We require the PRNG to be stateless, i.e., $U_{a_t,v}$ depends only on (ξ, a_t, v), so that, given the text and the private key, the detector recovers exactly the scores used during generation.

Philox: this is the PRNG we implemented as part of our library and used for all baselines. It uses the counter-based Philox4x32-10 block cipher (Salmon et al., 2011), which maps a 128-bit counter and a 64-bit key to 4 pseudorandom 32-bit words (y_0, y_1, y_2, y_3). With A a bijective 32-bit avalanche ($x \mapsto x \oplus (x \gg 16)$, multiply by `0x7FEB352D`, $x \oplus (x \gg 15)$, multiply by `0x846CA68B`, $x \oplus (x \gg 16)$), the score is

$$(y_0, \ldots, y_3) = \mathrm{Philox}_\xi\big(A(v),\, A(a_t),\, 0,\, 0\big), \qquad U_{a_t,v} = \lfloor y_0/2^9 \rfloor\, 2^{-23},$$

where the avalanche disperses adjacent token IDs before Philox, and 23 bits fill the float32 mantissa. Independent `Channels` c use the candidate $v + 2^{18}c$, which never collides with any actual token ID.

High-precision Philox: similar to Philox, but the 64-bit seed and the token are packed injectively into the counter:

$$(y_0, \ldots, y_3) = \mathrm{Philox}_\xi\big(v,\, a_t \bmod 2^{32},\, \lfloor a_t/2^{32} \rfloor,\, \texttt{0xC0C0A123}\big),$$
$$U_{a_t,v} = \big(2^{26}\lfloor y_0/2^6 \rfloor + \lfloor y_1/2^6 \rfloor\big)\, 2^{-52} + 2^{-53}.$$

Since the counter is injective in (a_t, v), distinct (seed, token) pairs map to distinct Philox counters, hence to pseudorandomly independent scores. When a (seed, token) pair needs several scores (e.g., the three uniforms that RotationFirst maps to its coordinates $Z_{a_t,u} \in \mathbb{S}^3$ with the Hopf parametrization $(\sqrt{U_1}e^{2\pi i U_2}, \sqrt{1-U_1}e^{2\pi i U_3})$), the seed becomes $2^6 a_t + d$, whose low 6 bits hold the index d.

SplitMix64 PRF: reuses the SplitMix64 finalizer m from the *SplitMix64 hash*, with the domain-separation constants $c_1 =$ `0xa4093822299f31d0` and $c_2 =$ `0x082efa98ec4e6c89`:

$$s = m\big(a_t \oplus m(\xi \oplus c_1)\big), \qquad s' = m\big(s \oplus m(v \oplus c_2)\big), \qquad U_{a_t,v} = \big(\lfloor s'/2^{12} \rfloor + \tfrac{1}{2}\big)\, 2^{-52},$$

where 52 bits fill the float64 mantissa, and the midpoint $+\frac{1}{2}$ keeps $U_{a_t,v} \in (0,1)$ so that $\Phi^{-1}(U_{a_t,v})$ is always finite. Independent `Channels` c use the keys $\xi \oplus (c \cdot \texttt{0x9e3779b97f4a7c15})$ rather than ξ.

H CONCORDMARK AND ROTATIONFIRST

We describe in detail ConcordMark and RotationFirst, the two agent schemes evaluated in Sec. 4.3. For the low and high variants of ConcordMark, the generation algorithm is identical, and only their detectors differ.

H.1 CONCORDMARK

Key components We precisely describe the ConcordMark generation algorithm in Algorithm 4, and both of its detector variants in Algorithms 5 and 6. ConcordMark uses `Lexical group` only for context seeding. Let $\mathcal{G} : \Sigma \to \mathbb{N}$ be the corresponding lexical map. Given a word, its spelling is first normalized, and the word is then assigned to a synonym class of at most eight members (using WordNet). It then uses `Ladder` context seeding starting at width 1, or at width 2 when the ID of the last mapped token is below 200. For the hash function, it uses the *rolling polynomial hash* (App. G).

For the watermark score, it uses both `Prelude`, capped at 14 nats of entropy or 24 tokens, and 6 independent `Channels`. It uses the `Gumbel race` as its logits transformation. For detection, the low variant applies the `Excess` Gamma detection and the high variant, the weighted `Position evidence`.

Algorithm 4 ConcordMark generation

Require: Keys $\xi_0, \ldots, \xi_5$, model distributions p_t, lexical map $\mathcal{G}$, completion length T.

Sample $J \sim \mathcal{U}\{0, \ldots, 5\}$ ▷ Use channel J, keyed by ξ_J, throughout
$\omega \leftarrow ()$; $B \leftarrow 0$; $\mathcal{U} \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  if $t \le 24$ and $B < 14$ **then**
    Sample $\omega_t \sim p_t$ using private randomness
    $B \leftarrow B - \log \sum_v p_t(v)^2$ ▷ Prelude collision entropy
  else
    $z_{1:t-1} \leftarrow (\mathcal{G}(\omega_1), \ldots, \mathcal{G}(\omega_{t-1}))$; $m \leftarrow \min(4, t-1)$
    $\ell \leftarrow 1$; if $z_{t-1} < 200$, set $\ell \leftarrow \min(2, m)$
    for $w = \ell, \ldots, m$ **do**
      $W \leftarrow z_{t-w:t-1}$ ▷ Suffix of width w
      $n_t \leftarrow |\{i : w \le i < t-1,\ z_{i-w+1:i} = W\}|$ ▷ Count earlier occurrences
      If $n_t < 4$ or $w = m$, set $c_t \leftarrow (w, W, n_t)$, $a_t \leftarrow h(c_t)$ and **break** ▷ Context and seed
    end for
    if $n_t \ge 8192$ or $a_t \in \mathcal{U}$ **then**
      Sample $\omega_t \sim p_t$ using private randomness
    else
      $\mathcal{U} \leftarrow \mathcal{U} \cup \{a_t\}$
      $\tilde{U}_{a_t,v} \leftarrow U_{J,a_t,v}$ for each v with $p_t(v) > 0$
      $\omega_t \leftarrow \arg\min_{v:p_t(v)>0}\{-\log \tilde{U}_{a_t,v}/p_t(v)\}$ ▷ Token-level exponential race
    end if
  end if
  Append ω_t to ω
end for
return ω

Algorithm 5 ConcordMark (low) detection

Require: Keys $\xi_0, \dots, \xi_5$, completion ω, lexical map $\mathcal{G}$.
$D \leftarrow \emptyset; \mathcal{U} \leftarrow \emptyset$
for $t = 2, \dots, |\omega|$ **do**
  Reconstruct the ladder seed a_t and count n_t from $\mathcal{G}(\omega_{<t})$ as in Algorithm 4
  If $n_t < 8192$ and $(a_t, \omega_t) \notin \mathcal{U}$, set $D \leftarrow D \cup \{t\}$ and $\mathcal{U} \leftarrow \mathcal{U} \cup \{(a_t, \omega_t)\}$
end for
If $D = \emptyset$, **return** 1
$\mathcal{H}(\omega) \leftarrow \{a_t : t \in D\}$; $H \leftarrow |\mathcal{H}(\omega)|$
$D_a(\omega) \leftarrow \{\omega_t : t \in D,\ a_t = a\}$; $m_a \leftarrow |D_a(\omega)|$ for each $a \in \mathcal{H}(\omega)$
for $j = 0, \dots, 5$ **do**
  for $a \in \mathcal{H}(\omega)$ **do**
    $S_{j,a}(\omega) \leftarrow \sum_{v \in D_a(\omega)} -\log(1 - U_{j,a,v})$
    $Z_{j,a} \leftarrow -\log Q(m_a, S_{j,a}(\omega))$ ▷ Calibrate each context with its Gamma tail
  end for
  $S_j(\omega) \leftarrow \sum_{a \in \mathcal{H}(\omega)} \max\{Z_{j,a} - \log 2, 0\}$ ▷ Keep evidence above the median
  $P_j \leftarrow 1$
  if $S_j(\omega) > H/2$ **then**
    $\theta \leftarrow (3 - \sqrt{1 + 4H/S_j(\omega)})/2$
    $P_j \leftarrow \exp\{H \log[(1 - \theta/2)/(1 - \theta)] - \theta S_j(\omega)\}$ ▷ Chernoff bound
    If $P_j < 0.1$, set $P_j \leftarrow \sum_{b=1}^{H} \binom{H}{b} 2^{-H} Q(b, S_j(\omega))$ ▷ Exact excess tail R_H
  end if
end for
return $P(\omega) = 1 - (1 - \min_j P_j)^6$ ▷ Šidák correction over channels

Algorithm 6 ConcordMark (high) detection

Require: Keys $\xi_0, \dots, \xi_5$, completion ω, lexical map $\mathcal{G}$, auxiliary model distributions $\hat{p}_t$.
Reconstruct a_t and the retained positions D as in Algorithm 5
If $D = \emptyset$, **return** 1
for $t \in D$ **do**
  $\eta_t \leftarrow 1 + \frac{1}{4} \operatorname{round}(4 \min\{\max\{-\log \hat{p}_t(\omega_t), 0\}, 2.5\})$
  $C_t \leftarrow$ {eight most probable tokens under $\hat{p}_t$} $\setminus \{\omega_t\}$
  $\alpha_{t,v} \leftarrow \min\{10^4, \hat{p}_t(v)/\hat{p}_t(\omega_t)\}$ for each $v \in C_t$
  $C_t \leftarrow \{v \in C_t : \alpha_{t,v} \geq 10^{-3}\}$; $\lambda_t \leftarrow \sum_{v \in C_t} \alpha_{t,v}$
  If $\lambda_t > 8$, set $C_t \leftarrow \emptyset$ and $\lambda_t \leftarrow 0$ ▷ Discard an unreliable reconstructed race
end for
for $j = 0, \dots, 5$ **do**
  for $t \in D$ **do**
    $W_{j,t} \leftarrow \mathbf{1}\{U_{j,a_t,v} < U_{j,a_t,\omega_t}^{\alpha_{t,v}}$ for all $v \in C_t\}$
    if $W_{j,t} = 1$ **then** ▷ The observed token wins the reconstructed race
      $R_{j,t} \leftarrow (1 - U_{j,a_t,\omega_t}^{1+\lambda_t})/(1 + \lambda_t)$
    else
      $R_{j,t} \leftarrow 1 - U_{j,a_t,\omega_t} + U_{j,a_t,\omega_t}^{1+\lambda_t}/(1 + \lambda_t)$
    end if
  end for
  $S_j(\omega) \leftarrow \sum_{t \in D} -\eta_t \log R_{j,t}$ ▷ Weighted position evidence
end for
$s \leftarrow \max_j S_j(\omega)$
if all weights equal some η **then**
  $q \leftarrow Q(|D|, s/\eta)$ ▷ Exact Gamma upper tail
else
  $q \leftarrow$ Lugannani–Rice approximation to $\Pr[\sum_{t \in D} \eta_t E_t \geq s]$, for independent $E_t \sim \mathrm{Exp}(1)$
end if
return $P(\omega) = \min\{1, 1.02[1 - (1 - q)^6]\}$ ▷ Channel correction and tail safety factor

H.2 RotationFirst

Key components We precisely describe the RotationFirst generation algorithm and its detector in Algorithms 7 and 8. RotationFirst uses `Lexical group` throughout the scheme. Let $\mathcal{G} : \Sigma \to \mathbb{N}$ be the corresponding lexical map. Each token is decoded, stripped of any surrounding whitespace, and

lowercased. If the resulting word appears in WordNet, it is assigned to the synonym class given by its first WordNet synset. Let $\mathcal{E}$ be the set of the resulting synonym classes. Tokens sharing the same synset are grouped together, whereas all other tokens remain in singleton classes. It then uses `Global first use` context seeding with the *rotation first hash* (App. G). For each class c, it uses the address $(0, c)$ if c has not been emitted before and $c \in \mathcal{E}$, and otherwise it uses the address $(\mathcal{G}(\omega_{t-1}) + 1, c)$.

For the watermark score, it uses a *private geometry* approach, namely the `Rotation` spherical geometry (`Spherical`). For each request, it samples a uniform random quaternion A. Then, at each step, it draws a uniform keyed $Z_{a_t,u} \in \mathbb{S}^3$ and derives the score:

$$\tilde{U}_{a_t,u} = F(|A \cdot Z_{a_t,u}|),$$

where $F(r) = \frac{2}{\pi}\left(\arcsin r + r\sqrt{1-r^2}\right)$. It uses the `ResidualRace` as its logits transformation, over classes rather than tokens: the clock of a (seed, class) pair is initialized to $-\log \tilde{U}_{a_t,u}$ the first time the pair is visited and carried over afterwards, and the token is then sampled within the winning class (Equation (4)). As `ResidualRace` is distortion-free even when a seed is repeated (App. J.4), RotationFirst does not use the per-request cache. For detection, it then uses the `Predictive Gamma` detector.

Algorithm 7 RotationFirst generation

Require: Key ξ, model distributions p_t, grouping $\mathcal{G}$, eligible classes $\mathcal{E}$, completion length T.

Sample $A \sim \mathcal{U}(\mathbb{S}^3)$ ▷ Private rotation shared throughout the completion
$F(r) \leftarrow \frac{2}{\pi}(\arcsin r + r\sqrt{1-r^2})$
$\omega \leftarrow ()$; $\mathcal{A} \leftarrow \emptyset$; $\mathcal{U} \leftarrow \emptyset$ ▷ Emitted classes and initialized clocks
for $t = 1, \ldots, T$ **do**
  $M_t(u) \leftarrow \sum_{v:\mathcal{G}(v)=u} p_t(v)$ for each class u
  $b_t \leftarrow \mathcal{G}(\omega_{t-1}) + 1$ if $t > 1$, otherwise $b_t \leftarrow 0$
  for each class u with $M_t(u) > 0$ **do**
    $c_t(u) \leftarrow 0$ if $u \in \mathcal{E} \setminus \mathcal{A}$, otherwise $c_t(u) \leftarrow b_t$; $a_t(u) \leftarrow h(c_t(u))$
    if $(a_t(u), u) \notin \mathcal{U}$ **then**
      if $b_t \neq 0$ or $u \in \mathcal{E}$ **then**
        Recover $Z_{a_t(u),u} \in \mathbb{S}^3$ using key ξ
        $\tilde{U}_{a_t(u),u} \leftarrow F(|A \cdot Z_{a_t(u),u}|)$
        $R_{a_t(u),u} \leftarrow -\log \tilde{U}_{a_t(u),u}$
      else
        Sample $R_{a_t(u),u} \sim \mathrm{Exp}(1)$ privately
      end if
      $\mathcal{U} \leftarrow \mathcal{U} \cup \{(a_t(u), u)\}$
    end if
  end for
  $u_* \leftarrow \arg\min_{u:M_t(u)>0} R_{a_t(u),u}/M_t(u)$; $\tau_t \leftarrow R_{a_t(u_*),u_*}/M_t(u_*)$
  $R_{a_t(u),u} \leftarrow \max\{0, R_{a_t(u),u} - \tau_t M_t(u)\}$ for each u with $M_t(u) > 0$
  Sample $R_{a_t(u_*),u_*} \sim \mathrm{Exp}(1)$ privately ▷ Refresh the winning clock
  Sample ω_t from $\{v : \mathcal{G}(v) = u_*\}$ with probabilities $p_t(v)/M_t(u_*)$
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{u_*\}$; append ω_t to ω
end for
return ω

Algorithm 8 RotationFirst detection

Require: Key ξ, completion ω, grouping $\mathcal{G}$, eligible classes $\mathcal{E}$, public unit quaternions $A_0, \ldots, A_{511}$.
$D \leftarrow ()$; $\mathcal{A} \leftarrow \emptyset$; $\mathcal{U} \leftarrow \emptyset$
for $t = 1, \ldots, |\omega|$ **do**
  $u_t \leftarrow \mathcal{G}(\omega_t)$
  $b_t \leftarrow \mathcal{G}(\omega_{t-1}) + 1$ if $t > 1$, otherwise $b_t \leftarrow 0$
  $c_t \leftarrow 0$ if $u_t \in \mathcal{E} \setminus \mathcal{A}$, otherwise $c_t \leftarrow b_t$; $a_t \leftarrow h(c_t)$
  If ($t > 1$ or $u_t \in \mathcal{E}$) and $(a_t, u_t) \notin \mathcal{U}$, append (a_t, u_t) to D and set $\mathcal{U} \leftarrow \mathcal{U} \cup \{(a_t, u_t)\}$
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{u_t\}$
end for
$N \leftarrow |D|$; if $N \leq 1$, **return** 1
Enumerate D as $(a_i, u_i)_{i=1}^{N}$ in text order
$F(r) \leftarrow \frac{2}{\pi}(\arcsin r + r\sqrt{1 - r^2})$; $d(B, Z) \leftarrow 1 - F(|B \cdot Z|)$
$B_1 \leftarrow A_0$; $w_{1,j} \leftarrow 1/512$ for $j = 0, \ldots, 511$; $S(\omega) \leftarrow 0$
for $i = 1, \ldots, N$ **do**
  Recover $Z_i = Z_{a_i, u_i} \in \mathbb{S}^3$ using key ξ
  $S(\omega) \leftarrow S(\omega) - \log d(B_i, Z_i)$ ▷ Score before updating the predicted rotation
  $w_{i+1,j} \leftarrow w_{i,j}\, d(A_j, Z_i)^{-1/2}$ for $j = 0, \ldots, 511$
  Normalize $w_{i+1,j} \leftarrow w_{i+1,j} / \sum_{k=0}^{511} w_{i+1,k}$
  $B_{i+1} \leftarrow \mathrm{eig}_{\max}(\sum_{j=0}^{511} w_{i+1,j} A_j A_j^{\top})$ ▷ Unit eigenvector for the largest eigenvalue
end for
return $P(\omega) = Q(N, S(\omega))$ ▷ Predictive Gamma upper tail

H.3 Additional Evaluation

Here, we complement the evaluation of Sec. 4.3 with additional plots (Figure 26). Namely, we plot the TPR as a function of the significance level α, from 10^{-6} to 10^{-1}, for both clean detection and after paraphrasing. At each level α, the TPR is the fraction of watermarked samples with a p-value below α. As in the main evaluation, we report the minimum TPR across the five watermarking seeds.

