COULD LLM WATERMARK DETECTION BE PUBLIC?

Anonymous authors
Paper under double-blind review

ABSTRACT

Watermarking large language models is popular for tracing their outputs, yet detectors remain unreleased since exposing them could allow attackers to steer their edits with the detector's feedback. However, watermarks are already vulnerable to uninformed tampering attacks. We thus first quantify whether a public detector would be an additional liability in a deployment setting at varying levels of access, from token-level scores to a binary verdict. Second, we introduce a split-key public-private watermarking method that exposes one key through a public detector while keeping the other for full verification and forensics. An informed attacker can only move the public signal, which creates an imbalance between public and private scores. We introduce a statistical test for this imbalance, and combine it with the full key verdict in a two-stage mechanism. Third, we evaluate the split-key method on a wide range of removal and forgery attacks, comparing the uninformed to detector-informed settings. Public detection improves removal only at small edit budgets, since plain rephrasing already strips the watermark at a lower quality cost, but it does enable forgery, which the private pipeline can identify. Overall, releasing half of the watermark enables transparency and interoperability, and tampering with the released half stays detectable. This bounds the provider's liability and questions the need to keep detectors fully private.

1 INTRODUCTION

Regulations increasingly require AI-generated content to be watermarked (European Parliament and Council of the European Union, 2024; European Commission, 2026; California State Legislature, 2024; Ministry of Electronics and Information Technology, Government of India, 2025), yet detection remains closed and fragmented, as most providers keep their detectors secret. Verifying a text in the wild would mean querying every provider in turn (assuming providers share access at all), raising both effort and the chance of false positives. Cross-industry efforts like C2PA are underway but slow, and sharing even part of a detector would already make verification interoperable.

However, sharing a detector is considered risky because it eases two attacks: *removal* (scrubbing), which strips the watermark to hide LLM use and evade filters in schools or against spam and misinformation, and *forgery* (spoofing), which adds watermarking signal to falsely blame a provider, for instance by inserting a name into a defamatory sentence or fabricating compliance evidence. Importantly, both attacks are already possible without a detector, at a certain distortion. Removal

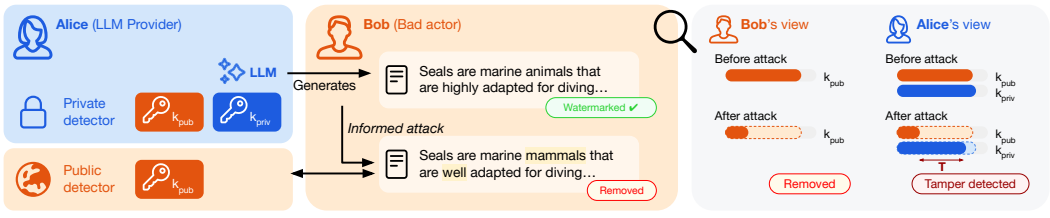


Figure 1: **Overview of our approach.** Alice, an LLM provider, splits the watermark signal across two keys, releasing a detector that uses only the public half k_{pub} while keeping k_{priv} private. Bob may try to use the public detector for informed attacks that remove (or forge) the watermark more effectively than uninformed attacks. Alice, who sees both keys, flags it as tampered from the imbalance $\mathbf{T}$ between the two scores. Therefore, Bob has a better chance of evasion by using uninformed attacks, an already existing risk for Alice.

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.

either rewrites the text globally or edits it locally, and often degrades quality, while forgery either only "piggybacks" on already-watermarked text (Pang et al., 2024) or reverse-engineers the watermark from a large corpus of watermarked outputs (Jovanović et al., 2024). A public detector leaks information about the watermark and lets an attacker steer edits at lower distortion. Watermarking research takes this for granted, and it is the standard reason detectors stay private, yet how much an attacker actually gains has not been measured.

Cryptographic approaches instead embed an unforgeable, publicly-verifiable signature. The methods of Fairoze et al. (2025a;b) are either weakly robust, since detection needs a contiguous span of the text to survive, or intractable to construct for low-entropy text. Since no cryptographic construction is yet robust, fast, and publicly verifiable at once, two questions remain open: (a) How much does public detection actually help an attacker at each level of access, and is the added liability large enough to justify keeping detectors secret? (b) Since cryptographic public detection is impractical, is there a feasible middle ground for statistical LLM watermarks?