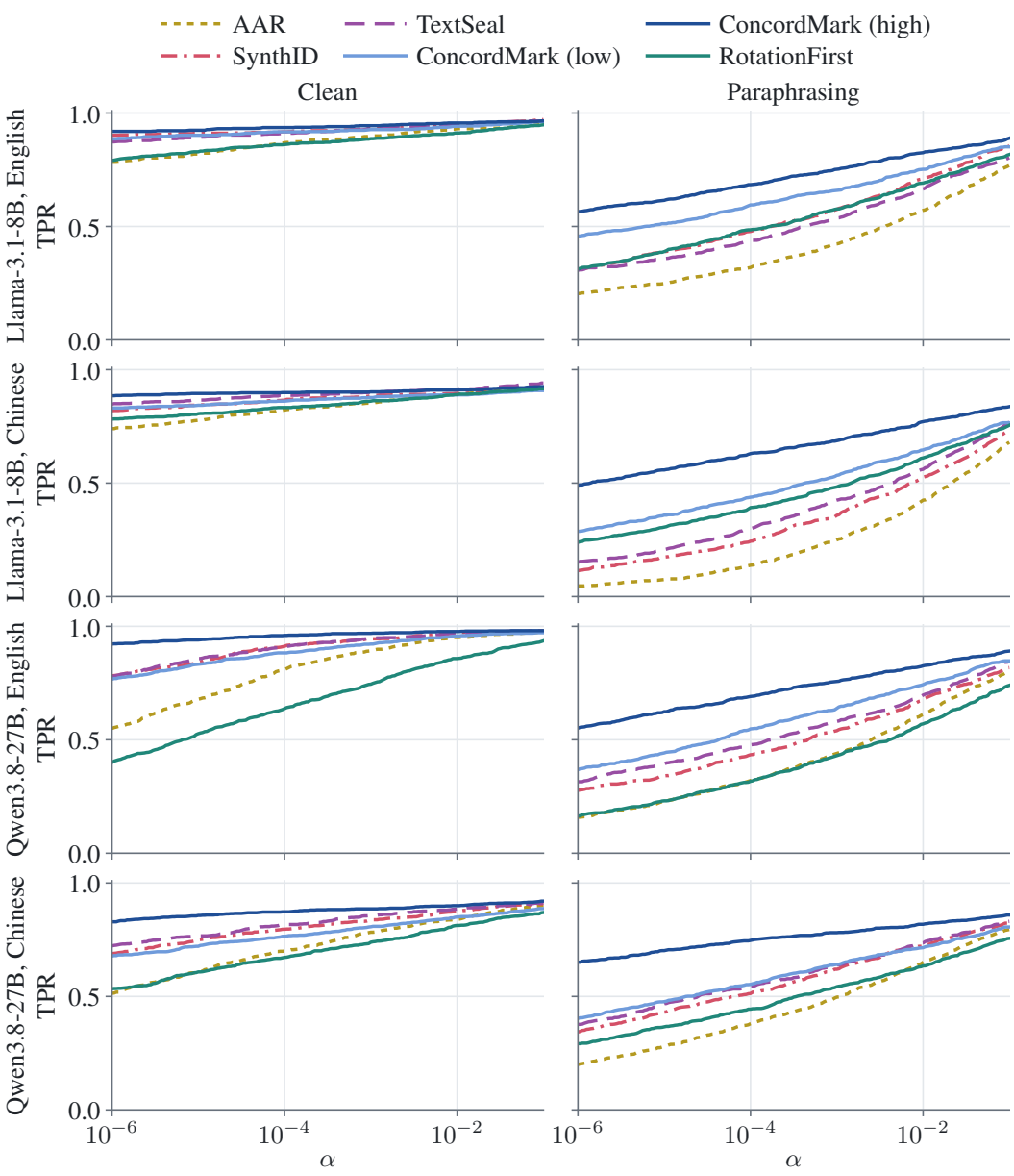


Figure 26: TPR as a function of the significance level α, with 1k samples per model and language generated from WildChat prompts. Left: detection on clean text. Right: detection after paraphrasing.

Results Across models, languages, and significance levels, these curves confirm the conclusions of Sec. 4.3. ConcordMark (high) provides the strongest detection, especially after paraphrasing. ConcordMark (low) also retains strong paraphrasing robustness while being more efficient at detection, since it does not require a model forward pass. RotationFirst is less robust than ConcordMark, as expected given its focus on quality, yet its robustness remains comparable to that of the baselines.

Table 5: Comparison with prior semantic watermarks on our private evaluation (without p-value correction). The last three columns report our verification tests: distortion-freeness (DF), soundness over the key (SoK), and soundness given a key (SgK) (a dash marks a test we did not run). Bold marks the best value of each column among the schemes that pass DF and SoK.

		Robustness TPR@1 ↑				Quality deviation (%) ↓			Verification		
Scheme	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3	DF	SoK	SgK
Prior semantic watermarks											
PASA	33	1	0	0	0	6.4	1.2	2.9	✓	✓	–
WatME ($\delta = 3$)	100	98	97	100	88	120	6.8	11	✗	✓	✗
WatME ($\delta = 2$)	100	78	75	96	69	51	4.1	6.6	✗	✓	✗
SemaMark	96	63	31	68	32	43	5.3	8.2	✗	✓	✗
Baselines											
AAR	100	19	30	92	41	2.7	2.4	9.8	✓	✓	✗
SynthID	**100**	30	44	95	56	1.6	0.6	3.7	✓	✓	✗
TextSeal	**100**	34	54	97	58	2.5	**0.5**	5.1	✓	✓	✓
Agent schemes											
ConcordMark (high)	**100**	79	**94**	**100**	**84**	4.4	0.8	3.4	✓	✓	✓
RotationFirst	100	**90**	86	97	80	**0.5**	1.0	**1.9**	✓	✓	✓

I Comparison with Semantic Watermarks

Several agent schemes, including ConcordMark and RotationFirst (App. H), rely on the semantics of tokens, e.g., with the `Lexical group` (App. A.2). In this section, we compare them with prior watermarks that also use token semantics to improve robustness.

Semantic watermarks We implement three prior semantic watermarks in our watermarking library. PASA (Ai & He, 2026) clusters the vocabulary into 4 semantic classes (with KMeans on GTE-Qwen2-7B token embeddings) and seeds the watermark with the classes of the 3 preceding tokens. It caps the probability of each class at 0.4, samples a class (or an overflow state) by inverse-transform sampling, and then samples the token within this class. Its detector counts the class matches, using a model forward pass to recover the capped class probabilities, and computes an exact p-value that accounts for repeated contexts. WatME (Chen et al., 2024) is a Red-Green watermark in which the vocabulary is split into disjoint synonym clusters (we use the first WordNet synset of each word token), and each cluster is split evenly between the green and red lists, so that the synonyms of most words remain green. SemaMark (Ren et al., 2024) is a Red-Green watermark seeded by the semantics of the 15 preceding tokens: their input embeddings are averaged with linearly increasing weights, projected onto a ring, and the angle is discretized into 4 bins, whose index seeds the green list. We keep the hyperparameters of each paper ($\gamma = 0.3$ and $\delta = 3$ for WatME, which we also evaluate with the milder $\delta = 2$, and $\gamma = 0.25$ and $\delta = 2$ for SemaMark) and, as for all schemes, sample from the top-50 tokens. We deviate from the original papers in three places. First, PASA's class probabilities are computed on the top-50 distribution, and its detector uses Llama-3.1-8B-Instruct without the prompt. Second, instead of SemaMark's contrastively trained projection, we project onto the top two principal directions of the embedding matrix and place the 4 bins at quantiles of natural text, so that each bin is equally likely. Third, both Red-Green schemes use the exact binomial test on deduplicated (seed, token) pairs (Fernandez et al., 2023), as SemaMark's offset search and WatME's z-test do not provide valid p-values.

Experimental setup We run our full private verification and evaluation (App. D) on the three schemes, without p-value correction, and compare them to the no-correction results of Table 6. PASA requires a model forward pass on every text at detection, and our soundness given a key test samples from a corpus of over a million texts. Hence, we only run the distortion-freeness and soundness over the key tests for PASA.

Results We report the results in Table 5. WatME and SemaMark are Red-Green watermarks: they are not distortion-free, fail our soundness given a key test, and significantly degrade quality. For SemaMark, the few possible seeds make the green lists nearly fixed across texts, as in the naive scheme of Sec. 3.2, so that many keys exceed the target false positive rate. WatME is the most robust scheme, but at the cost of more than doubling the perplexity of the replies, and with a milder δ it falls below ConcordMark in robustness while still degrading quality. PASA is distortion-free and

sound over the key, but has low power: on low-entropy replies, most tokens are sampled from the overflow state, which carries no signal, so it is not robust to any attack. PASA and SemaMark also vary strongly across keys (Table 16), e.g., PASA detects between 33% and 94% of the unedited watermarked replies, and SemaMark between 32% and 66% of the paraphrased replies, depending on the key. This is because their few semantic seeds ($4^3 = 64$ contexts for PASA and 4 green lists for SemaMark) are reused across all replies, so that a handful of pseudo-random draws determines the strength of each key. Overall, no prior semantic watermark is both valid and robust. In contrast, ConcordMark and RotationFirst, which also rely on token semantics, pass all our tests and are more robust than all prior semantic watermarks but WatME with $\delta = 3$, at a negligible cost in quality.

J PROOFS

In this section, we prove the theoretical claims made throughout the paper. In App. J.1, we state our assumptions and two lemmas used in the rest of the section. In App. J.2, we prove that a detector that is sound over the key can be turned into a detector that is sound given a key (Sec. 3.2). In App. J.3, we prove that the three statistical tests of Sec. 3.3 are valid. In App. J.4, we prove that `ResidualRace` is distortion-free (App. A.5). Lastly, in App. J.5, we derive the null distribution of the `Predictive Gamma` and `Position evidence` detectors (App. A.6). For all other mathematical results in the paper, we consider their proofs sufficiently straightforward and consequently do not write them down here.

J.1 PRELIMINARIES

Setup Let Ξ be the key space, and let ξ be drawn from Ξ as in Equation (2). For every text $\omega \in \Sigma^*$, the detector returns a p-value $P_\xi(\omega) \in [0, 1]$, and we assume $(\xi, \omega) \mapsto P_\xi(\omega)$ is measurable (which holds trivially for a finite key space). Humans write texts without knowing the private key: for any human text distribution $\mathcal{H}_0$, the text $\Omega \sim \mathcal{H}_0$ is independent of ξ. As in Sec. 3.1, we model the PRNG as ideal: the seed a_t depends only on the context, and, over a uniformly drawn key, the scores of distinct (seed, unit) pairs are independent, with $U_{a,u} \sim \mathcal{U}([0, 1])$ (and $Z_{a,u}$ following its geometry for private-geometry scores). We also assume that the hash function has no collisions (App. G). Finally, we write $T(s, p, c) := \Pr_{Z\sim\mathrm{Binomial}(s,p)}[Z \geq c]$ to denote the binomial upper tail, as in Algorithm 3.

Lemma 1. *Let X be a real random variable and $\varphi : \mathbb{R} \to [0, \infty)$ a continuous nonincreasing function such that $\Pr[X \geq s] \leq \varphi(s)$ for all $s \in \mathbb{R}$. Then $\min\{1, \varphi(X)\}$ is a valid p-value, i.e., $\Pr[\min\{1, \varphi(X)\} \leq x] \leq x$ for all $x \in [0, 1]$.*

Proof. The case $x = 1$ is trivial, and the case $x = 0$ follows from the case $x > 0$, since $\Pr[\min\{1, \varphi(X)\} \leq 0] \leq \Pr[\min\{1, \varphi(X)\} \leq x]$ for every $x > 0$. Let $x \in (0, 1)$ and $I := \{s \in \mathbb{R} : \varphi(s) \leq x\}$. If $I = \emptyset$, the event is empty. Otherwise, I is an interval unbounded above, and it is bounded below, since otherwise $\Pr[X \geq s] \leq x < 1$ for all s, which contradicts $\Pr[X \geq s] \to 1$ as $s \to -\infty$. Let $s^* := \inf I$. By continuity, $\varphi(s^*) \leq x$, and since $\{\varphi(X) \leq x\} \subseteq \{X \geq s^*\}$,

$$\Pr[\min\{1, \varphi(X)\} \leq x] \leq \Pr[X \geq s^*] \leq \varphi(s^*) \leq x. \qquad \square$$

Lemma 2. *Let X be a sum of s independent Bernoulli random variables with parameters at most $p \in [0, 1]$. Then $\Pr[X \geq c] \leq T(s, p, c)$ for all $c \in \mathbb{N}$, and $\Pr[T(s, p, X) \leq \eta] \leq \eta$ for all $\eta \in [0, 1]$.*

Proof. Let $p_1, \ldots, p_s \leq p$ be the Bernoulli parameters and $V_1, \ldots, V_s$ i.i.d. uniform random variables on $[0, 1]$. Then X has the same distribution as $\sum_i \mathbb{1}\{V_i \leq p_i\} \leq \sum_i \mathbb{1}\{V_i \leq p\} \sim \mathrm{Binomial}(s, p)$, which gives the first claim. For the second claim, since $c \mapsto T(s, p, c)$ is nonincreasing and $T(s, p, s + 1) = 0$, let $c^* := \min\{c \in \{0, \ldots, s + 1\} : T(s, p, c) \leq \eta\}$. Then $\{T(s, p, X) \leq \eta\} = \{X \geq c^*\}$, and, by the first claim, $\Pr[X \geq c^*] \leq T(s, p, c^*) \leq \eta$, which concludes the proof. $\square$

J.2 SOUNDNESS GIVEN A KEY

We now formally prove the claim made in Sec. 3.2: multiplying the p-value of any detector that is sound over the key (Equation (2)) by $1/\beta$ gives a detector that is also sound given a key (Equation (3)).

Proposition 1. *Let $\beta \in (0, 1]$ and let P_ξ be a detector that satisfies Equation* (2)*. Define the calibrated detector*

$$P_\xi^\beta(\omega) := \min\left\{1, \frac{P_\xi(\omega)}{\beta}\right\}. \tag{14}$$

Then P_ξ^β satisfies Equation (2)*, and for every human text distribution $\mathcal{H}_0$ and every $\alpha \in [0, 1]$,*

$$\Pr_\xi\left[\Pr_{\Omega\sim\mathcal{H}_0}[P_\xi^\beta(\Omega) \le \alpha] > \alpha\right] \le \beta,$$

i.e., P_ξ^β also satisfies Equation (3)*.*

Proof. We first relate the rejection regions of both detectors. For $\alpha \in [0, 1)$, since $\min\{1, x\} \le \alpha < 1$ holds if and only if $x \le \alpha$, we have for every key ξ and text ω

$$P_\xi^\beta(\omega) \le \alpha \iff P_\xi(\omega) \le \alpha\beta. \tag{15}$$

For $\alpha = 1$, both claims are trivial, as all probabilities are at most 1. We henceforth assume $\alpha \in [0, 1)$.

Soundness over the key. Let $\omega \in \Sigma^*$. By Equation (15) and Equation (2) applied at the threshold $\alpha\beta \in [0, 1]$,

$$\Pr_\xi[P_\xi^\beta(\omega) \le \alpha] = \Pr_\xi[P_\xi(\omega) \le \alpha\beta] \le \alpha\beta \le \alpha.$$

Soundness given a key. Let $\mathcal{H}_0$ be a human text distribution, and define for each key the false positive rate

$$g(\xi) := \Pr_{\Omega\sim\mathcal{H}_0}[P_\xi^\beta(\Omega) \le \alpha] = \Pr_{\Omega\sim\mathcal{H}_0}[P_\xi(\Omega) \le \alpha\beta],$$

where the equality follows from Equation (15). Because Ω is independent of ξ, Fubini's theorem allows us to swap the two expectations, and applying Equation (2) to each fixed text gives

$$\mathbb{E}_\xi[g(\xi)] = \Pr_{\xi,\Omega}[P_\xi(\Omega) \le \alpha\beta] = \mathbb{E}_{\Omega\sim\mathcal{H}_0}\left[\Pr_\xi[P_\xi(\Omega) \le \alpha\beta]\right] \le \alpha\beta.$$

If $\alpha = 0$, then $g \ge 0$ and $\mathbb{E}_\xi[g(\xi)] \le 0$, so $g(\xi) = 0$ almost surely and $\Pr_\xi[g(\xi) > 0] = 0 \le \beta$. Otherwise, $\alpha > 0$ and Markov's inequality gives

$$\Pr_\xi[g(\xi) > \alpha] \le \frac{\mathbb{E}_\xi[g(\xi)]}{\alpha} \le \frac{\alpha\beta}{\alpha} = \beta,$$

which concludes the proof. □

J.3 Validity of the Verification Tests

We prove that each of the three tests of Sec. 3.3 rejects any valid scheme with probability at most δ.

Proposition 2. *Let f be a distortion-free logits transformation (Equation* (1)*). For each context t, let $v_1, \dots, v_k$ be the k most likely tokens under p_t and let*

$$G_t(q) := \Big(q(v_1), \dots, q(v_k), 1 - \sum_{r=1}^{k} q(v_r)\Big)$$

map a distribution $q \in \Delta(\Sigma)$ to a probability vector over the head tokens and the pooled remaining tokens. For m independent keys, let $\zeta_t^{(j)}$ be the watermark score $\tilde{U}_{a_t}$ that the scheme feeds to f at context $\omega_{<t}$ under the j-th key (including any private per-request randomness of the scheme), $d_j := G_t(f(\zeta_t^{(j)}, p_t)) - G_t(p_t)$, $\bar{d}$ their mean, $h := \lfloor m/2 \rfloor$, $u := \operatorname{sign}(h^{-1}\sum_{j=1}^{h} d_j)$ and $b := \max\{0, (m-h)^{-1}\sum_{j=h+1}^{m} u^\top d_j\}$. Algorithm 1 combines the p-values

$$p_{\mathrm{coord}} := \min\{1, 2(k+1)\exp(-2m\|\bar{d}\|_\infty^2)\} \quad \textit{and} \quad p_{\mathrm{split}} := \exp(-(m-h)b^2/2)$$

into $q_t := \min\{1, 2\min(p_{\mathrm{coord}}, p_{\mathrm{split}})\}$, and rejects f if $Q := \min\{1, N\min_t q_t\} \le \delta$, where N is the number of contexts. Then Algorithm 1 rejects f with probability at most its significance level δ.

Proof. Fix a context t. Because the m keys are drawn independently, the scores $\zeta_t^{(1)}, \dots, \zeta_t^{(m)}$ are i.i.d., and by Equation (1), the differences $d_1, \dots, d_m$ are i.i.d. with $\mathbb{E}[d_j] = G_t(\mathbb{E}[f(\zeta_t^{(j)}, p_t)]) - G_t(p_t) = 0$, as G_t is affine.

Coordinate test. Each coordinate of $G_t(q)$ lies in $[0, 1]$ for every $q \in \Delta(\Sigma)$, so each coordinate of d_j lies in a fixed interval of length 1. By Hoeffding's inequality for each coordinate and a union bound over the $k + 1$ coordinates, $\Pr[\|\bar{d}\|_\infty \geq s] \leq 2(k + 1)\exp(-2ms^2)$ for all $s \geq 0$. Hence, by Lemma 1 applied to $X = \|\bar{d}\|_\infty$ using $\varphi(s) = 2(k + 1)\exp(-2m\max\{0, s\}^2)$, which is continuous, nonincreasing, and at least 1 for $s \leq 0$, $p_{\text{coord}} = \min\{1, \varphi(X)\}$ is a valid p-value, as $X \geq 0$.

Split test. The direction u only depends on $d_1, \dots, d_h$, so given u, the variables $Y_j := u^\top d_j$ for $j > h$ are i.i.d. with mean zero. Since $G_t(q)$ and $G_t(p_t)$ are probability vectors and $u \in [-1, 1]^{k+1}$, $u^\top G_t(q) \in [-1, 1]$, so Y_j lies in a fixed interval of length 2. By Hoeffding's inequality, $\Pr[b \geq s \mid u] \leq \exp(-(m - h)s^2/2)$ for all $s > 0$, where $m - h \geq 1$ as $m \geq 2$. Hence, by Lemma 1 with $\varphi(s) = \exp(-(m-h)\max\{0, s\}^2/2)$, which is continuous, nonincreasing, and equal to 1 for $s \leq 0$, $p_{\text{split}} = \varphi(b)$ is a valid p-value conditional on u, as $b \geq 0$, and thus also unconditionally.

Combination. By a union bound over the two tests, $\Pr[q_t \leq x] \leq \Pr[p_{\text{coord}} \leq x/2] + \Pr[p_{\text{split}} \leq x/2] \leq x$. By another union bound over the N contexts, $\Pr[Q \leq \delta] \leq \sum_{t=1}^{N} \Pr[q_t \leq \delta/N] \leq \delta$. □

Proposition 3. *Let P_ξ be a detector that is sound over the key (Equation (2)), i.e.,*

$$\forall \omega \in \Sigma^*, \forall \alpha \in [0, 1], \quad \Pr_\xi[P_\xi(\omega) \leq \alpha] \leq \alpha.$$

Given a corpus $C = (\omega_1, \dots, \omega_n)$, a finite set of thresholds $\mathcal{A}$ and m independent keys $\xi_1^i, \dots, \xi_m^i$ per text, Algorithm 2 counts the false positives $X_i(\alpha) := \sum_{j=1}^{m} \mathbb{1}\{P_{\xi_j^i}(\omega_i) \leq \alpha\}$, and rejects P_ξ if $q_i(\alpha) := T(m, \alpha, X_i(\alpha)) \leq \eta := (1 - (1 - \delta)^{1/n})/|\mathcal{A}|$ for some text ω_i and threshold $\alpha \in \mathcal{A}$. Then, for every fixed text corpus C, Algorithm 2 rejects P_ξ with probability at most its significance level δ.

Proof. Fix a text ω_i and a threshold $\alpha \in \mathcal{A}$. Because the keys $\xi_1^i, \dots, \xi_m^i$ are i.i.d., $X_i(\alpha)$ is a sum of m i.i.d. Bernoulli random variables with parameter $\Pr_\xi[P_\xi(\omega_i) \leq \alpha] \leq \alpha$ by soundness over the key. Hence, by Lemma 2, $\Pr[q_i(\alpha) \leq \eta] \leq \eta$, and by a union bound over the thresholds, the event $E_i := \{\exists \alpha \in \mathcal{A} : q_i(\alpha) \leq \eta\}$ has probability at most $|\mathcal{A}|\eta = 1 - (1 - \delta)^{1/n}$. The events $E_1, \dots, E_n$ depend on disjoint sets of independent keys, so they are independent, and

$$\Pr[\mathcal{R} \neq \emptyset] = 1 - \prod_{i=1}^{n}(1 - \Pr[E_i]) \leq 1 - (1 - \delta) = \delta. \qquad \square$$

Proposition 4. *Let P_ξ be a detector that is sound given a key (Equation (3)) for the human text distribution $\mathcal{H}_0$, the fraction β, and the thresholds $\mathcal{A}$, i.e.,*

$$\forall \alpha \in \mathcal{A}, \quad \Pr_\xi[\Pr_{\Omega \sim \mathcal{H}_0}[P_\xi(\Omega) \leq \alpha] > \alpha] \leq \beta.$$

Algorithm 3 draws d independent keys ξ_j and, for each key, n independent texts $\omega_1^j, \dots, \omega_n^j \sim \mathcal{H}_0$, and counts the false positives $X_j(\alpha) := \sum_{t=1}^{n} \mathbb{1}\{P_{\xi_j}(\omega_t^j) \leq \alpha\}$. For each threshold $\alpha \in \mathcal{A}$ and screening level γ in a predeclared finite set $\mathcal{S} \subset (0, 1)$, a key screens positive if $X_j(\alpha) \geq c$, where $c := \min\{k \in \{1, \dots, n + 1\} : T(n, \alpha, k) \leq \gamma\}$ is the smallest count that is significant at level γ for a key with false positive rate α. Let $r := T(n, \alpha, c)$, $b := \beta + (1 - \beta)r$, and B the number of keys that screen positive. Algorithm 3 rejects P_ξ if $q := T(d, b, B) \leq \eta := \delta/(|\mathcal{A}||\mathcal{S}|)$ for some pair $(\alpha, \gamma) \in \mathcal{A} \times \mathcal{S}$. Then Algorithm 3 rejects such a P_ξ with probability at most its significance level δ.

Proof. Fix $(\alpha, \gamma) \in \mathcal{A} \times \mathcal{S}$, and note that c, r and b are deterministic. For a key ξ, let $g(\xi) := \Pr_{\Omega \sim \mathcal{H}_0}[P_\xi(\Omega) \leq \alpha]$ be its false positive rate, and say that ξ is bad if $g(\xi) > \alpha$. Given ξ_j, the texts $\omega_1^j, \dots, \omega_n^j$ are i.i.d., so $X_j(\alpha) \sim \text{Binomial}(n, g(\xi_j))$. If ξ_j is not bad, $g(\xi_j) \leq \alpha$, and Lemma 2 gives $\Pr[X_j(\alpha) \geq c \mid \xi_j] \leq T(n, \alpha, c) = r$. As a key is bad with probability $\pi \leq \beta$ by soundness given a key,

$$\Pr[X_j(\alpha) \geq c] \leq \pi + (1 - \pi)r = r + \pi(1 - r) \leq \beta + (1 - \beta)r = b.$$

By independence across keys, B is a sum of d independent Bernoulli variables with parameters at most b, so by Lemma 2, $\Pr[q \leq \eta] \leq \eta$. By a union bound over $\mathcal{A} \times \mathcal{S}$, $\Pr[\mathcal{R} \neq \emptyset] \leq |\mathcal{A}||\mathcal{S}|\eta = \delta$. □

J.4 Distortion-Freeness of Residual Clocks

We prove that `ResidualRace` (Equation (6)) is distortion-free even when a context is repeated within a request, whereas the `Gumbel race` requires the per-request cache to guarantee this (App. A.3). The proof relies on the following classical property of exponential races, which we prove for completeness.

Lemma 3. *Let $E_1, \dots, E_K$ be i.i.d.* $\mathrm{Exp}(1)$ *random variables and $m_1, \dots, m_K > 0$ weights with $\sum_k m_k = 1$. Let $\tau := \min_k E_k/m_k$ and $\kappa := \arg\min_k E_k/m_k$. Then $\Pr[\kappa = k] = m_k$ for every k, and conditional on (τ, κ), the residual clocks $(E_j - \tau m_j)_{j \neq \kappa}$ are i.i.d.* $\mathrm{Exp}(1)$ *random variables.*

Proof. Since E_k/m_k has density $m_k e^{-m_k s}$, for all k, $s \geq 0$ and $x_j \geq 0$,

$$\Pr[\kappa = k, \tau \in \mathrm{d}s, E_j - s m_j > x_j \, \forall j \neq k] = m_k e^{-m_k s}\,\mathrm{d}s \prod_{j \neq k} e^{-(s m_j + x_j)} = m_k \cdot e^{-s}\,\mathrm{d}s \cdot \prod_{j \neq k} e^{-x_j}.$$

The density factorizes, so κ, τ and the residuals are independent with the claimed distributions. □

Proposition 5. *Assume that the initial scores $\tilde{U}_{a,u}$ of distinct (seed, unit) pairs are i.i.d. uniform. Then, for* `ResidualRace`, $\Pr[\omega_t = v \mid \omega_{<t}] = p_t(v)$ *for every step t and token $v \in \Sigma$, even if the seed a_t has already been used within the same request, so that no per-request cache (App. A.3) is required.*

Proof. Let $\mathcal{F}_t$ be the information generated by $\omega_{\leq t}$ and the race outcomes $(\tau_s, u_s)_{s \leq t}$. We show by induction the invariant: conditional on $\mathcal{F}_t$, the current clocks $R_{a,u}$ of all (seed, unit) pairs are i.i.d. $\mathrm{Exp}(1)$, where we identify a clock that is not yet initialized with its initial value $-\log \tilde{U}_{a,u}$. At step $t = 0$, the invariant holds because the initial scores are i.i.d. uniform and each $-\log \tilde{U}_{a,u} \sim \mathrm{Exp}(1)$.

Assume the invariant holds at $t-1$. Given $\mathcal{F}_{t-1}$, the distribution p_t, the masses M_t and the seed a_t are fixed, and the race of Equation (6) is an exponential race over the i.i.d. $\mathrm{Exp}(1)$ clocks $R_{a_t,u}$ with weights $M_t(u)$. By Lemma 3, $\Pr[u_t = u \mid \mathcal{F}_{t-1}] = M_t(u)$, and since the token is then sampled inside u_t with private randomness,

$$\Pr[\omega_t = v \mid \mathcal{F}_{t-1}] = M_t(\mathcal{G}(v)) \cdot \frac{p_t(v)}{M_t(\mathcal{G}(v))} = p_t(v).$$

Because p_t is a function of $\omega_{<t}$, taking the expectation given $\omega_{<t}$ proves the claim at step t. For the invariant, by Lemma 3, conditional on $\mathcal{F}_{t-1}$ and (τ_t, u_t), the updated losing clocks $R_{a_t,u} - \tau_t M_t(u)$ are i.i.d. $\mathrm{Exp}(1)$. The winning clock is replaced by a fresh $\mathrm{Exp}(1)$, and all other clocks (other seeds, or units with $M_t(u) = 0$) are unchanged and independent of the race. Given u_t, the token ω_t only depends on private randomness independent of the clocks, so the invariant also holds at step t. □

J.5 Validity of the Detectors

We derive the null distribution of the detectors of App. A.6 whose validity is not immediate. Throughout, the text ω is fixed and only the key is random, so an exact null distribution implies Equation (2).

Proposition 6. *Assume that, for every fixed geometry B and every score Z drawn according to its geometry, $d(B, Z) \sim \mathcal{U}([0,1])$. Then, for every text ω, the* `Predictive Gamma` *statistic of Equation* (12) *satisfies $S(\omega) \sim \mathrm{Gamma}(N, 1)$.*

Proof. The retained pairs D only depend on ω, so N is fixed and, by the ideal PRNG assumption, $Z_1, \dots, Z_N$ are i.i.d. The estimate B_i is computed from $A_0, \dots, A_{n_A - 1}$ and $Z_1, \dots, Z_{i-1}$ only, and Z_i is independent of $(Z_1, \dots, Z_{i-1})$. Hence, conditional on $Z_1, \dots, Z_{i-1}$, B_i is fixed and $d_i = d(B_i, Z_i) \sim \mathcal{U}([0,1])$. Since $d_1, \dots, d_{i-1}$ are functions of $Z_1, \dots, Z_{i-1}$, d_i is independent of $(d_1, \dots, d_{i-1})$, and by induction $d_1, \dots, d_N$ are i.i.d. uniform. Thus, the $-\log d_i$ are i.i.d. $\mathrm{Exp}(1)$ and $S(\omega) \sim \mathrm{Gamma}(N, 1)$. □

Proposition 7. *Assume that $\hat{p}_t$ only depends on $\omega_{<t}$. Then, for every text ω, the variables $(R_t)_{t \in D_{\mathrm{pos}}}$ of the* `Position evidence` *detector are i.i.d. uniform on $[0,1]$. In particular, $S(\omega) \sim \mathrm{Gamma}(|D_{\mathrm{pos}}|, 1)$, and the weighted statistic $\sum_{t \in D_{\mathrm{pos}}} -\eta_t \log R_t$ has the distribution of $\sum_{t \in D_{\mathrm{pos}}} \eta_t E_t$ for i.i.d. $E_t \sim \mathrm{Exp}(1)$.*

Proof. As $\hat{p}_t$ depends only on $\omega_{<t}$, the competitors C_t, the exponents $\alpha_{t,v}$, λ_t and the weights η_t are fixed. The seeds a_t for $t \in D_{\text{pos}}$ are distinct and $\omega_t \notin C_t$, so all scores U_{a_t,ω_t} and $U_{a_t,v}$ for $v \in C_t$ have distinct (seed, token) pairs and are thus independent and uniform. Hence, each R_t is a function of its own disjoint set of scores, so the R_t are independent. We now show that each is uniform.

Fix t and write $U := U_{a_t,\omega_t}$, $\lambda := \lambda_t$ and $W := W_t$. Given $U = u$, the competitors are independent, so $\Pr[W = 1 \mid U = u] = \prod_{v \in C_t} \Pr[U_{a_t,v} < u^{\alpha_{t,v}}] = u^{\lambda}$. Let $Y := W + U$, which ranks every winning outcome above every losing one, and then by the value of U. The variable Y has a continuous distribution, and its survival function is, for $W = 1$ and for $W = 0$ respectively,

$$\Pr[Y' \geq 1 + u] = \int_u^1 x^{\lambda} \, \mathrm{d}x = \frac{1 - u^{1+\lambda}}{1 + \lambda},$$

$$\Pr[Y' \geq u] = \frac{1}{1 + \lambda} + \int_u^1 (1 - x^{\lambda}) \, \mathrm{d}x = 1 - u + \frac{u^{1+\lambda}}{1 + \lambda},$$

where Y' is an independent copy of Y. Hence $R_t = \Pr[Y' \geq Y \mid Y]$, and by the probability integral transform, $R_t \sim \mathcal{U}([0, 1])$. □

K PROMPTS

In this section, we provide the initial prompt given to the autonomous research agent (App. K.1) and the prompts used for the back-translation and paraphrasing attacks during our evaluation (App. K.2).

K.1 AGENT PROMPT

Below, we reproduce the initial prompt that the autonomous research agent receives in our harness.

Watermark design prompt

Your goal is to design a new distortion-free watermarking scheme in the `lm-wm-tools` library that significantly outperforms prior work across all key dimensions: detectability, robustness, and diversity. For diversity and PPL, the goal is to be as close as possible to the corresponding unwatermarked values (the leaderboard ranks based on the distance to the unwatermarked values).
You should focus on new experimental ideas and reason mathematically about watermarking to make genuine breakthroughs rather than tweaking hyperparameters. Also, a constraint of the challenge is to focus only on distortion-free schemes.
A key challenge of watermark development is to make sure that:

- The scheme is distortion-free.
- The scheme is robust to edits to the generated text.

Each of your submissions should be accompanied by:

- A proof that your scheme is distortion-free and well-calibrated.
- A commit message.

Before an official submission, use the `lm-wm-tools verify` and `lm-wm-tools evaluate` commands to evaluate your proposed schemes. We provide public configuration but you are also welcome to use your owns. You have a limited number of official submissions (use `wm-harness limit` to see how many remain), so only evaluate schemes that you are confident (i) are sound and (ii) outperform prior work or submissions. Soundness, in particular, is a significant constraint: your scheme should satisfy the following properties as much as possible (reported results are for baselines verification).
Let K be the watermark key, Σ the vocabulary, $\mathcal{L}$ the LLM probability distribution, $\mathcal{L}_K$ the watermarked LLM probability distribution, and $p_K(\omega)$ the detector p-value for text ω under key K.
Distortion-free

$$\forall \omega \in \Sigma^*, \mathbb{E}_K[\mathcal{L}_K(\omega)] = \mathcal{L}(\omega)$$

Scheme	Context 1	Context 2	Context 3	Context 4	Context 5
AAR	1	0.4194	0.5642	1	—
SynthID	0.5092	0.6584	1	0.4254	1
TextSeal	1	1	1	1	—

Soundness over the key

$$\forall \omega \in \Sigma^*, \forall \alpha \in [0,1], \Pr_K[p_K(\omega) \leq \alpha] \leq \alpha$$

Scheme	Context 1	Context 2	Context 3	Context 4	Context 5
AAR	0.009677	0.02317	0.1124	0.2463	—
SynthID	0.1516	0.4806	0.5053	1	0.9043
TextSeal	0.3678	1	1	1	—

Soundness for a deployed key

$$\forall K, \Pr_{\Omega \sim \mathcal{H}_0}[p_K(\Omega) \leq \alpha] \leq \alpha$$

Scheme	Context 1	Context 2	Context 3	Context 4	Context 5
AAR	2.981e-82	4.811e-14	3.813e-05	0.07999	—
SynthID	8.056e-122	9.575e-41	4.136e-09	6.433e-05	0.2573
TextSeal	5.265e-67	0.001826	1	1	—

You are operating autonomously.

Official submissions
When you have a submission you like, submit it for private grading through the black-box `wm-harness` command:

```
wm-harness evaluate configs/MyScheme.yaml
wm-harness leaderboard
```

Run `wm-harness limit` before submitting to see the run's evaluation budget: `limit`, `used`, and `remaining` (`null` means unlimited). Each new official evaluation job consumes one call, even if verification or grading fails. Duplicate requests do not consume another call; removing a queued job does not refund it. Stop submitting when no calls remain.
The `wm-harness leaderboard` command shows you all prior evaluation results, and you should consider it trustworthy.
`wm-harness evaluate` runs official verification first, then evaluates the same committed snapshot only if every check passes. One job covers both stages. Results include `verification.all_passed` and `verification.checks`; rejected submissions have no evaluation metrics, and the CLI exits with code 2. By default, the command waits for both stages; use `--no-wait` to return the queued job immediately.
Before calling `wm-harness evaluate`:

1. Keep the working tree clean.
2. Commit the exact scheme, configuration, documentation, and required assets.
3. Submit the committed configuration path. The API grades an immutable archive of that commit and rejects dirty or uncommitted work.

Use `wm-harness status JOB_ID` to inspect an asynchronous job. Status includes `progress.stage`, a readable `progress.message`, `progress.updated_at` (last stage or completed grading pass), and `progress.heartbeat_at` (worker heartbeat every five seconds while running). Long model startup or grading passes can leave the stage unchanged.
We automatically queue submissions, so it is better to wait until the submission is done: you can use `wm-harness wait JOB_ID` to wait indefinitely for one job, or add `--timeout <seconds>` to bound the wait (run this command in the foreground). It is okay to wait for long

periods (e.g., a few hours) to save tokens; repeatedly waiting for only one minute is inefficient. You can also run `wm-harness list` to list all queued API queries along with their IDs.
Use `wm-harness remove JOB_ID` to remove a job that is still queued. Running or completed jobs cannot be removed. Removed jobs disappear from the job list and can be submitted again. Be aware that official evaluation can take a few minutes to a few hours, depending on the scheme implementation.
To help you, `lm-wm-tools verify` and `lm-wm-tools evaluate` are public subsets of the verification and evaluation stages run by `wm-harness evaluate`. You can use them locally to estimate whether your submission is worth evaluating on the private test.
API credentials for public paraphrasing evaluation are installed at `/home/sandbox/secret.sh`. Before running `lm-wm-tools evaluate`, load them in the same shell with `set -a; source /home/sandbox/secret.sh; set +a` so the CLI inherits the credentials.