To answer (a), we compare an *uninformed* attacker with an *informed* one that queries the detector. We build informed removal and forgery attacks: we consider both rephrasing, via autoregressive regeneration, and in-place word edits steered by the detector. We evaluate them at each level of access, from a fully transparent detector to the binary verdict a public API would return. To answer (b), we present an approach with no unforgeability claim but such that the strongest informed attacks remain *detectable*: we split the watermark across independent keys and release only one for public detection, keeping the full set private. This follows an old idea from asymmetric video watermarking, where publishing only part of the key keeps the watermark publicly detectable yet robust (Hartung & Girod, 1997), but its security is secrecy, not cryptography. This key split exists in TextSeal (Sander et al., 2026), which already routes each token to one of two keys, and extends to other schemes, like Maryland (Kirchenbauer et al., 2023) or SynthID-Text (Dathathri et al., 2024).

Our main contributions are: *(i)* we are the first to measure what a public detector actually costs the provider that releases it, with attackers that query it at every level of access: scores or p-values enable strong informed attacks, while a binary verdict can help removal by indicating when to stop. *(ii)* We introduce a split-key construction with a two-stage detector, and a novel, calibrated tampering test that identifies the public-private imbalance an informed attacker creates. *(iii)* We show the split limits the threat caused by informed attacks, as our strongest informed forgery succeeds on almost all carrier texts yet the majority is caught, while informed removal helps only at small edit budgets. In short, we establish that public detection carries a real but bounded liability, and enable future research on watermark interoperability and transparency.

2 Background and Threat Model

2.1 LLM Watermarking

LLM watermarking alters the tokens' sampling in a key-dependent way to leave a statistical trace only the key holder can test for. The *key* is a fixed secret shared by sampler and detector: a pseudo-random function PRF maps it along with the h preceding tokens to a score $r_t(v)$ per candidate token v, which the sampler combines with the LLM's next-token distribution p_t to pick the next token x_t.

Schemes differ in how r_t alters sampling, and divide into *distortionary* and *non-distortionary* families depending on whether that alteration changes the model's output distribution. Distortionary methods bias sampling, e.g. Maryland randomly partitions the vocabulary into green and red lists from the previous token and the secret key and adds δ to green logits (Kirchenbauer et al., 2023). Non-distortionary methods preserve the original model distribution in expectation over all possible keys. The exact definitions are in App. C.1. In the rest, we mainly consider non-distortionary methods because they are the ones deployed in production systems (Dathathri et al., 2024; Sander et al., 2026; Anthropic, 2026)

Gumbel-max is a common single-token non-distortionary method (Aaronson, 2022). It is based on the Gumbel-softmax trick, and uses the pseudo-random $r_t(v) = \mathrm{PRF}(v; k, x_{t-h:t-1}) \sim U(0,1)$ to choose the next token as $x_t = \arg\max_v r_t(v)^{1/p_t(v)}$. This samples according to p_t on expectation, but with key-dependent randomness. The detector recomputes $r_t(x_t)$ and tests the statistic

$\mathbf{S} = \sum_t -\log(1 - r_t(x_t))$, which follows $\Gamma(n, 1)$ under the null hypothesis H_0 that the text is unwatermarked, and takes larger values under the alternative H_1. While Gumbel-max samples deterministically, TextSeal restores diversity by instantiating it with 2 independent keys k_1, k_2 (Sander et al., 2026), which directly provides our public/private split. Each token is sampled using one of the two keys selected at random with probabilities α and $1 - \alpha$. Since routing is unknown at detection time, scores under both keys are merged into a *fused score* $\mathbf{S} = \alpha\mathbf{S}_1 + (1 - \alpha)\mathbf{S}_2$. Its null is a weighted sum of exponentials and is calibrated via a moment-matched Gamma to get a p-value p_{fus}.

2.2 Threat Model

We assume an LLM provider that exposes a public watermark detector. The attacker's goal is either to *remove* the watermark so that AI-generated text evades detection, or to *forge* it, i.e. embed it into text the platform's model did not generate, as depicted in Fig. 2. We also define: an *uninformed* attacker, who has no detector access and tampers blindly, as commonly assumed in the watermarking literature, and an *informed* attacker, who directly exploits the public detector to steer the attack.

We consider four scenarios of increasing detector transparency: the uninformed *no-box* (uninformed), where no public detector exists, as in current watermark deployments. *Black-box*, where the detector returns a binary watermark decision. *Gray-box*, where it returns the text's p-value, and *white-box*, a fully transparent detector that returns token-level scores. The last three allow for *informed* attacks of increasing strength. *White-box* is the most permissive setting so it upper-bounds the liability of publishing a detector, and we show that it is more advantageous for forgery. *Black-box* answers "watermarked or not", only serving as a stopping rule for attacks (Sec. 4.2). The gray-box tier behaves like the white-box one, since a token's score can be approximated from the p-value difference caused by its presence in the text (App. D.7).