Modifying `lm-wm-tools`
If you want to implement a scheme that does not fit well with the `lm-wm-tools` framework, you can modify the library more extensively. However, for the harness to work, your changes must remain compatible with the `lm-wm-tools verify` and `lm-wm-tools evaluate` commands. You should also make your watermark efficient and can change the library's inner workings to achieve the required efficiency.

Do not finish while official jobs are pending
When you are done, do not produce your final answer while any official evaluation job is queued or running. Instead, run in the foreground:

```
wm-harness wait
```

It waits until the official queue is empty and then returns the results of every submitted job. Do not background it, replace it with a custom monitoring script, or end your turn while it is running. Inspect all returned results, act on them, improve rejected or underperforming candidates, submit them with `wm-harness evaluate`, and run `wm-harness wait` again if that creates more queued work. Only finish after the official queue is empty and you have checked the returned results.

Notes
The private configuration, datasets, seeds, raw reports, and grader filesystem are intentionally unavailable. Do not attempt to infer, access, or reproduce them: this would be considered cheating.

K.2 Attack Prompts

Here, we detail the prompts used for the back-translation and paraphrasing attacks. In the examples below, `<text>` is replaced with the actual text to translate or paraphrase. For back-translation, the prompt is applied twice, first to translate the text into French and then back into the original language.

Back-translation prompt

Translate the following text to `{language}`. Your reply should only contain the translated text.
`<text>`

Paraphrasing prompt

Please rewrite the following text and return only the rewritten text: `<text>`

L Full Experimental Results

In this section, we report the full numerical results underlying the figures of this paper. In App. L.1, we give the results of every scheme submitted during the autoresearch trajectories of Sec. 4.1, and in App. L.2, those of every configuration evaluated during the component ablation study of App. A.

L.1 Autoresearch Trajectories

Table 6 reports the results of the three baselines and of the 52 submissions of the four research agents from Figure 2, in submission order and under each of the three p-value corrections described below.

Setup We use the private evaluation of Sec. 4: we report the worst TPR and the median quality distance over the 5 watermark keys, each evaluated on 1000 replies. We evaluate every scheme under the three p-value corrections of App. C: *guaranteed* (p-values multiplied by $1/\beta = 20$), *empirical* (smallest multiplier $\kappa \geq 1$ that passes our soundness tests, Figure 24), and *no correction* ($\kappa = 1$). As in App. C, we first remove the outer multiplier chosen by the agent (if any) and keep the rest of the detector unchanged. For each watermark key, all settings share the same generated replies and the same attacked texts: only the p-value multiplier changes. Hence, the quality columns are identical across all three settings.

Reading the table Each TPR cell reports the guaranteed / empirical / no-correction results, and the first column gives the submission index (i.e., the x-axis of Figure 2). Parentheses mark the results without correction of the schemes that then fail our soundness given a key test (i.e., whose empirical multiplier is above 1). Figure 2 instead uses the multiplier chosen by each agent: 20 for GPT-6 Astra (except for the first two submissions of the low variant), between 1 and 6 for Opus 5, and none for Gemini-3.8 Flash and the baselines. As we evaluate the settings on newly generated replies (with the same watermark keys), the results can differ by a few points from those of Figure 2.

L.2 Component Ablation

Tables 7–10 report all the configurations of the component ablation of App. A, following the same structure: watermark unit and context seeding, watermark score, logits transformation, and detection.

Setup We use the experimental setup of App. A.1: each configuration swaps one or a few components of the baseline scheme (identity grouping, `Fixed-length` context seeding with $k = 2$, $\tilde{U} = U$, the `Gumbel race`, and the `Gamma` detector). We evaluate all configurations on the private evaluation of Sec. 4 using a single watermark key, and multiply their p-values by $1/\beta = 20$ so that they are sound given a key (Sec. 3.2). Unlike the difference bar plots of App. A, we report absolute values with the same columns as Table 1: the TPR@1%FPR on clean text and under each attack, and the relative distances (in %) to the unwatermarked model in perplexity and in bigram and trigram Self-BLEU.

Reading the tables Each cell reports the result with the identity grouping (left) and with the `Lexical group` (App. A.2) (right), so that comparing the two sides of a cell gives the effect of the watermark unit on that configuration. The first row of each table is the baseline scheme, which every other configuration modifies. In Table 10, only the detector changes: each block rescores the exact same generated texts, so the quality columns are those of the corresponding generation. A dash on the right side of a cell indicates that we evaluated the configuration only with the identity grouping.

L.3 Per-Key Results

Throughout the paper, we report the worst TPR and the median quality distance over the 5 watermark keys. To show the variability across keys, Table 11 reports the per-key results of the selected schemes of Table 1, Tables 12–15 those of every submission of Table 6, and Table 16 those of the prior semantic watermarks of Table 5. In all cases, the setup is the same as for the aggregated tables, and each scheme spans 5 rows, one per watermark key.

Table 6: Full results of every submission of Figure 2, in submission order. Each TPR cell reports the guaranteed / empirical / no-correction settings of App. C (the quality columns do not depend on the setting). Parentheses mark values without correction for schemes that then fail our soundness given a key test. Within each group, bold marks the best (unrounded) value of each column and each setting.

			Robustness TPR@1 ↑				Quality deviation (%) ↓		
#	Scheme	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
Baselines									
	TextSeal	100 / **100 / 100**	**14 / 34 / 34**	**28 / 54 / 54**	**89 / 97 / 97**	**38 / 58 / 58**	2.5	**0.5**	5.1
	AAR	**100** / 100 / (100)	4 / 19 / (19)	8 / 30 / (30)	76 / 92 / (92)	19 / 41 / (41)	2.7	2.4	9.8
	SynthID	**100 / 100 / (100)**	9 / 30 / (30)	20 / 44 / (44)	88 / 95 / (95)	35 / 56 / (56)	**1.6**	0.6	**3.7**
GPT-6 ASTRA (xhigh)									
1	CascadeCopula	100 / 100 / (100)	60 / 79 / (80)	60 / 78 / (79)	95 / 99 / (99)	58 / 76 / (76)	0.9	2.3	8.0
2	LexicalCopula	100 / 100 / (100)	57 / 78 / (78)	69 / 83 / (84)	94 / 98 / (98)	63 / 78 / (79)	0.5	2.4	7.9
3	PortfolioRace	**100** / 100 / 100	51 / 67 / 67	70 / 80 / 80	94 / 98 / 98	59 / 71 / 71	1.7	1.0	1.4
4	TailRace	99 / 100 / (100)	44 / 66 / (67)	55 / 73 / (73)	87 / 94 / (94)	50 / 70 / (70)	1.7	2.0	6.1
5	DirectionalRenewal	**100** / 100 / 100	67 / 83 / 83	73 / 86 / 86	96 / **99 / 99**	64 / 78 / 78	2.1	2.5	6.5
6	HaarCap	**100 / 100 / 100**	64 / 80 / 80	69 / 84 / 84	94 / 98 / 98	62 / 76 / 76	0.6	0.8	2.7
7	HaarGamma	100 / 100 / 100	67 / 80 / 80	74 / 84 / 84	96 / 98 / 98	66 / 76 / 76	0.5	0.6	2.3
8	HaarPredictive	100 / **100 / (100)**	71 / 85 / (85)	77 / 88 / (88)	**96** / 98 / (98)	68 / 80 / (80)	1.0	**0.3**	2.1
9	HaarGraph	100 / 100 / 100	72 / 86 / 86	77 / 88 / 88	96 / 98 / 98	69 / 82 / 82	1.0	1.1	1.3
10	HaarLexical	99 / 99 / (99)	83 / 92 / (92)	76 / 86 / (86)	92 / 97 / (97)	66 / 79 / (79)	0.5	0.5	1.7
11	HaarLexicalFirst	99 / 100 / (100)	**84** / 91 / (91)	**81 / 89 / (89)**	94 / 98 / (98)	68 / 82 / (82)	1.4	0.6	1.8
12	ProjectiveFirst	99 / 100 / 100	81 / 91 / 91	75 / 86 / 86	93 / 97 / 97	64 / 78 / 78	0.7	0.9	2.0
13	AxialFirst	99 / 100 / (100)	83 / **92 / (92)**	78 / 88 / (88)	95 / 98 / (98)	**70 / 83 / (83)**	**0.4**	0.7	**1.0**
14	RotationFirst	99 / 100 / 100	81 / 90 / 90	77 / 86 / 86	93 / 97 / 97	68 / 80 / 80	0.5	1.0	1.9
GPT-6 ASTRA (low)									
1	FreshRace	99 / 100 / (100)	13 / 34 / (35)	14 / 37 / (37)	74 / 90 / (90)	28 / 50 / (51)	1.4	**0.9**	4.3
2	FreshRace-selective	100 / 100 / (100)	26 / 52 / (53)	28 / 54 / (54)	89 / 96 / (96)	44 / 65 / (65)	2.0	2.6	9.1
3	FreshRace-deployment	100 / **100 / (100)**	39 / 61 / (62)	43 / 69 / (69)	92 / 97 / (97)	53 / 72 / (72)	1.3	1.7	7.8
4	FreshRace-channels	**100 / 100 / 100**	38 / 60 / 60	46 / 65 / 65	94 / 98 / 98	55 / 71 / 71	1.9	1.0	4.9
5	FreshRace-adaptive-channels	100 / 100 / (100)	42 / 65 / (65)	44 / 66 / (67)	91 / 96 / (96)	53 / 70 / (70)	2.6	1.2	3.7
6	FreshRace-split	**100 / 100 / (100)**	58 / 75 / (75)	60 / 78 / (78)	**96 / 98 / (98)**	**62 / 76 / (76)**	3.2	2.2	6.2
7	FreshRace-antithetic-reservoir	**100 / 100 / 100**	49 / 67 / 67	50 / 68 / 68	93 / 97 / 97	53 / 68 / 68	3.3	1.2	**1.6**
8	FreshRace-balanced	**100 / 100 / 100**	40 / 60 / 60	50 / 67 / 67	94 / 97 / 97	55 / 70 / 70	**0.6**	1.4	5.4
9	FreshRace-lexical	**100 / 100 / 100**	51 / 71 / 71	64 / **81 / 81**	95 / 98 / 98	59 / 75 / 75	3.5	1.4	4.2
10	FreshRace-lexical	100 / 100 / 100	51 / 69 / 69	**66** / 81 / 81	95 / 98 / 98	59 / 74 / 74	3.5	1.3	4.4
11	RenewalRace	95 / 98 / (98)	**80 / 90 / (91)**	52 / 71 / (74)	82 / 92 / (92)	52 / 69 / (72)	4.0	2.5	5.1
OPUS 5									
1	TallyMark	100 / 100 / (100)	37 / 63 / (64)	57 / 77 / (78)	86 / 94 / (95)	44 / 63 / (64)	2.6	2.5	7.2
2	TallyMark	100 / 100 / (100)	32 / 57 / (57)	34 / 60 / (60)	87 / 95 / (95)	42 / 64 / (64)	1.7	1.6	5.8
3	TallyMark	100 / **100 / (100)**	33 / 61 / (62)	60 / 80 / (81)	88 / 95 / (95)	47 / 67 / (67)	2.7	3.3	7.6
4	TallyMark	100 / 100 / (100)	36 / 64 / (64)	57 / 80 / (80)	88 / 97 / (97)	49 / 69 / (70)	1.9	3.0	7.7
5	ChoirMark	100 / **100 / (100)**	57 / 75 / (75)	78 / 87 / (87)	96 / 99 / (99)	62 / 75 / (75)	3.8	11	29
6	CanonMark	100 / 100 / 100	43 / 59 / 59	66 / 78 / 78	93 / 96 / 96	51 / 64 / 64	4.0	0.7	**1.1**
7	LadderMark	100 / 100 / 100	51 / 66 / 66	67 / 78 / 78	95 / 98 / 98	56 / 68 / 68	3.3	**0.6**	1.9
8	PreludeMark	**100 / 100 / (100)**	**73 / 88 / (88)**	79 / 90 / (91)	98 / 99 / (99)	70 / 84 / (84)	**1.2**	5.4	16
9	ConsortMark	100 / **100 / (100)**	65 / 79 / (79)	70 / 82 / (82)	97 / 99 / (99)	62 / 75 / (75)	4.8	0.8	2.4
10	SestetMark	100 / **100 / (100)**	66 / 82 / (82)	78 / 86 / (86)	97 / 99 / (99)	67 / 78 / (78)	5.5	1.0	3.1
11	MotetMark	100 / 100 / (100)	66 / 82 / (82)	77 / 87 / (87)	97 / 99 / (99)	62 / 78 / (78)	6.5	1.0	3.2
12	CodaMark	100 / 100 / (100)	66 / 81 / (81)	77 / 86 / (86)	97 / 99 / (99)	65 / 79 / (79)	6.3	1.1	3.3
13	AccentMark	**100 / 100 / 100**	69 / 82 / 82	83 / 90 / 90	**99 / 100 / 100**	74 / 84 / 84	4.9	1.3	3.8
14	RowMark	**100 / 100 / 100**	67 / 84 / 84	82 / 91 / 91	**99** / 100 / 100	74 / **86 / 86**	5.9	1.1	3.7
15	UnisonMark	100 / **100 / 100**	68 / 82 / 82	84 / 93 / 93	99 / 100 / 100	74 / 85 / 85	5.3	1.0	3.4
16	ConcordMark	**100 / 100 / 100**	65 / 79 / 79	**89 / 94 / 94**	99 / 100 / 100	**75** / 84 / 84	4.4	0.8	3.4
GEMINI-3.8 FLASH (high)									
1	OmniSeal	**100 / 100 / 100**	**18 / 41 / 41**	34 / 58 / 58	91 / 97 / 97	**45** / 62 / 62	3.1	4.0	12
2	OmniSeal	**100 / 100 / 100**	16 / 40 / 40	34 / 57 / 57	**93 / 98 / 98**	44 / 64 / 64	2.4	4.2	13
3	OmniSeal	**100 / 100 / 100**	16 / 40 / 40	32 / 58 / 58	92 / 98 / 98	44 / 63 / 63	**2.3**	3.7	12
4	OmniSeal	100 / **100 / 100**	16 / 40 / 40	**35 / 61 / 61**	92 / 97 / 97	42 / **64 / 64**	3.1	4.0	13
5	OmniSeal-G0	100 / **100 / 100**	14 / 37 / 37	32 / 58 / 58	90 / 96 / 96	41 / 63 / 63	3.1	4.0	12
6	OmniSeal-K40	100 / **100 / 100**	15 / 37 / 37	31 / 58 / 58	91 / 97 / 97	38 / 59 / 59	7.1	4.6	14
7	OmniSeal-K45	**100 / 100 / 100**	16 / 40 / 40	34 / 58 / 58	92 / 98 / 98	42 / 62 / 62	4.1	3.9	13
8	OmniSeal-K35	100 / 100 / 100	13 / 36 / 36	29 / 56 / 56	92 / 98 / 98	38 / 60 / 60	10	5.7	15
9	OmniSeal-W05	100 / **100 / 100**	17 / 39 / 39	31 / 57 / 57	92 / 98 / 98	41 / 61 / 61	2.6	3.5	12
10	OmniSeal-C4	**100 / 100 / 100**	5 / 16 / 16	14 / 34 / 34	85 / 94 / 94	25 / 45 / 45	2.6	**2.4**	**7.9**
11	OmniSeal-K30	100 / **100 / 100**	12 / 32 / 32	26 / 53 / 53	92 / 98 / 98	38 / 60 / 60	16	6.2	17

Table 7: Watermark unit and context seeding (App. A.2 and A.3). Each cell reports identity grouping / `Lexical group`.

		Robustness TPR@1 ↑				Quality deviation (%) ↓		
Variant	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
Core (fixed $k = 2$, `Gumbel race`, `Gamma`)	100 / 100	56 / 54	67 / 76	97 / 98	64 / 68	0.1 / 1.1	4.5 / 4.6	16 / 16
Fixed-length								
$k = 0$	100 / 99	92 / 88	70 / 61	93 / 90	67 / 60	30 / 32	12 / 13	29 / 31
$k = 1$	99 / 99	57 / 51	51 / 55	89 / 89	50 / 49	2.7 / 2.5	5.5 / 4.1	14 / 11
$k = 3$	100 / 100	25 / 25	42 / 60	96 / 97	52 / 54	0.9 / 0.4	3.1 / 2.1	11 / 8.1
Unordered								
Pair $\{\omega_{t-1}, v\}$	100 / 100	91 / 90	91 / 94	99 / 99	82 / 85	0.2 / 0.0	13 / 12	34 / 32
$k = 2$	100 / 100	58 / 56	68 / 79	97 / 98	62 / 67	0.6 / 1.0	4.1 / 3.7	15 / 14
$k = 3$	100 / 100	22 / 21	43 / 56	97 / 97	50 / 51	0.0 / 0.9	2.5 / 2.2	9.5 / 8.0
Adaptive and global length								
`Adaptive length`	100 / 100	73 / 70	77 / 79	98 / 98	71 / 71	2.3 / 1.9	5.7 / 5.4	17 / 16
+ `Excess` detector	100 / –	72 / –	78 / –	98 / –	72 / –	1.0 / –	5.6 / –	17 / –
`Global + local`	100 / 100	88 / 85	76 / 77	96 / 96	70 / 69	16 / 18	10 / 9.2	25 / 23
`Global first use`	100 / 100	90 / 85	76 / 80	97 / 98	72 / 72	15 / 16	13 / 11	34 / 28
Anchored								
`Anchored` ($S_{\max} = 4$)	96 / 95	38 / 35	21 / 33	71 / 72	30 / 28	3.0 / 1.0	5.6 / 3.9	16 / 12
Suffix cascade and counting								
`Suffix cascade`	100 / 100	78 / 76	85 / 86	99 / 99	76 / 78	2.3 / 0.5	9.5 / 7.6	26 / 22
`Tally`	100 / 100	75 / 71	89 / 93	99 / 99	78 / 77	1.2 / 2.2	25 / 13	61 / 34
`Ladder` $w_{\max} = 3$	100 / 100	72 / 72	80 / 90	98 / 99	75 / 73	4.6 / 1.5	25 / 14	61 / 36
`Ladder` $w_{\max} = 3$ + floor	100 / 100	76 / 76	81 / 91	99 / 99	73 / 75	2.8 / 1.1	25 / 13	61 / 35
`Ladder` $w_{\max} = 4$	100 / 100	77 / 73	84 / 87	99 / 99	77 / 76	0.7 / 0.5	25 / 14	62 / 35
`Ladder` $w_{\max} = 4$ + floor	100 / 100	77 / 75	80 / 86	99 / 99	74 / 78	2.4 / 0.1	25 / 13	62 / 36
On `Signed sphere` + `ResidualRace`								
Fixed-length $k = 2$	100 / 100	33 / 35	47 / 63	96 / 95	49 / 53	0.7 / 1.0	0.4 / 0.4	0.4 / 0.3
Fixed-length $k = 1$	100 / 100	75 / 75	74 / 81	98 / 97	72 / 71	0.5 / 0.1	0.3 / 0.7	2.1 / 0.3
Unordered pair	100 / 100	72 / 76	72 / 82	97 / 97	67 / 72	1.2 / 1.0	0.1 / 0.1	1.7 / 0.9
`Global + local` ($k = 1$)	100 / 100	86 / 84	74 / 76	96 / 94	68 / 65	0.5 / 0.7	0.6 / 0.8	2.5 / 2.1
`Global first use` ($k = 1$)	100 / 100	89 / 85	78 / 81	96 / 96	73 / 70	0.7 / 1.8	0.1 / 0.3	1.1 / 1.4

Table 8: Watermark score (App. A.4). Each cell reports identity grouping / `Lexical group`.

		Robustness TPR@1 ↑				Quality deviation (%) ↓		
Variant	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
Core (fixed $k = 2$, `Gumbel race`, `Gamma`)	100 / 100	56 / 54	67 / 76	97 / 98	64 / 68	0.1 / 1.1	4.5 / 4.6	16 / 16
`Prelude`								
On the core	100 / 100	54 / 52	67 / 74	96 / 97	62 / 63	3.2 / 2.6	2.4 / 2.8	11 / 11
On `Ladder` $w_{\max} = 4$ + floor	100 / 100	74 / 70	78 / 84	99 / 98	74 / 73	3.2 / 1.0	6.2 / 5.3	18 / 15
`Gaussian copula`								
$\rho = 0.7$	99 / 100	23 / 20	29 / 39	83 / 84	34 / 37	0.3 / 0.4	0.3 / 0.6	1.7 / 3.5
$\rho = 0.8$	100 / 100	34 / 33	42 / 56	89 / 91	45 / 50	0.9 / 1.2	0.5 / 0.4	4.6 / 4.1
$\rho = 0.85$	100 / 100	36 / 38	50 / 63	93 / 95	49 / 55	0.1 / 3.0	1.4 / 0.9	6.5 / 5.5
$\rho = 0.9$	100 / 100	44 / 44	57 / 67	94 / 95	54 / 59	1.7 / 0.1	1.0 / 2.0	7.1 / 8.3
$\rho = 0.95$	100 / 100	52 / 52	63 / 71	97 / 97	62 / 62	0.6 / 1.4	2.2 / 2.5	9.7 / 10
`Gaussian copula` + common-pair filter								
$\rho = 0.7$	99 / 100	20 / 19	28 / 38	80 / 84	34 / 35	1.6 / 0.3	0.1 / 0.1	2.2 / 2.5
$\rho = 0.8$	100 / 100	32 / 35	39 / 52	90 / 92	43 / 48	1.1 / 0.7	0.3 / 0.9	3.5 / 4.6
$\rho = 0.85$	100 / 100	38 / 36	47 / 60	92 / 92	49 / 53	1.3 / 1.4	0.6 / 1.5	4.7 / 6.7
$\rho = 0.9$	100 / 100	44 / 43	50 / 64	94 / 95	53 / 55	0.3 / 0.8	0.8 / 1.5	5.9 / 7.0
$\rho = 0.95$	100 / 100	48 / 50	59 / 72	96 / 98	61 / 64	0.1 / 1.0	1.2 / 2.3	7.3 / 9.4
$\rho = 1$	100 / 100	57 / 56	64 / 75	97 / 98	63 / 68	2.4 / 1.6	3.3 / 4.0	13 / 14
`Private sampling`								
$r = 0.1$	100 / 100	47 / 45	58 / 67	95 / 97	57 / 62	0.9 / 1.3	2.2 / 2.6	9.7 / 10
$r = 0.2$	100 / 100	42 / 38	52 / 62	93 / 93	49 / 57	2.5 / 1.6	1.1 / 1.5	6.6 / 7.1
$r = 0.3$	99 / 99	30 / 30	39 / 52	87 / 89	44 / 48	4.0 / 3.5	0.6 / 0.9	4.6 / 5.1
`Tail indicator`								
`Tail indicator`	99 / 99	27 / 26	42 / 48	84 / 86	39 / 42	2.2 / 1.3	0.3 / 0.7	4.0 / 4.6
Independent `Channels`								
$L = 4$	100 / 100	51 / 47	60 / 71	96 / 97	60 / 61	2.0 / 1.4	2.3 / 1.9	8.1 / 6.9
$L = 16$	100 / 100	39 / 38	56 / 63	96 / 94	48 / 52	2.3 / 0.2	0.3 / 0.0	2.1 / 1.0
$L = 32$	100 / 100	36 / 34	48 / 62	94 / 95	46 / 50	0.0 / 1.1	0.5 / 0.3	0.4 / 0.2
$L = 64$	100 / 100	32 / 30	46 / 57	92 / 94	42 / 49	0.3 / 1.5	0.6 / 0.5	0.8 / 0.7
$L = 128$	100 / 100	29 / 28	41 / 57	91 / 92	42 / 48	0.8 / 0.4	0.4 / 0.6	0.7 / 1.3
Correlated `Channels`								
`Latin` $L = 4$	100 / 100	50 / 48	64 / 72	96 / 97	56 / 61	2.0 / 2.9	1.6 / 1.5	6.5 / 5.8
`Antithetic` $L = 4$	100 / 100	46 / 43	58 / 70	97 / 95	60 / 59	0.3 / 0.4	1.6 / 0.6	6.6 / 4.3
`Latin` $L = 16$	100 / 100	40 / 40	51 / 64	95 / 96	50 / 52	0.4 / 0.6	0.2 / 0.1	0.5 / 1.2
`Stratified` $L = 16$	100 / 100	40 / 37	52 / 64	94 / 94	50 / 48	1.0 / 1.7	0.1 / 0.3	1.0 / 0.0
`Gaussian direction` + `Gumbel race`								
$d = 1$	100 / 100	36 / 37	50 / 59	95 / 95	52 / 53	1.4 / 0.5	2.8 / 2.4	11 / 9.0
$d = 2$	100 / 100	28 / 28	41 / 50	92 / 93	46 / 47	0.2 / 0.5	0.5 / 0.6	3.5 / 3.1
$d = 4$	100 / 100	21 / 18	31 / 44	89 / 90	37 / 38	2.3 / 1.4	0.8 / 0.5	0.6 / 0.3
$d = 8$	100 / 100	15 / 15	24 / 37	83 / 86	31 / 33	1.8 / 1.0	0.1 / 0.7	0.0 / 0.7
`Gaussian direction` + `ResidualRace`								
$d = 1$	100 / 100	44 / 43	58 / 72	97 / 98	55 / 62	1.5 / 2.1	5.4 / 3.6	18 / 13
$d = 2$	100 / 100	34 / 33	48 / 65	96 / 96	51 / 55	0.9 / 1.1	1.1 / 1.1	5.1 / 5.1
$d = 4$	100 / 100	29 / 27	39 / 53	94 / 95	45 / 50	0.2 / 0.8	0.2 / 0.3	1.3 / 0.7
$d = 8$	100 / 100	20 / 20	32 / 48	91 / 92	38 / 41	1.5 / 2.7	0.9 / 0.6	0.9 / 0.3
`Spherical` + `Gumbel race`								
`Signed sphere`	100 / 100	29 / 28	41 / 57	90 / 94	43 / 46	2.2 / 1.2	0.9 / 1.0	1.6 / 1.9
`Unoriented axis`	100 / 100	30 / 27	41 / 56	91 / 92	45 / 45	2.4 / 0.5	0.2 / 0.5	0.2 / 0.7
`Complex ray`	100 / 100	24 / 23	34 / 50	88 / 91	39 / 42	0.0 / 1.6	1.5 / 0.5	2.7 / 0.8
`Rotation`	100 / 100	25 / 24	39 / 51	89 / 91	39 / 45	1.1 / 2.1	0.9 / 1.1	1.5 / 2.3
`Spherical` + `ResidualRace`								
`Signed sphere`	100 / 100	33 / 35	47 / 63	96 / 95	49 / 53	0.7 / 1.0	0.4 / 0.4	0.4 / 0.3
`Unoriented axis`	100 / 100	34 / 33	48 / 62	94 / 95	46 / 53	3.3 / 0.1	0.6 / 0.7	1.1 / 1.1
`Complex ray`	100 / 100	29 / 28	44 / 56	93 / 94	45 / 48	0.4 / 0.0	1.1 / 1.0	2.1 / 1.7
`Rotation`	100 / 100	33 / 32	47 / 63	94 / 95	47 / 52	1.0 / 1.2	0.4 / 0.9	0.5 / 1.7

Table 9: Logits transformation (App. A.5). Each cell reports identity grouping / `Lexical group`.

Variant	Clean TPR@1 ↑	Robustness TPR@1 ↑				Quality deviation (%) ↓		
		Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
Core (fixed $k = 2$, `Gumbel race`, `Gamma`)	100 / 100	56 / 54	67 / 76	97 / 98	64 / 68	0.1 / 1.1	4.5 / 4.6	16 / 16
Races with private randomness								
`BandRace`	98 / 98	16 / 16	20 / 29	73 / 75	29 / 29	2.0 / 0.7	0.3 / 0.1	1.7 / 2.5
`PoolRace` ($n = 16$)	100 / 100	33 / 27	44 / 50	93 / 92	46 / 52	3.2 / 0.2	0.8 / 1.0	5.3 / 5.6
`MomentBridge` ($K = 8$)	88 / 88	3 / 4	6 / 7	37 / 36	8 / 11	1.9 / 1.2	0.5 / 0.8	0.2 / 0.7
Quantile								
`QuantileRace`	93 / 91	4 / 5	7 / 12	44 / 46	12 / 13	1.4 / 0.3	2.6 / 1.6	10 / 7.5
+ `Wheel`	85 / 86	0 / 0	1 / 2	19 / 21	3 / 4	2.7 / 0.8	0.5 / 0.7	1.3 / 1.4
Transport (L independent channels)								
`TransportRace` $L = 32$	99 / 99	14 / 11	26 / 35	83 / 85	31 / 32	0.3 / 0.5	0.3 / 0.6	0.7 / 1.4
`TransportRace` $L = 128$	100 / 100	23 / 20	32 / 48	87 / 88	38 / 40	1.8 / 0.9	0.8 / 1.2	2.1 / 2.5
`QuotaTransport` $L = 32$	99 / 100	19 / 18	30 / 40	87 / 87	37 / 35	2.2 / 0.2	1.1 / 1.0	2.4 / 1.9
`QuotaTransport` $L = 128$	100 / 100	22 / 20	34 / 48	88 / 90	41 / 39	0.3 / 0.3	0.6 / 1.4	1.4 / 2.8
Residual clocks								
`ResidualRace`	100 / 100	63 / 63	72 / 82	98 / 99	70 / 74	1.5 / 0.9	9.0 / 6.3	27 / 21
Skeleton								
`Last important` + `Content gate`	99 / 98	54 / 47	26 / 30	84 / 80	46 / 44	0.5 / 0.6	1.0 / 0.1	3.0 / 1.0
`Interleaved` + `Interleaved gate`	99 / 98	49 / 45	42 / 46	88 / 85	48 / 46	0.3 / 1.3	2.3 / 1.4	6.0 / 4.1

Table 10: Detection (App. A.6), on a fixed set of generated texts: only the detector changes, so the quality is that of the corresponding generation. Each cell reports identity grouping / `Lexical group`.

Variant	Clean TPR@1 ↑	Robustness TPR@1 ↑				Quality deviation (%) ↓		
		Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
Core texts, no forward pass								
`Gamma`	100 / 100	56 / 54	67 / 76	97 / 98	64 / 68	0.1 / 1.1	4.5 / 4.6	16 / 16
`Gaussian`	100 / 100	40 / 40	51 / 64	95 / 96	54 / 57	0.1 / 1.1	4.5 / 4.6	16 / 16
`Clustering`	100 / 100	57 / 55	67 / 77	97 / 98	64 / 68	0.1 / 1.1	4.5 / 4.6	16 / 16
`Excess`	100 / 100	57 / 55	68 / 77	97 / 98	64 / 68	0.1 / 1.1	4.5 / 4.6	16 / 16
Core texts, forward pass								
`Weighted Gamma`	100 / 100	59 / 56	73 / 81	99 / 100	73 / 79	0.1 / 1.1	4.5 / 4.6	16 / 16
`Position evidence`	100 / 100	57 / 56	70 / 80	98 / 99	68 / 74	0.1 / 1.1	4.5 / 4.6	16 / 16
+ weighting	100 / 100	60 / 60	76 / 85	99 / 100	76 / 82	0.1 / 1.1	4.5 / 4.6	16 / 16
Independent `Channels`								
$L = 4$, Bonferroni	100 / 100	51 / 47	60 / 71	96 / 97	60 / 61	2.0 / 1.4	2.3 / 1.9	8.1 / 6.9
$L = 4$, Šidák	100 / 100	51 / 47	60 / 71	96 / 97	60 / 61	2.0 / 1.4	2.3 / 1.9	8.1 / 6.9
$L = 16$, Bonferroni	100 / 100	39 / 38	56 / 63	96 / 94	48 / 52	2.3 / 0.2	0.3 / 0.0	2.1 / 1.0
$L = 16$, Šidák	100 / 100	39 / 38	56 / 63	96 / 94	48 / 52	2.3 / 0.2	0.3 / 0.0	2.1 / 1.0
$L = 32$, Bonferroni	100 / 100	36 / 34	48 / 62	94 / 95	46 / 50	0.0 / 1.1	0.5 / 0.3	0.4 / 0.2
$L = 32$, Šidák	100 / 100	36 / 34	48 / 62	94 / 95	46 / 50	0.0 / 1.1	0.5 / 0.3	0.4 / 0.2
$L = 64$, Bonferroni	100 / 100	32 / 30	46 / 57	92 / 94	42 / 49	0.3 / 1.5	0.6 / 0.5	0.8 / 0.7
$L = 64$, Šidák	100 / 100	32 / 30	46 / 57	92 / 94	42 / 49	0.3 / 1.5	0.6 / 0.5	0.8 / 0.7
$L = 128$, Bonferroni	100 / 100	29 / 28	41 / 57	91 / 92	42 / 48	0.8 / 0.4	0.4 / 0.6	0.7 / 1.3
$L = 128$, Šidák	100 / 100	29 / 28	41 / 57	91 / 92	42 / 48	0.8 / 0.4	0.4 / 0.6	0.7 / 1.3
`Signed sphere` + `ResidualRace` texts								
`Predictive Gamma`	100 / 100	33 / 35	47 / 63	96 / 95	49 / 53	0.7 / 1.0	0.4 / 0.4	0.4 / 0.3
`Resultant length`	100 / 100	19 / 19	27 / 43	91 / 92	37 / 40	0.7 / 1.0	0.4 / 0.4	0.4 / 0.3
`Spherical caps`	100 / 100	22 / 24	31 / 50	89 / 91	37 / 40	0.7 / 1.0	0.4 / 0.4	0.4 / 0.3
Length + caps	100 / 100	24 / 26	35 / 53	92 / 93	40 / 44	0.7 / 1.0	0.4 / 0.4	0.4 / 0.3
`Anchor distance`	100 / 100	32 / 32	43 / 59	94 / 94	44 / 49	0.7 / 1.0	0.4 / 0.4	0.4 / 0.3
`Gaussian direction` + `Gumbel race` texts								
`Gaussian norm` $d = 1$	100 / 100	36 / 37	50 / 59	95 / 95	52 / 53	1.4 / 0.5	2.8 / 2.4	11 / 9.0
`Residual energy` $d = 1$	100 / 100	50 / 51	65 / 70	97 / 97	61 / 63	1.4 / 0.5	2.8 / 2.4	11 / 9.0
`Gaussian norm` $d = 2$	100 / 100	28 / 28	41 / 50	92 / 93	46 / 47	0.2 / 0.5	0.5 / 0.6	3.5 / 3.1
`Residual energy` $d = 2$	100 / 100	36 / 36	49 / 58	93 / 94	51 / 52	0.2 / 0.5	0.5 / 0.6	3.5 / 3.1
`Circle search` $d = 2$	100 / 100	43 / 41	55 / 65	96 / 96	56 / 57	0.2 / 0.5	0.5 / 0.6	3.5 / 3.1
`Gaussian norm` $d = 4$	100 / 100	21 / 18	31 / 44	89 / 90	37 / 38	2.3 / 1.4	0.8 / 0.5	0.6 / 0.3
`Residual energy` $d = 4$	100 / 100	24 / 22	35 / 47	89 / 90	40 / 40	2.3 / 1.4	0.8 / 0.5	0.6 / 0.3
`Gaussian norm` $d = 8$	100 / 100	15 / 15	24 / 37	83 / 86	31 / 33	1.8 / 1.0	0.1 / 0.7	0.0 / 0.7
`Residual energy` $d = 8$	100 / 100	16 / 14	24 / 36	82 / 85	31 / 31	1.8 / 1.0	0.1 / 0.7	0.0 / 0.7

Table 11: Per-key results of the selected agent schemes of Table 1. Each row reports the results of one of the 5 watermark keys, and each cell reports English / Chinese results. Table 1 reports, for each column, the worst TPR and the median quality deviation over these 5 keys.