2.3 Attacks on a Public Detector

Tampering with LLM watermarks can be done by two high-level operations. *Rephrasing* rewrites the whole text through global LLM paraphrasing or back-translation, whereas *word edits* apply local lexical substitutions, typically via BERT-based attacks (Li et al., 2020; Garg & Ramakrishnan, 2020). Both are traditionally studied in the no-box setting: prior work scores candidates with proxies of the watermark signal (self-information (Cheng et al., 2025), green-list frequency (Chen et al., 2025), or learned corpora (Jovanović et al., 2024)), because exact token scores were private. A public detector provides exact feedback that lets an informed attacker alter the watermark score directly. We quantify the vulnerability this access adds by extending previous attacks to the informed setting, in the white-box and the black-box tiers [1].

Removal.

- **Rephrasing**: In the no-box setting, an LLM paraphrases the text. In the black-box setting, the attacker tries paraphrases at increasing sampling temperatures and keeps the first one the detector no longer flags. The tuned parameter is then the maximum temperature it will try, and the query cost is the number of paraphrases submitted. In the white-box setting, the detector allows for regenerating the text with sampling explicitly steered away from the watermark. We use

[1] We also considered the tool github.com/guillaumemeyer/watermarks-remover which removes watermarks by character-level normalization then rephrasing with proprietary LLMs. The former is unnecessary, while the latter is equivalent to our rephrasing attack run through a closed API.

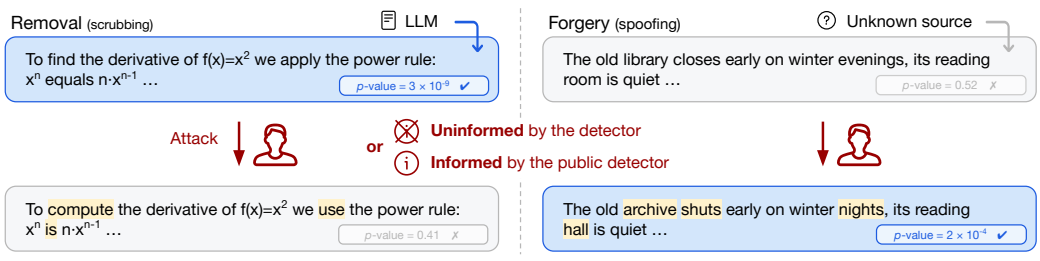


Figure 2: Removal and forgery attacks, under levels of detector access ranging from none to token-level scores.

two mechanisms: (i) *Gumbel-max* (Gumbel) applies the Gumbel trick with the opposite r, thus $x_t = \arg\max_v (1 - r_v)^{1/p_v}$, so tokens with low watermark score are favored (ii) *Additive* (Add.) modifies logits before multinomial sampling, $\ell_v \leftarrow \ell_v - \lambda \log r_v$ with strength $\lambda > 0$, similar to the tunable score-based logit bias of Jovanović et al. (2024); Chen et al. (2025). In both cases, the sampling temperature is used to control rephrasing quality.

- **Word edits**: In the no-box setting, a ratio of tokens is randomly selected, expanded to a full word (avoiding subword artifacts), and masked. A masked language model (MLM) then creates a list of plausible replacement candidates for it, and the most likely replacement is chosen. In the black-box setting, replacements are chosen one word at a time and the detector is queried after each edit, and the attack stops at the first edit the detector no longer flags, or terminates after reaching the edit ratio. In the white-box setting, at each step the highest-scoring token is identified and replaced by the lowest-scoring of the MLM candidates. Edits repeat until the ratio of substitutions is reached, matching the quality degradation of the no-box setting. Prompts, hyperparameters and the full procedure (Alg. 1) are in App. C.2.

Forgery. Existing approaches learn a reverse-engineered watermark from a corpus and forge it onto carrier text (Jovanović et al., 2024; Chen et al., 2025), which is treated in App. D.8. Alternatively, the "piggyback" attack maliciously substitutes tokens to change a watermarked text's meaning (Pang et al., 2024), which is identical to the removal workflow. However, the public detector enables a new type of forgery attack, since its feedback can also be used to raise the score. We thus adapt the removal attacks using inverted objectives: for rephrasing, we replace $1 - r_v$ by r_v (Gumbel) and add $+\lambda \log r_v$ to logits (Add.), while for word edits, we target low-r tokens to pick the highest-scoring replacement. Black-box forgery via an oracle can be done by iteratively sampling text until a false positive is created (Giboulot & Furon, 2024).