			Robustness TPR@1 ↑				Quality deviation (%) ↓		
Scheme	Key	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
Llama-3.1-8B									
AAR	1	92 / 92	10 / 18	12 / 24	70 / 18	66 / 44	1.9 / 2.2	4.7 / 6.1	11 / 14
	2	92 / 89	10 / 18	11 / 22	71 / 17	64 / 45	0.1 / 1.7	5.3 / 6.2	12 / 15
	3	92 / 89	11 / 16	11 / 23	70 / 17	64 / 43	1.9 / 3.0	5.6 / 6.3	13 / 15
	4	92 / 90	8 / 17	12 / 26	70 / 19	64 / 42	2.9 / 1.5	5.2 / 6.4	12 / 15
	5	92 / 90	10 / 19	12 / 24	72 / 18	64 / 45	2.5 / 1.7	5.1 / 6.5	11 / 15
SynthID	1	94 / 91	18 / 27	17 / 32	77 / 28	72 / 53	1.1 / 0.3	2.7 / 3.0	6.7 / 7.7
	2	95 / 91	16 / 25	16 / 34	78 / 26	72 / 53	1.6 / 0.4	3.1 / 3.3	7.3 / 8.0
	3	94 / 92	17 / 26	16 / 31	79 / 24	74 / 55	1.3 / 0.5	3.0 / 2.6	6.9 / 6.7
	4	95 / 90	16 / 28	14 / 31	76 / 26	73 / 55	0.4 / 0.0	3.6 / 2.9	8.5 / 7.7
	5	95 / 90	16 / 27	14 / 35	79 / 25	72 / 53	1.0 / 1.0	2.9 / 2.9	7.0 / 7.3
TextSeal (high)	1	95 / 94	21 / 32	15 / 38	81 / 30	75 / 57	1.5 / 4.4	2.6 / 3.6	6.1 / 8.6
	2	96 / 91	20 / 31	15 / 41	78 / 28	74 / 60	0.7 / 0.7	3.0 / 3.5	6.6 / 8.6
	3	95 / 92	20 / 31	17 / 40	79 / 30	75 / 58	1.7 / 1.5	3.0 / 3.1	6.8 / 7.7
	4	95 / 92	17 / 32	19 / 39	79 / 31	76 / 56	3.6 / 0.9	2.4 / 3.5	5.9 / 8.7
	5	95 / 92	19 / 35	17 / 40	79 / 29	76 / 59	1.3 / 0.4	2.8 / 3.4	6.2 / 8.1
ConcordMark (low)	1	93 / 89	38 / 54	43 / 63	79 / 42	75 / 66	1.6 / 1.8	1.7 / 2.7	3.8 / 5.7
	2	93 / 91	40 / 55	45 / 66	81 / 39	78 / 67	1.3 / 0.2	2.0 / 2.5	4.4 / 5.2
	3	92 / 90	37 / 57	42 / 66	78 / 42	74 / 66	0.1 / 1.4	1.8 / 2.3	4.1 / 4.8
	4	93 / 90	37 / 56	42 / 65	81 / 37	77 / 65	0.3 / 0.6	2.1 / 1.8	4.5 / 4.2
	5	94 / 89	35 / 58	43 / 62	79 / 40	75 / 65	1.1 / 0.7	2.3 / 2.1	4.8 / 4.4
ConcordMark (high)	1	96 / 91	42 / 61	50 / 70	87 / 58	83 / 78	1.6 / 1.8	1.7 / 2.7	3.8 / 5.7
	2	97 / 92	44 / 61	53 / 70	89 / 55	86 / 77	1.3 / 0.2	2.0 / 2.5	4.4 / 5.2
	3	96 / 92	41 / 64	49 / 74	87 / 56	84 / 77	0.1 / 1.4	1.8 / 2.3	4.1 / 4.8
	4	97 / 92	41 / 63	52 / 70	89 / 56	84 / 77	0.3 / 0.6	2.1 / 1.8	4.5 / 4.2
	5	97 / 91	39 / 63	51 / 69	87 / 56	86 / 77	1.1 / 0.7	2.3 / 2.1	4.8 / 4.4
RotationFirst	1	89 / 89	51 / 60	38 / 62	70 / 38	69 / 64	0.7 / 0.2	0.6 / 0.7	1.6 / 1.4
	2	90 / 89	48 / 60	40 / 64	71 / 40	67 / 61	2.0 / 0.1	0.6 / 0.8	1.3 / 1.9
	3	88 / 90	51 / 62	40 / 64	73 / 38	67 / 66	0.9 / 0.6	0.8 / 1.2	1.6 / 2.6
	4	88 / 89	50 / 60	39 / 63	72 / 40	67 / 61	0.7 / 0.1	0.5 / 0.7	1.2 / 1.7
	5	88 / 89	49 / 63	38 / 63	70 / 37	66 / 61	0.4 / 0.6	0.5 / 0.5	1.2 / 1.3
Qwen3.8-27B									
AAR	1	92 / 86	7 / 8	11 / 12	65 / 18	76 / 67	0.9 / 0.6	6.5 / 7.4	13 / 16
	2	91 / 84	8 / 8	11 / 13	66 / 18	76 / 66	2.5 / 0.2	6.0 / 7.3	12 / 16
	3	92 / 85	7 / 8	10 / 13	64 / 17	76 / 67	0.7 / 1.6	6.1 / 7.2	13 / 16
	4	92 / 85	9 / 6	10 / 11	65 / 17	78 / 66	1.0 / 0.8	6.6 / 7.3	14 / 16
	5	92 / 86	8 / 9	9 / 15	67 / 15	77 / 65	0.9 / 0.2	6.2 / 6.6	13 / 14
SynthID	1	93 / 88	13 / 13	14 / 16	74 / 26	77 / 74	1.2 / 1.1	3.1 / 3.5	6.6 / 8.1
	2	94 / 88	14 / 11	12 / 16	76 / 22	76 / 73	2.1 / 0.7	2.9 / 3.8	6.3 / 8.6
	3	94 / 88	12 / 10	13 / 17	74 / 21	78 / 72	0.1 / 1.4	2.8 / 4.0	6.0 / 9.0
	4	95 / 88	13 / 14	12 / 20	78 / 28	78 / 73	0.1 / 0.5	2.7 / 3.3	6.0 / 7.6
	5	94 / 88	11 / 10	14 / 17	74 / 26	78 / 74	2.8 / 1.6	3.3 / 3.5	7.0 / 8.0
TextSeal (high)	1	95 / 88	16 / 12	15 / 19	76 / 26	78 / 75	1.0 / 1.2	2.9 / 3.2	6.1 / 7.2
	2	95 / 89	14 / 11	13 / 21	76 / 28	79 / 74	0.9 / 0.7	2.8 / 3.4	5.9 / 7.5
	3	94 / 89	15 / 13	13 / 20	75 / 28	79 / 75	0.7 / 0.7	2.7 / 3.1	5.8 / 7.1
	4	95 / 90	15 / 10	14 / 17	74 / 26	77 / 74	1.8 / 0.6	2.1 / 3.3	4.9 / 7.5
	5	94 / 89	14 / 11	14 / 20	75 / 27	78 / 74	1.0 / 0.6	2.7 / 2.9	5.8 / 6.5
ConcordMark (low)	1	92 / 86	25 / 23	34 / 35	75 / 34	77 / 73	1.4 / 0.7	1.9 / 1.8	4.0 / 3.6
	2	92 / 86	28 / 25	34 / 34	75 / 33	76 / 72	3.2 / 0.1	1.9 / 1.7	4.1 / 3.6
	3	92 / 85	26 / 25	32 / 36	75 / 29	76 / 73	1.6 / 0.8	2.0 / 1.8	4.1 / 3.7
	4	92 / 87	30 / 24	35 / 35	74 / 30	78 / 73	0.0 / 0.7	1.9 / 1.6	3.9 / 3.4
	5	93 / 85	29 / 25	36 / 37	78 / 31	76 / 74	0.9 / 0.7	2.2 / 2.0	4.4 / 4.1
ConcordMark (high)	1	97 / 91	32 / 32	41 / 44	85 / 52	85 / 82	1.4 / 0.7	1.9 / 1.8	4.0 / 3.6
	2	96 / 90	33 / 33	40 / 44	85 / 51	84 / 83	3.2 / 0.1	1.9 / 1.7	4.1 / 3.6
	3	97 / 90	32 / 31	41 / 44	85 / 48	84 / 82	1.6 / 0.8	2.0 / 1.8	4.1 / 3.7
	4	96 / 91	35 / 33	42 / 43	86 / 47	86 / 82	0.0 / 0.7	1.9 / 1.6	3.9 / 3.4
	5	96 / 91	35 / 32	44 / 43	87 / 51	83 / 82	0.9 / 0.7	2.2 / 2.0	4.4 / 4.1
RotationFirst	1	85 / 82	38 / 24	30 / 37	65 / 23	70 / 65	0.9 / 0.1	0.3 / 0.0	0.8 / 0.3
	2	85 / 83	35 / 25	29 / 39	63 / 22	68 / 67	2.0 / 0.8	0.6 / 0.4	1.1 / 1.1
	3	85 / 82	36 / 25	28 / 38	63 / 25	72 / 65	2.3 / 0.2	0.4 / 0.5	0.9 / 1.3
	4	84 / 81	35 / 24	27 / 35	61 / 24	69 / 64	0.4 / 0.9	0.1 / 0.7	0.4 / 1.6
	5	85 / 82	36 / 25	28 / 37	62 / 23	68 / 66	2.9 / 0.5	0.4 / 0.5	0.9 / 1.4

Table 12: Per-key results of every submission of Figure 2, in submission order. Each row reports the results of one of the 5 watermark keys, with the same columns and settings as Table 6, which reports, for each column, the worst TPR and the median quality deviation over these keys. Each TPR cell reports the guaranteed / empirical / no-correction settings of App. C, and parentheses mark values without correction for schemes that then fail our soundness given a key test.

				Robustness TPR@1 ↑				Quality deviation (%) ↓		
#	Scheme	Key	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
Baselines										
	TextSeal	1	100 / 100 / 100	14 / 37 / 37	31 / 57 / 57	90 / 97 / 97	41 / 60 / 60	2.1	0.2	2.5
		2	100 / 100 / 100	17 / 36 / 36	31 / 56 / 56	91 / 98 / 98	40 / 61 / 61	2.5	0.5	5.8
		3	100 / 100 / 100	14 / 34 / 34	32 / 60 / 60	91 / 97 / 97	39 / 58 / 58	0.4	1.4	5.1
		4	100 / 100 / 100	14 / 34 / 34	28 / 54 / 54	90 / 97 / 97	39 / 60 / 60	3.5	0.1	2.9
		5	100 / 100 / 100	15 / 35 / 35	28 / 55 / 55	89 / 97 / 97	38 / 60 / 60	4.3	2.3	7.3
	AAR	1	100 / 100 / (100)	4 / 19 / (19)	8 / 32 / (33)	77 / 92 / (92)	21 / 45 / (45)	2.7	2.4	9.8
		2	100 / 100 / (100)	4 / 19 / (19)	9 / 33 / (33)	82 / 95 / (95)	25 / 46 / (46)	2.7	1.8	9.5
		3	100 / 100 / (100)	4 / 20 / (20)	9 / 35 / (35)	79 / 94 / (94)	19 / 41 / (41)	0.4	2.9	9.8
		4	100 / 100 / (100)	4 / 19 / (19)	11 / 32 / (32)	77 / 93 / (93)	24 / 49 / (49)	2.5	1.9	8.3
		5	100 / 100 / (100)	5 / 20 / (20)	9 / 30 / (30)	76 / 93 / (93)	24 / 48 / (48)	3.9	3.9	12
	SynthID	1	100 / 100 / (100)	12 / 31 / (31)	22 / 51 / (51)	88 / 96 / (96)	40 / 60 / (60)	1.0	0.1	3.7
		2	100 / 100 / (100)	11 / 30 / (30)	22 / 49 / (49)	89 / 95 / (95)	40 / 61 / (61)	1.0	0.3	5.6
		3	100 / 100 / (100)	9 / 30 / (31)	21 / 49 / (50)	90 / 97 / (97)	36 / 56 / (56)	1.6	0.6	3.7
		4	100 / 100 / (100)	9 / 30 / (30)	20 / 48 / (48)	88 / 96 / (96)	39 / 60 / (60)	2.5	0.7	1.4
		5	100 / 100 / (100)	10 / 30 / (30)	20 / 44 / (44)	88 / 96 / (96)	35 / 58 / (58)	1.6	2.0	7.1
GPT-6 ASTRA (xhigh)										
1	CascadeCopula	1	100 / 100 / (100)	61 / 79 / (80)	60 / 78 / (79)	95 / 99 / (99)	58 / 76 / (76)	2.3	1.5	6.1
		2	100 / 100 / (100)	64 / 84 / (84)	70 / 85 / (85)	98 / 99 / (99)	61 / 80 / (80)	0.6	2.3	9.3
		3	100 / 100 / (100)	64 / 83 / (83)	69 / 85 / (85)	96 / 99 / (99)	60 / 80 / (80)	1.3	2.0	6.1
		4	100 / 100 / (100)	62 / 83 / (84)	66 / 85 / (86)	97 / 99 / (99)	69 / 85 / (86)	0.9	2.6	8.0
		5	100 / 100 / (100)	60 / 81 / (81)	62 / 81 / (81)	96 / 99 / (99)	61 / 77 / (78)	0.5	3.9	10
2	LexicalCopula	1	100 / 100 / (100)	60 / 78 / (78)	69 / 83 / (84)	94 / 98 / (98)	63 / 79 / (80)	1.0	1.0	5.1
		2	100 / 100 / (100)	59 / 80 / (80)	75 / 89 / (89)	98 / 99 / (99)	70 / 86 / (86)	0.5	1.8	8.2
		3	100 / 100 / (100)	61 / 84 / (84)	76 / 89 / (89)	97 / 99 / (99)	63 / 78 / (79)	0.3	2.4	6.8
		4	100 / 100 / (100)	61 / 82 / (82)	76 / 89 / (89)	97 / 100 / (100)	72 / 87 / (87)	0.1	2.6	7.9
		5	100 / 100 / (100)	57 / 79 / (79)	70 / 86 / (86)	95 / 98 / (98)	63 / 79 / (80)	0.8	3.4	9.4
3	PortfolioRace	1	100 / 100 / 100	51 / 68 / 68	70 / 80 / 80	96 / 98 / 98	61 / 74 / 74	1.1	1.1	1.3
		2	100 / 100 / 100	54 / 67 / 67	73 / 82 / 82	94 / 98 / 98	61 / 73 / 73	2.1	0.6	1.8
		3	100 / 100 / 100	55 / 71 / 71	75 / 85 / 85	97 / 99 / 99	59 / 71 / 71	0.1	0.3	0.2
		4	100 / 100 / 100	57 / 71 / 71	73 / 84 / 84	97 / 99 / 99	63 / 76 / 76	3.1	1.0	1.4
		5	100 / 100 / 100	53 / 68 / 68	70 / 82 / 82	95 / 98 / 98	59 / 72 / 72	1.7	2.1	4.7
4	TailRace	1	99 / 100 / (100)	44 / 66 / (67)	56 / 73 / (73)	87 / 94 / (94)	50 / 70 / (70)	4.8	0.9	4.0
		2	99 / 100 / (100)	47 / 73 / (73)	64 / 82 / (82)	92 / 98 / (98)	55 / 74 / (74)	0.6	1.7	7.6
		3	100 / 100 / (100)	50 / 74 / (75)	62 / 82 / (82)	92 / 97 / (97)	51 / 72 / (72)	1.7	2.0	5.7
		4	100 / 100 / (100)	45 / 69 / (70)	55 / 76 / (77)	90 / 96 / (96)	55 / 76 / (76)	0.7	2.1	6.1
		5	99 / 100 / (100)	45 / 67 / (67)	59 / 78 / (79)	90 / 97 / (97)	53 / 71 / (71)	2.4	3.1	8.3
5	DirectionalRenewal	1	100 / 100 / 100	69 / 84 / 84	74 / 86 / 86	97 / 99 / 99	64 / 80 / 80	1.4	2.3	6.1
		2	100 / 100 / 100	70 / 85 / 85	74 / 86 / 86	96 / 99 / 99	66 / 80 / 80	1.9	2.5	9.0
		3	100 / 100 / 100	72 / 85 / 85	75 / 88 / 88	96 / 99 / 99	71 / 82 / 82	2.1	2.5	6.5
		4	100 / 100 / 100	67 / 83 / 83	74 / 87 / 87	96 / 99 / 99	65 / 78 / 78	2.7	1.6	4.7
		5	100 / 100 / 100	70 / 84 / 84	73 / 86 / 86	96 / 99 / 99	70 / 81 / 81	4.0	4.0	9.5
6	HaarCap	1	100 / 100 / 100	65 / 80 / 80	69 / 84 / 84	94 / 98 / 98	62 / 76 / 76	0.1	0.9	0.3
		2	100 / 100 / 100	64 / 80 / 80	71 / 85 / 85	96 / 99 / 99	63 / 77 / 77	1.0	0.8	5.9
		3	100 / 100 / 100	66 / 81 / 81	71 / 86 / 86	96 / 98 / 98	67 / 78 / 78	0.6	0.6	2.7
		4	100 / 100 / 100	67 / 83 / 83	72 / 85 / 85	97 / 99 / 99	63 / 77 / 77	0.4	0.6	0.6
		5	100 / 100 / 100	65 / 80 / 80	70 / 85 / 85	94 / 99 / 99	62 / 76 / 76	1.5	1.6	4.8
7	HaarGamma	1	100 / 100 / 100	67 / 80 / 80	74 / 84 / 84	96 / 98 / 98	67 / 77 / 77	0.2	0.4	1.3
		2	100 / 100 / 100	70 / 81 / 81	77 / 86 / 86	97 / 99 / 99	69 / 79 / 79	1.5	0.6	5.4
		3	100 / 100 / 100	72 / 83 / 83	78 / 85 / 85	96 / 98 / 98	69 / 78 / 78	1.5	0.5	2.3
		4	100 / 100 / 100	70 / 81 / 81	76 / 85 / 85	96 / 98 / 98	68 / 77 / 77	0.5	1.0	0.2
		5	100 / 100 / 100	68 / 80 / 80	74 / 84 / 84	96 / 98 / 98	66 / 76 / 76	0.5	2.3	6.7
8	HaarPredictive	1	100 / 100 / (100)	72 / 86 / (86)	78 / 88 / (88)	98 / 99 / (99)	68 / 80 / (80)	0.5	0.6	0.8
		2	100 / 100 / (100)	72 / 85 / (85)	80 / 90 / (90)	97 / 99 / (99)	71 / 84 / (84)	1.8	0.2	4.5
		3	100 / 100 / (100)	71 / 86 / (86)	77 / 89 / (89)	96 / 98 / (98)	70 / 82 / (82)	1.8	0.3	2.1
		4	100 / 100 / (100)	72 / 85 / (85)	80 / 89 / (89)	97 / 99 / (99)	73 / 83 / (83)	1.0	0.3	0.9
		5	100 / 100 / (100)	74 / 86 / (86)	77 / 88 / (88)	97 / 98 / (98)	72 / 84 / (84)	0.3	2.6	7.0
9	HaarGraph	1	100 / 100 / 100	73 / 86 / 86	79 / 89 / 89	97 / 99 / 99	72 / 83 / 83	1.0	0.7	0.4
		2	100 / 100 / 100	73 / 86 / 86	78 / 88 / 88	96 / 99 / 99	69 / 82 / 82	0.6	1.3	1.3
		3	100 / 100 / 100	72 / 86 / 86	79 / 88 / 88	96 / 98 / 98	71 / 82 / 82	1.8	0.0	1.4
		4	100 / 100 / 100	75 / 86 / 86	80 / 89 / 89	97 / 99 / 99	73 / 84 / 84	3.2	1.1	0.4
		5	100 / 100 / 100	72 / 86 / 86	77 / 89 / 89	97 / 99 / 99	70 / 82 / 82	0.4	2.0	5.5
10	HaarLexical	1	99 / 100 / (100)	84 / 93 / (93)	76 / 86 / (86)	93 / 98 / (98)	66 / 79 / (79)	0.5	0.2	0.4
		2	99 / 100 / (100)	84 / 92 / (92)	78 / 88 / (88)	94 / 97 / (97)	68 / 81 / (81)	0.1	0.2	2.5
		3	100 / 100 / (100)	84 / 92 / (92)	80 / 90 / (90)	94 / 98 / (98)	70 / 82 / (82)	2.1	0.5	1.7
		4	99 / 99 / (99)	84 / 92 / (92)	76 / 87 / (87)	93 / 98 / (98)	67 / 82 / (82)	0.1	0.9	1.1
		5	99 / 100 / (100)	83 / 92 / (92)	76 / 89 / (89)	92 / 97 / (97)	67 / 82 / (82)	1.3	2.0	5.4

Table 13: Per-key results of every submission of Figure 2 (continued from Table 12).

#	Scheme	Key	Clean TPR@1 ↑	Robustness TPR@1 ↑				Quality deviation (%) ↓		
				Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
GPT-6 ASTRA (xhigh) (continued)										
11	HaarLexicalFirst	1	100 / 100 / (100)	86 / 94 / (94)	81 / 90 / (90)	96 / 99 / (99)	71 / 83 / (83)	1.7	0.4	0.5
		2	100 / 100 / (100)	85 / 91 / (91)	81 / 90 / (90)	96 / 98 / (98)	68 / 83 / (83)	1.4	0.2	2.9
		3	100 / 100 / (100)	86 / 93 / (93)	82 / 91 / (91)	95 / 98 / (98)	73 / 85 / (85)	2.4	0.6	1.8
		4	100 / 100 / (100)	86 / 93 / (93)	81 / 91 / (91)	95 / 98 / (98)	73 / 85 / (85)	0.3	1.0	1.1
		5	99 / 100 / (100)	84 / 93 / (93)	82 / 89 / (89)	94 / 98 / (98)	69 / 82 / (82)	1.4	2.2	5.7
12	ProjectiveFirst	1	99 / 100 / 100	83 / 92 / 92	78 / 89 / 89	94 / 98 / 98	66 / 80 / 80	0.7	0.9	1.4
		2	99 / 100 / 100	81 / 91 / 91	75 / 86 / 86	93 / 98 / 98	68 / 81 / 81	0.4	0.6	2.0
		3	99 / 100 / 100	82 / 91 / 91	79 / 88 / 88	94 / 98 / 98	68 / 82 / 82	1.7	0.5	0.8
		4	99 / 100 / 100	82 / 91 / 91	76 / 87 / 87	93 / 97 / 97	65 / 79 / 79	0.2	1.9	3.2
		5	99 / 100 / 100	83 / 92 / 92	78 / 88 / 88	93 / 97 / 97	64 / 78 / 78	1.0	1.7	4.2
13	AxialFirst	1	100 / 100 / (100)	83 / 92 / (92)	78 / 89 / (89)	95 / 98 / (98)	70 / 83 / (83)	2.4	0.8	1.0
		2	100 / 100 / (100)	86 / 93 / (93)	83 / 91 / (91)	96 / 98 / (98)	73 / 85 / (85)	0.4	0.3	2.4
		3	99 / 100 / (100)	84 / 92 / (92)	79 / 90 / (90)	95 / 98 / (98)	71 / 84 / (84)	0.4	0.0	0.5
		4	100 / 100 / (100)	86 / 94 / (94)	79 / 89 / (89)	95 / 99 / (99)	72 / 85 / (85)	3.1	0.7	0.7
		5	99 / 100 / (100)	84 / 93 / (93)	80 / 88 / (88)	95 / 98 / (98)	70 / 83 / (83)	0.1	2.3	5.7
14	RotationFirst	1	99 / 100 / 100	85 / 92 / 92	81 / 90 / 90	95 / 98 / 98	71 / 83 / 83	0.8	1.0	1.0
		2	100 / 100 / 100	82 / 92 / 92	77 / 87 / 87	95 / 98 / 98	68 / 82 / 82	0.3	0.2	2.5
		3	100 / 100 / 100	85 / 93 / 93	79 / 90 / 90	96 / 98 / 98	69 / 82 / 82	0.0	0.1	0.4
		4	99 / 100 / 100	82 / 90 / 90	77 / 86 / 86	93 / 97 / 97	68 / 80 / 80	1.0	1.2	1.9
		5	99 / 100 / 100	81 / 91 / 91	78 / 88 / 88	94 / 97 / 97	69 / 81 / 81	0.5	1.6	4.0
GPT-6 ASTRA (low)										
1	FreshRace	1	99 / 100 / (100)	15 / 38 / (38)	18 / 45 / (45)	77 / 91 / (91)	30 / 54 / (54)	0.4	0.3	3.1
		2	100 / 100 / (100)	13 / 37 / (38)	16 / 42 / (42)	78 / 92 / (92)	33 / 57 / (57)	1.4	1.5	6.2
		3	100 / 100 / (100)	14 / 35 / (36)	16 / 44 / (45)	76 / 90 / (90)	28 / 53 / (53)	0.3	0.6	2.2
		4	99 / 100 / (100)	14 / 34 / (35)	14 / 37 / (37)	74 / 90 / (91)	30 / 52 / (52)	4.7	0.9	4.3
		5	100 / 100 / (100)	17 / 42 / (42)	24 / 54 / (55)	78 / 91 / (91)	28 / 50 / (51)	7.2	1.9	7.1
2	FreshRace-selective	1	100 / 100 / (100)	30 / 58 / (59)	36 / 64 / (64)	90 / 97 / (97)	48 / 69 / (70)	0.4	1.0	6.1
		2	100 / 100 / (100)	28 / 54 / (55)	33 / 61 / (62)	92 / 98 / (98)	51 / 71 / (71)	2.0	2.6	9.5
		3	100 / 100 / (100)	26 / 52 / (53)	34 / 60 / (60)	91 / 97 / (97)	44 / 66 / (66)	0.0	1.1	7.2
		4	100 / 100 / (100)	28 / 54 / (55)	28 / 54 / (54)	89 / 96 / (96)	44 / 65 / (65)	3.9	2.7	9.1
		5	100 / 100 / (100)	30 / 57 / (57)	40 / 69 / (69)	91 / 97 / (97)	45 / 65 / (65)	2.4	3.8	12
3	FreshRace-deployment	1	100 / 100 / (100)	44 / 69 / (70)	54 / 75 / (75)	95 / 98 / (98)	58 / 74 / (75)	1.3	0.6	5.7
		2	100 / 100 / (100)	40 / 66 / (66)	48 / 69 / (69)	94 / 99 / (99)	58 / 77 / (77)	0.1	1.7	7.8
		3	100 / 100 / (100)	39 / 61 / (62)	49 / 73 / (73)	93 / 98 / (98)	54 / 72 / (72)	0.8	0.4	6.0
		4	100 / 100 / (100)	41 / 66 / (66)	43 / 69 / (69)	92 / 97 / (97)	53 / 74 / (74)	4.9	2.3	8.7
		5	100 / 100 / (100)	45 / 72 / (73)	57 / 78 / (78)	94 / 98 / (98)	56 / 72 / (73)	3.3	2.9	11
4	FreshRace-channels	1	100 / 100 / 100	46 / 68 / 68	55 / 75 / 75	96 / 99 / 99	59 / 74 / 74	1.7	0.5	4.0
		2	100 / 100 / 100	43 / 63 / 63	56 / 73 / 73	97 / 99 / 99	62 / 77 / 77	1.9	2.2	7.2
		3	100 / 100 / 100	43 / 64 / 64	54 / 72 / 72	96 / 99 / 99	56 / 73 / 73	0.3	0.3	3.5
		4	100 / 100 / 100	38 / 60 / 60	46 / 65 / 65	94 / 98 / 98	55 / 71 / 71	5.4	1.0	4.9
		5	100 / 100 / 100	50 / 72 / 72	60 / 78 / 78	97 / 99 / 99	58 / 73 / 73	8.8	2.1	8.0
5	FreshRace-adaptive-channels	1	100 / 100 / (100)	48 / 70 / (70)	52 / 72 / (72)	91 / 97 / (97)	53 / 70 / (70)	1.9	0.1	0.9
		2	100 / 100 / (100)	49 / 70 / (70)	50 / 70 / (70)	92 / 97 / (97)	56 / 73 / (73)	3.4	1.2	3.7
		3	100 / 100 / (100)	43 / 65 / (65)	48 / 69 / (69)	92 / 97 / (97)	54 / 72 / (72)	0.8	0.1	2.7
		4	100 / 100 / (100)	42 / 65 / (65)	44 / 66 / (67)	91 / 96 / (96)	53 / 71 / (71)	2.6	1.5	3.7
		5	100 / 100 / (100)	49 / 71 / (71)	55 / 74 / (74)	94 / 98 / (98)	53 / 71 / (71)	9.8	1.7	5.3
6	FreshRace-split	1	100 / 100 / (100)	62 / 80 / (80)	63 / 79 / (79)	97 / 99 / (99)	62 / 77 / (77)	2.7	2.1	6.2
		2	100 / 100 / (100)	62 / 80 / (80)	64 / 81 / (81)	97 / 99 / (99)	68 / 82 / (82)	3.7	2.2	6.4
		3	100 / 100 / (100)	59 / 77 / (77)	62 / 78 / (78)	96 / 98 / (98)	63 / 76 / (76)	1.9	0.6	4.7
		4	100 / 100 / (100)	58 / 75 / (75)	60 / 78 / (78)	96 / 99 / (99)	62 / 78 / (78)	3.2	2.2	5.9
		5	100 / 100 / (100)	66 / 81 / (81)	68 / 82 / (82)	98 / 100 / (100)	63 / 77 / (77)	8.0	3.3	9.5
7	FreshRace-antithetic-reservoir	1	100 / 100 / 100	53 / 69 / 69	56 / 72 / 72	94 / 98 / 98	56 / 71 / 71	0.9	0.5	0.0
		2	100 / 100 / 100	54 / 71 / 71	57 / 73 / 73	94 / 97 / 97	60 / 74 / 74	5.2	1.2	3.0
		3	100 / 100 / 100	49 / 67 / 67	54 / 70 / 70	93 / 97 / 97	55 / 69 / 69	1.6	1.2	0.4
		4	100 / 100 / 100	50 / 68 / 68	50 / 68 / 68	95 / 98 / 98	53 / 68 / 68	3.3	0.5	1.6
		5	100 / 100 / 100	53 / 70 / 70	58 / 74 / 74	96 / 99 / 99	55 / 70 / 70	5.4	1.9	5.3
8	FreshRace-balanced	1	100 / 100 / 100	47 / 67 / 67	56 / 75 / 75	97 / 99 / 99	58 / 74 / 74	0.3	0.8	4.0
		2	100 / 100 / 100	45 / 66 / 66	57 / 74 / 74	95 / 98 / 98	59 / 75 / 75	0.5	2.0	6.4
		3	100 / 100 / 100	42 / 61 / 61	51 / 70 / 70	94 / 98 / 98	56 / 70 / 70	0.6	0.2	4.2
		4	100 / 100 / 100	40 / 60 / 60	50 / 67 / 67	94 / 97 / 97	55 / 72 / 72	7.4	1.4	5.4
		5	100 / 100 / 100	48 / 68 / 68	56 / 75 / 75	97 / 99 / 99	61 / 76 / 76	5.7	2.8	9.0
9	FreshRace-lexical	1	100 / 100 / 100	56 / 74 / 74	72 / 86 / 86	96 / 98 / 98	62 / 77 / 77	2.5	0.8	3.4
		2	100 / 100 / 100	58 / 77 / 77	74 / 86 / 86	97 / 99 / 99	68 / 82 / 82	3.5	2.2	5.9
		3	100 / 100 / 100	53 / 73 / 73	69 / 84 / 84	96 / 99 / 99	59 / 75 / 75	1.2	0.4	3.7
		4	100 / 100 / 100	51 / 71 / 71	64 / 81 / 81	95 / 98 / 98	60 / 77 / 77	7.3	1.4	4.2
		5	100 / 100 / 100	57 / 75 / 75	71 / 85 / 85	97 / 99 / 99	59 / 76 / 76	6.9	2.4	7.4
10	FreshRace-lexical	1	100 / 100 / 100	52 / 73 / 73	70 / 85 / 85	96 / 98 / 98	65 / 79 / 79	0.7	1.1	4.1
		2	100 / 100 / 100	55 / 75 / 75	72 / 86 / 86	96 / 99 / 99	67 / 80 / 80	5.2	2.1	6.1
		3	100 / 100 / 100	51 / 70 / 70	66 / 82 / 82	95 / 98 / 98	59 / 75 / 75	0.0	0.3	3.7
		4	100 / 100 / 100	51 / 69 / 69	67 / 81 / 81	97 / 99 / 99	61 / 77 / 77	3.5	1.3	4.4
		5	100 / 100 / 100	58 / 77 / 77	73 / 87 / 87	97 / 99 / 99	60 / 74 / 74	4.7	2.6	7.7
11	RenewalRace	1	97 / 99 / (99)	84 / 94 / (94)	56 / 74 / (77)	86 / 94 / (95)	61 / 76 / (79)	1.1	1.9	2.6
		2	97 / 99 / (99)	80 / 92 / (93)	54 / 71 / (74)	83 / 92 / (93)	53 / 70 / (72)	4.0	2.5	5.1
		3	99 / 100 / (100)	90 / 96 / (97)	67 / 82 / (85)	90 / 96 / (96)	62 / 80 / (82)	4.5	1.5	2.3
		4	95 / 98 / (98)	80 / 90 / (91)	52 / 72 / (75)	82 / 92 / (92)	52 / 69 / (72)	4.7	2.9	5.2
		5	98 / 100 / (100)	87 / 94 / (95)	62 / 80 / (82)	90 / 96 / (96)	64 / 79 / (81)	3.3	3.2	5.8

Table 14: Per-key results of every submission of Figure 2 (continued from Table 13).