3 Split-Key as a Defense Against Informed Attacks

Motivation behind the key split. A detector that fully exposes the watermark is a liability, as we will show with informed attacks (Sec. 4.2). This motivates a split construction with two independent keys, one kept private and one made public, each still detecting the watermark on its own.

We score against each key separately: $\mathbf{S}_{\text{pub}}$ and $\mathbf{S}_{\text{priv}}$ are the accumulated per-token scores under k_{pub} and k_{priv}, each tested against its own null to give two p-values p_{pub} and p_{priv}. Since routing sends only a fraction α of the tokens to the public key, each channel carries a fraction of the signal. The public detector returns p_{pub} to any user, while the platform keeps the detector based on the fused score, which tests $\alpha\mathbf{S}_{\text{pub}} + (1-\alpha)\mathbf{S}_{\text{priv}}$ against the full null for its forensic verdict p_{fus}.

This split benefits the provider on two counts. First, the private signal is never exposed and is thus much harder to tamper with: an attacker who exploits the public detector steers p_{pub}, the only quantity they observe, which does not determine the platform's verdict on p_{fus}. Second, the resulting gap between the public and private scores is itself informative, which we exploit in the next paragraph to detect targeted tampering, as shown in Fig. 1. The two keys need not share the same h-gram context, and giving them different ones adds a second asymmetry: a longer context is seeded by a wider window, so editing destroys its signal faster, and the gap between the two channels is more sensitive to edits. Feng et al. (2026) proposed such a construction concurrently. We keep equal contexts throughout the main experiments, and study the unequal variant in App. E.

Tampering test. We fix α=0.5 in TextSeal (Sec. 2.1), which makes both channels exchangeable on untampered text (watermarked or not), making per-token differences $d_t = s_{\text{pub},t} - s_{\text{priv},t}$ symmetric around zero. An attacker exploiting the public detector disrupts this symmetry, which we measure via $\mathbf{T} = \mathbf{S}_{\text{pub}} - \mathbf{S}_{\text{priv}} = \sum_{t=1}^{n} d_t$: removal decreases public signal ($\mathbf{T} < 0$), whereas forgery increases it ($\mathbf{T} > 0$). Both tails test the same null H_0, that the keys are exchangeable, against $\mathbf{T} < 0$ for removal and $\mathbf{T} > 0$ for forgery. Because $\mathbf{T}$'s distribution lacks a closed form under watermarking, we calibrate this null by an exact sign-flip permutation test (Ernst, 2004; Lehmann & Romano, 2022), drawing R random sign vectors $\boldsymbol{\varepsilon}^{(r)} \in \{-1, +1\}^n$ to compute $\mathbf{T}^{(r)} = \sum_{t=1}^{n} \varepsilon_t^{(r)} d_t$. The resulting p-values (with granularity controlled by R, see App. C.3) are:

$$p_{\text{removal}} = \frac{1 + \sum_{r=1}^{R} \mathbb{1}[\mathbf{T}^{(r)} \leq \mathbf{T}]}{1+R}, \quad p_{\text{forgery}} = \frac{1 + \sum_{r=1}^{R} \mathbb{1}[\mathbf{T}^{(r)} \geq \mathbf{T}]}{1+R}. \tag{1}$$

Two-stage mechanism. We construct a two-stage testing mechanism, consisting of the watermark test on the fused score, followed by the tampering test, similar to the image watermarking construction of Evennou & Kijak (2026). The first stage compares p_{fus} to the threshold set by the target False Positive Rate (FPR), which decides the binary watermark label y_{fus}. The second stage re-evaluates that watermark verdict with the tampering test. A text not detected as watermarked is tested for removal, while one detected as watermarked is tested for forgery. The two stages test different nulls: p_{fus} that the text carries no watermark, and p_{removal} or p_{forgery} that its two channels are balanced. Selecting the tail with y_{fus} leaves both tails calibrated, since relabeling the channels alters their difference but not their sum which determines y_{fus} (App. C.3). Unwatermarked text can be failed by either test, so its FPR is 2×10^{-3}, while a genuine watermarked text faces only the forgery tail, and ADR and ASR_{net} are conditional on the first-stage verdict, each at 10^{-3}. An attack informed by the public detector is subject to a tradeoff it cannot observe: tampering with p_{pub} enough to modify p_{fus} also enlarges $|\mathbf{T}|$, and is thus caught by the second stage. The process is depicted in Fig. 3.