#	Scheme	Key	Clean TPR@1 ↑	Robustness TPR@1 ↑				Quality deviation (%) ↓		
				Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
OPUS 5										
1	TallyMark	1	100 / 100 / (100)	37 / 63 / (64)	59 / 81 / (81)	87 / 96 / (96)	49 / 71 / (71)	1.6	1.5	4.6
		2	100 / 100 / (100)	40 / 65 / (67)	61 / 83 / (83)	92 / 98 / (98)	48 / 71 / (72)	2.6	2.1	4.9
		3	100 / 100 / (100)	38 / 64 / (64)	59 / 81 / (81)	91 / 96 / (97)	47 / 68 / (68)	4.1	2.5	7.2
		4	100 / 100 / (100)	39 / 66 / (66)	60 / 81 / (81)	87 / 96 / (96)	50 / 71 / (71)	3.9	3.0	7.9
		5	100 / 100 / (100)	38 / 64 / (65)	57 / 77 / (78)	86 / 94 / (95)	44 / 63 / (64)	0.4	3.1	8.3
2	TallyMark	1	100 / 100 / (100)	32 / 59 / (59)	34 / 60 / (60)	88 / 96 / (96)	48 / 68 / (69)	1.7	0.6	1.9
		2	100 / 100 / (100)	33 / 57 / (57)	35 / 60 / (60)	89 / 96 / (96)	46 / 67 / (67)	1.2	1.7	5.8
		3	100 / 100 / (100)	32 / 57 / (57)	37 / 62 / (62)	87 / 95 / (95)	42 / 64 / (64)	2.6	1.6	7.4
		4	100 / 100 / (100)	33 / 60 / (60)	37 / 64 / (64)	88 / 96 / (96)	47 / 70 / (70)	3.1	1.6	6.7
		5	100 / 100 / (100)	32 / 59 / (59)	36 / 63 / (63)	88 / 95 / (95)	47 / 68 / (68)	1.3	1.3	5.8
3	TallyMark	1	100 / 100 / (100)	40 / 66 / (67)	63 / 81 / (82)	88 / 95 / (95)	47 / 68 / (68)	0.8	2.2	6.7
		2	100 / 100 / (100)	33 / 61 / (62)	60 / 80 / (81)	90 / 97 / (97)	49 / 70 / (71)	0.7	3.4	7.6
		3	100 / 100 / (100)	44 / 69 / (70)	70 / 85 / (86)	92 / 98 / (98)	49 / 72 / (72)	2.7	3.6	10
		4	100 / 100 / (100)	42 / 68 / (68)	64 / 82 / (83)	88 / 95 / (96)	48 / 67 / (67)	5.2	2.7	7.1
		5	100 / 100 / (100)	48 / 72 / (73)	65 / 84 / (84)	93 / 97 / (97)	50 / 71 / (72)	2.8	3.3	8.7
4	TallyMark	1	100 / 100 / (100)	41 / 65 / (66)	65 / 82 / (83)	90 / 97 / (97)	49 / 70 / (71)	1.9	1.5	5.2
		2	100 / 100 / (100)	36 / 64 / (64)	57 / 80 / (80)	88 / 97 / (97)	52 / 71 / (72)	1.9	3.4	7.7
		3	100 / 100 / (100)	44 / 70 / (71)	69 / 85 / (85)	92 / 97 / (97)	50 / 69 / (70)	0.0	3.1	9.2
		4	100 / 100 / (100)	44 / 69 / (69)	67 / 86 / (87)	90 / 98 / (98)	51 / 73 / (74)	2.8	2.6	6.8
		5	100 / 100 / (100)	45 / 70 / (71)	63 / 83 / (84)	91 / 98 / (98)	49 / 69 / (70)	1.5	3.0	7.9
5	ChoirMark	1	100 / 100 / (100)	59 / 76 / (76)	79 / 88 / (88)	98 / 99 / (99)	65 / 79 / (79)	5.7	9.5	25
		2	100 / 100 / (100)	57 / 75 / (75)	78 / 87 / (87)	96 / 99 / (99)	62 / 75 / (75)	3.8	14	34
		3	100 / 100 / (100)	57 / 76 / (76)	78 / 88 / (88)	97 / 99 / (99)	62 / 77 / (77)	1.5	12	31
		4	100 / 100 / (100)	61 / 80 / (80)	81 / 89 / (89)	97 / 99 / (99)	70 / 83 / (83)	5.8	9.8	26
		5	100 / 100 / (100)	63 / 77 / (77)	80 / 90 / (90)	97 / 99 / (99)	66 / 78 / (78)	2.3	11	29
6	CanonMark	1	100 / 100 / 100	46 / 62 / 62	69 / 79 / 79	94 / 98 / 98	56 / 68 / 68	4.1	2.6	4.3
		2	100 / 100 / 100	43 / 59 / 59	66 / 78 / 78	94 / 97 / 97	51 / 64 / 64	0.3	1.7	3.9
		3	100 / 100 / 100	43 / 61 / 61	68 / 79 / 79	93 / 96 / 96	51 / 65 / 65	4.2	0.7	0.0
		4	100 / 100 / 100	48 / 64 / 64	70 / 81 / 81	96 / 98 / 98	62 / 75 / 75	4.0	0.0	1.1
		5	100 / 100 / 100	43 / 60 / 60	68 / 81 / 81	94 / 97 / 97	55 / 69 / 69	2.6	0.2	0.2
7	LadderMark	1	100 / 100 / 100	54 / 71 / 71	71 / 81 / 81	96 / 98 / 98	59 / 72 / 72	3.7	2.7	4.6
		2	100 / 100 / 100	51 / 66 / 66	67 / 78 / 78	96 / 98 / 98	56 / 68 / 68	0.7	0.6	1.9
		3	100 / 100 / 100	55 / 69 / 69	67 / 78 / 78	95 / 98 / 98	57 / 68 / 68	1.9	0.5	0.9
		4	100 / 100 / 100	55 / 68 / 68	67 / 79 / 79	96 / 98 / 98	62 / 75 / 75	3.3	0.1	0.8
		5	100 / 100 / 100	56 / 69 / 69	69 / 79 / 79	96 / 98 / 98	60 / 70 / 70	3.8	1.1	1.9
8	PreludeMark	1	100 / 100 / (100)	79 / 90 / (90)	82 / 92 / (93)	98 / 99 / (99)	75 / 88 / (88)	1.2	4.7	15
		2	100 / 100 / (100)	73 / 88 / (88)	83 / 93 / (93)	99 / 100 / (100)	76 / 88 / (88)	0.5	5.5	16
		3	100 / 100 / (100)	79 / 90 / (91)	82 / 93 / (93)	99 / 100 / (100)	76 / 87 / (87)	5.1	5.8	18
		4	100 / 100 / (100)	77 / 90 / (90)	83 / 93 / (94)	98 / 100 / (100)	80 / 90 / (90)	9.0	5.0	15
		5	100 / 100 / (100)	76 / 88 / (89)	79 / 90 / (91)	98 / 100 / (100)	70 / 84 / (84)	0.6	5.4	16
9	ConsortMark	1	100 / 100 / (100)	66 / 79 / (79)	72 / 84 / (84)	98 / 99 / (99)	65 / 78 / (78)	4.8	0.3	1.6
		2	100 / 100 / (100)	65 / 79 / (79)	71 / 83 / (83)	97 / 99 / (99)	62 / 76 / (76)	0.8	0.8	2.4
		3	100 / 100 / (100)	67 / 80 / (80)	75 / 86 / (86)	98 / 100 / (100)	65 / 78 / (78)	5.5	0.9	4.9
		4	100 / 100 / (100)	65 / 79 / (79)	70 / 82 / (82)	98 / 99 / (99)	70 / 83 / (83)	5.7	1.6	5.1
		5	100 / 100 / (100)	68 / 83 / (83)	74 / 86 / (86)	97 / 99 / (99)	64 / 75 / (75)	4.2	0.3	2.2
10	SestetMark	1	100 / 100 / (100)	70 / 84 / (84)	80 / 88 / (88)	98 / 99 / (99)	72 / 84 / (84)	9.0	0.7	0.4
		2	100 / 100 / (100)	68 / 82 / (82)	78 / 89 / (89)	97 / 99 / (99)	67 / 81 / (81)	0.4	1.0	2.8
		3	100 / 100 / (100)	69 / 83 / (83)	81 / 90 / (90)	98 / 99 / (99)	70 / 81 / (81)	8.4	1.3	5.1
		4	100 / 100 / (100)	66 / 82 / (83)	78 / 86 / (86)	97 / 99 / (99)	69 / 83 / (83)	5.5	1.1	3.9
		5	100 / 100 / (100)	72 / 85 / (85)	81 / 90 / (90)	97 / 99 / (99)	67 / 78 / (78)	4.6	0.8	3.1
11	MotetMark	1	100 / 100 / (100)	73 / 85 / (85)	80 / 90 / (90)	98 / 99 / (99)	67 / 83 / (83)	7.0	0.3	1.6
		2	100 / 100 / (100)	67 / 82 / (82)	77 / 88 / (88)	97 / 99 / (99)	62 / 78 / (78)	1.3	1.2	3.2
		3	100 / 100 / (100)	68 / 82 / (82)	80 / 89 / (89)	97 / 99 / (99)	64 / 79 / (79)	7.2	1.6	5.8
		4	100 / 100 / (100)	66 / 82 / (82)	77 / 87 / (87)	97 / 99 / (99)	70 / 82 / (82)	3.9	1.0	4.1
		5	100 / 100 / (100)	71 / 85 / (85)	80 / 89 / (89)	97 / 99 / (99)	65 / 79 / (79)	6.5	0.8	3.1
12	CodaMark	1	100 / 100 / (100)	72 / 85 / (85)	82 / 90 / (90)	98 / 99 / (99)	68 / 80 / (80)	8.0	0.6	0.9
		2	100 / 100 / (100)	66 / 81 / (81)	79 / 90 / (90)	98 / 99 / (99)	66 / 80 / (80)	0.6	1.1	2.9
		3	100 / 100 / (100)	68 / 84 / (84)	81 / 90 / (90)	98 / 99 / (99)	65 / 79 / (79)	6.3	1.3	5.4
		4	100 / 100 / (100)	66 / 81 / (81)	77 / 86 / (86)	98 / 99 / (99)	70 / 82 / (82)	3.2	1.5	4.7
		5	100 / 100 / (100)	70 / 84 / (84)	80 / 90 / (90)	97 / 99 / (99)	67 / 80 / (80)	6.3	1.0	3.3
13	AccentMark	1	100 / 100 / 100	73 / 85 / 85	85 / 92 / 92	100 / 100 / 100	77 / 87 / 87	8.0	0.7	0.5
		2	100 / 100 / 100	71 / 83 / 83	83 / 91 / 91	99 / 100 / 100	74 / 84 / 84	0.2	1.5	3.8
		3	100 / 100 / 100	72 / 83 / 83	83 / 92 / 92	99 / 100 / 100	75 / 84 / 84	6.5	1.5	6.0
		4	100 / 100 / 100	69 / 82 / 82	83 / 90 / 90	100 / 100 / 100	79 / 88 / 88	2.8	1.3	4.5
		5	100 / 100 / 100	75 / 88 / 88	86 / 94 / 94	100 / 100 / 100	77 / 86 / 86	4.9	1.3	3.7
14	RowMark	1	100 / 100 / 100	75 / 86 / 86	85 / 93 / 93	100 / 100 / 100	78 / 88 / 88	5.9	0.3	1.2
		2	100 / 100 / 100	67 / 84 / 84	83 / 92 / 92	99 / 100 / 100	74 / 86 / 86	1.8	1.5	3.7
		3	100 / 100 / 100	74 / 85 / 85	86 / 94 / 94	100 / 100 / 100	77 / 87 / 87	6.3	1.6	5.8
		4	100 / 100 / 100	70 / 84 / 84	82 / 91 / 91	100 / 100 / 100	80 / 89 / 89	5.9	1.1	3.8
		5	100 / 100 / 100	73 / 86 / 86	86 / 94 / 94	100 / 100 / 100	75 / 86 / 86	3.7	0.5	2.4
15	UnisonMark	1	100 / 100 / 100	72 / 86 / 86	87 / 95 / 95	100 / 100 / 100	78 / 88 / 88	4.6	0.7	0.6
		2	100 / 100 / 100	69 / 82 / 82	84 / 93 / 93	100 / 100 / 100	74 / 85 / 85	0.8	1.3	3.2
		3	100 / 100 / 100	71 / 84 / 84	87 / 94 / 94	99 / 100 / 100	76 / 87 / 87	6.2	1.1	4.9
		4	100 / 100 / 100	68 / 83 / 83	86 / 93 / 93	100 / 100 / 100	80 / 88 / 88	6.4	1.0	3.8
		5	100 / 100 / 100	73 / 86 / 86	89 / 94 / 94	100 / 100 / 100	77 / 87 / 87	5.3	1.0	3.4

Table 15: Per-key results of every submission of Figure 2 (continued from Table 14).

#	Scheme	Key	Clean TPR@1 ↑	Robustness TPR@1 ↑				Quality deviation (%) ↓		
				Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
OPUS 5 (continued)										
16	ConcordMark	1	100 / 100 / 100	68 / 83 / 83	89 / 95 / 95	99 / 100 / 100	79 / 88 / 88	5.3	0.6	0.4
		2	100 / 100 / 100	65 / 79 / 79	89 / 94 / 94	100 / 100 / 100	78 / 88 / 88	0.6	1.2	3.0
		3	100 / 100 / 100	70 / 84 / 84	92 / 96 / 96	100 / 100 / 100	77 / 87 / 87	4.4	0.8	4.3
		4	100 / 100 / 100	68 / 82 / 82	91 / 96 / 96	100 / 100 / 100	81 / 88 / 88	5.5	0.7	3.4
		5	100 / 100 / 100	69 / 84 / 84	91 / 95 / 95	100 / 100 / 100	75 / 84 / 84	4.3	1.0	3.6
GEMINI-3.8 FLASH (high)										
1	OmniSeal	1	100 / 100 / 100	19 / 44 / 44	38 / 65 / 65	93 / 97 / 97	45 / 62 / 62	3.1	2.9	11
		2	100 / 100 / 100	20 / 43 / 43	34 / 58 / 58	94 / 98 / 98	46 / 67 / 67	1.0	3.8	12
		3	100 / 100 / 100	20 / 42 / 42	38 / 63 / 63	95 / 99 / 99	46 / 66 / 66	2.4	4.1	12
		4	100 / 100 / 100	18 / 42 / 42	38 / 64 / 64	91 / 98 / 98	47 / 65 / 65	5.0	4.0	13
		5	100 / 100 / 100	18 / 41 / 41	34 / 60 / 60	93 / 98 / 98	47 / 67 / 67	6.8	5.2	16
2	OmniSeal	1	100 / 100 / 100	20 / 44 / 44	37 / 63 / 63	93 / 98 / 98	44 / 64 / 64	2.3	3.0	11
		2	100 / 100 / 100	18 / 42 / 42	34 / 60 / 60	93 / 98 / 98	45 / 66 / 66	0.4	3.8	13
		3	100 / 100 / 100	20 / 44 / 44	38 / 63 / 63	95 / 99 / 99	45 / 68 / 68	2.4	4.2	13
		4	100 / 100 / 100	18 / 43 / 43	38 / 62 / 62	93 / 98 / 98	49 / 68 / 68	4.6	4.2	13
		5	100 / 100 / 100	16 / 40 / 40	34 / 57 / 57	94 / 98 / 98	44 / 64 / 64	7.4	5.0	15
3	OmniSeal	1	100 / 100 / 100	18 / 41 / 41	37 / 64 / 64	93 / 98 / 98	46 / 66 / 66	2.3	3.0	10
		2	100 / 100 / 100	19 / 42 / 42	33 / 58 / 58	92 / 98 / 98	45 / 66 / 66	0.3	3.6	12
		3	100 / 100 / 100	20 / 42 / 42	36 / 62 / 62	94 / 98 / 98	44 / 65 / 65	1.6	4.0	12
		4	100 / 100 / 100	17 / 43 / 43	39 / 63 / 63	93 / 98 / 98	49 / 69 / 69	2.5	3.7	12
		5	100 / 100 / 100	16 / 40 / 40	32 / 59 / 59	93 / 98 / 98	44 / 63 / 63	5.6	4.9	15
4	OmniSeal	1	100 / 100 / 100	19 / 42 / 42	35 / 62 / 62	93 / 99 / 99	46 / 65 / 65	3.1	3.0	11
		2	100 / 100 / 100	19 / 42 / 42	36 / 62 / 62	94 / 98 / 98	48 / 66 / 66	1.3	3.9	13
		3	100 / 100 / 100	19 / 42 / 42	36 / 61 / 61	94 / 98 / 98	42 / 64 / 64	2.9	4.0	12
		4	100 / 100 / 100	20 / 42 / 42	36 / 65 / 65	92 / 98 / 98	45 / 65 / 65	5.0	4.2	13
		5	100 / 100 / 100	16 / 40 / 40	35 / 62 / 62	92 / 97 / 97	43 / 65 / 65	6.9	5.3	16
5	OmniSeal-G0	1	100 / 100 / 100	20 / 41 / 41	36 / 61 / 61	92 / 98 / 98	44 / 63 / 63	3.1	2.9	10
		2	100 / 100 / 100	14 / 38 / 38	32 / 58 / 58	91 / 98 / 98	44 / 64 / 64	1.6	3.6	12
		3	100 / 100 / 100	16 / 38 / 38	36 / 59 / 59	93 / 98 / 98	41 / 64 / 64	2.8	4.0	12
		4	100 / 100 / 100	15 / 38 / 38	35 / 63 / 63	90 / 96 / 96	45 / 66 / 66	5.0	4.3	13
		5	100 / 100 / 100	14 / 37 / 37	34 / 58 / 58	92 / 97 / 97	43 / 64 / 64	7.6	4.9	15
6	OmniSeal-K40	1	100 / 100 / 100	16 / 38 / 38	34 / 61 / 61	93 / 97 / 97	45 / 65 / 65	8.1	4.5	14
		2	100 / 100 / 100	17 / 39 / 39	32 / 58 / 58	91 / 98 / 98	39 / 59 / 59	7.1	4.6	14
		3	100 / 100 / 100	18 / 40 / 40	33 / 58 / 58	93 / 98 / 98	42 / 63 / 63	6.5	5.0	14
		4	100 / 100 / 100	16 / 39 / 39	34 / 61 / 61	93 / 98 / 98	44 / 63 / 63	3.4	4.2	13
		5	100 / 100 / 100	15 / 37 / 37	31 / 58 / 58	92 / 98 / 98	38 / 59 / 59	11	5.7	17
7	OmniSeal-K45	1	100 / 100 / 100	18 / 41 / 41	36 / 62 / 62	93 / 98 / 98	43 / 62 / 62	4.6	3.9	12
		2	100 / 100 / 100	17 / 41 / 41	35 / 58 / 58	94 / 98 / 98	42 / 64 / 64	2.9	3.9	13
		3	100 / 100 / 100	16 / 40 / 40	35 / 60 / 60	93 / 98 / 98	47 / 66 / 66	4.1	4.7	14
		4	100 / 100 / 100	17 / 41 / 41	35 / 64 / 64	93 / 98 / 98	49 / 68 / 68	1.0	3.9	13
		5	100 / 100 / 100	18 / 41 / 41	34 / 59 / 59	92 / 98 / 98	42 / 65 / 65	9.1	5.3	16
8	OmniSeal-K35	1	100 / 100 / 100	15 / 37 / 37	32 / 56 / 56	93 / 98 / 98	40 / 63 / 63	12	4.8	14
		2	100 / 100 / 100	14 / 39 / 39	29 / 56 / 56	92 / 98 / 98	38 / 60 / 60	9.3	5.9	17
		3	100 / 100 / 100	15 / 38 / 38	29 / 58 / 58	92 / 98 / 98	40 / 61 / 61	10	5.7	15
		4	100 / 100 / 100	13 / 36 / 36	33 / 59 / 59	93 / 98 / 98	46 / 67 / 67	7.4	5.0	15
		5	100 / 100 / 100	14 / 37 / 37	30 / 57 / 57	92 / 98 / 98	38 / 60 / 60	14	6.0	17
9	OmniSeal-W05	1	100 / 100 / 100	20 / 41 / 41	34 / 61 / 61	92 / 98 / 98	44 / 63 / 63	2.6	3.2	11
		2	100 / 100 / 100	20 / 40 / 40	34 / 60 / 60	92 / 98 / 98	44 / 65 / 65	0.3	3.5	12
		3	100 / 100 / 100	18 / 39 / 39	34 / 61 / 61	95 / 98 / 98	44 / 63 / 63	1.6	4.0	12
		4	100 / 100 / 100	18 / 40 / 40	39 / 63 / 63	92 / 98 / 98	47 / 67 / 67	3.3	3.4	12
		5	100 / 100 / 100	17 / 39 / 39	31 / 57 / 57	93 / 98 / 98	41 / 61 / 61	5.1	5.1	16
10	OmniSeal-C4	1	100 / 100 / 100	7 / 19 / 19	14 / 34 / 34	87 / 97 / 97	27 / 45 / 45	2.6	1.7	6.1
		2	100 / 100 / 100	7 / 20 / 20	15 / 36 / 36	86 / 96 / 96	27 / 48 / 48	1.7	2.2	7.9
		3	100 / 100 / 100	7 / 21 / 21	17 / 40 / 40	88 / 96 / 96	28 / 47 / 47	1.3	2.4	6.9
		4	100 / 100 / 100	5 / 17 / 17	15 / 38 / 38	86 / 95 / 95	34 / 53 / 53	3.6	2.8	8.9
		5	100 / 100 / 100	5 / 16 / 16	15 / 37 / 37	85 / 94 / 94	25 / 45 / 45	6.1	3.7	11
11	OmniSeal-K30	1	100 / 100 / 100	13 / 34 / 34	28 / 54 / 54	92 / 98 / 98	38 / 60 / 60	16	5.2	15
		2	100 / 100 / 100	13 / 33 / 33	27 / 55 / 55	92 / 98 / 98	42 / 63 / 63	14	6.2	17
		3	100 / 100 / 100	14 / 35 / 35	26 / 53 / 53	92 / 98 / 98	43 / 64 / 64	16	6.3	17
		4	100 / 100 / 100	12 / 35 / 35	30 / 56 / 56	92 / 98 / 98	47 / 68 / 68	11	5.7	16
		5	100 / 100 / 100	13 / 32 / 32	28 / 54 / 54	93 / 98 / 98	41 / 62 / 62	18	7.0	19

Table 16: Per-key results of the prior semantic watermarks of Table 5. Each row reports the results of one of the 5 private watermark keys, without p-value correction. Table 5 reports, for each column, the worst TPR and the median quality deviation over these 5 keys.

			Robustness TPR@1 ↑				Quality deviation (%) ↓		
Scheme	Key	Clean TPR@1 ↑	Deletion	Substitution	Back-trans.	Paraphrase	ΔPPL	ΔSB-2	ΔSB-3
PASA	1	33	1	0	0	0	13	0.9	3.3
	2	64	2	2	9	2	3.7	0.9	1.7
	3	77	3	14	27	13	2.8	1.2	4.0
	4	89	4	20	53	30	6.4	1.3	0.8
	5	94	8	38	63	30	18	1.8	2.9
WatME ($\delta = 3$)	1	100	99	97	100	91	104	6.8	12
	2	100	99	98	100	90	120	6.8	10
	3	100	99	99	100	90	119	5.8	11
	4	100	99	97	100	89	129	8.0	15
	5	100	98	97	100	88	120	6.3	11
WatME ($\delta = 2$)	1	100	86	79	98	75	43	4.8	8.8
	2	100	86	78	98	74	46	4.1	5.5
	3	100	86	89	98	71	51	3.4	6.6
	4	100	78	75	96	69	53	5.7	11
	5	100	82	80	97	72	54	2.9	5.5
SemaMark	1	99	85	52	90	66	47	5.3	11
	2	96	67	38	76	41	43	5.6	8.2
	3	99	76	31	78	44	35	2.3	4.2
	4	99	79	32	84	60	40	3.9	7.9
	5	97	63	31	68	32	54	5.5	11