4 Experiments

After setup (Sec. 4.1), we measure transparent detector risks (Sec. 4.2), evaluate the split-key defense against informed attacks (Sec. 4.3), test an attacker adapting to the tampering test (Sec. 4.4), and look into other watermark types (Sec. 4.5).

4.1 Experimental setup

Text generation and watermark parameters. We generate texts with Qwen2.5-7B (Qwen Team, 2024) unless stated otherwise (we generalize in App. D.2), watermarked with TextSeal using 2 keys, α=0.5 and symmetric h=3-gram contexts in each channel. The core watermarking mechanism is Gumbel-max, with Maryland and SynthID-Text deferred to Sec. 4.5. We use 1000 samples from ELI5 (Fan et al., 2019), a dataset of open-ended questions paired with human-written answers (we extend to a multilingual domain in App. D.9). Each dataset entry gives (a) a prompt for the model, used to obtain watermarked text for removal attacks, and (b) the human answer as carrier text for forgery attacks. We set the maximum generation length to 400 tokens (effectively achieving an average of 360 tokens), and default top_p=0.95, top_k=50 and temperature of 1.0. For forgery carriers, we select the 1k longest ELI5 responses, which have 220 tokens on average. We set the detection threshold for both the watermark test and the tampering test at FPR of 10^{-3}, and report the watermark strength before any attack in App. D.3.

Attack parameters. We report the white-box and black-box settings of Sec. 2.2, and defer gray-box to App. D.7. The black-box attacker queries the deployed public detector looking to change its decision: word edit attacks, all using RoBERTa MLM (Liu et al., 2019), sweep up to the same edit fractions as the other tiers, and rephrasing attacks try temperatures in steps of 0.2 up to a maximum. The rephrasing proxy is Qwen2.5-3B at strength λ=1 unless stated otherwise (*cross*-tokenizer rephrasing in App. D.4).

Metrics. An attack succeeds when both the watermark detection test and the tampering test return the wrong label. Writing y for the true label, y_c for the label of detector $c \in \{\text{pub}, \text{fus}\}$, and $t = 1$ the tampering flag event, we define from the two stages of Sec. 3, the Attack Success Rate (ASR)

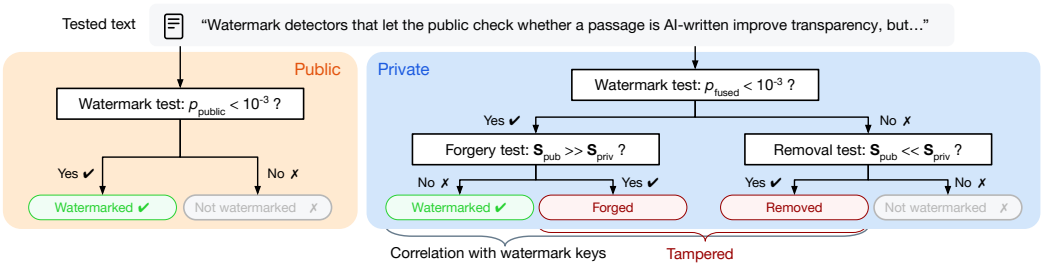


Figure 3: **Overview of the two-stage test.** *(Left)* The public detector only tests based on the public p-value p_{pub}. *(Right)* The private detector first tests p_{fus} to get y_{fus}, then re-evaluates it with the tampering test on $\mathbf{T} = \mathbf{S}_{\text{pub}} - \mathbf{S}_{\text{priv}}$. If $y_{\text{fus}} = 1$ we test for forgery ($s_{\text{pub}} \gg s_{\text{priv}}$), if $y_{\text{fus}} = 0$ for removal ($s_{\text{pub}} \ll s_{\text{priv}}$). Genuine watermarked texts have $\mathbb{E}[\mathbf{T}] = 0$, while tampered ones are flagged as forged or removed.

Table 1: Removal and forgery ASR at three access levels, on baselines with K keys, where every key is published, for Gumbel-max and Maryland (γ=0.25, δ=2) watermarks. Attacks are additive rephrasing and 10% word edits, while distortion is 1−BERT-F1 (↓), pooled over the four watermarks. A white-box detector is the most exploitable tier, and the only setting that directly enables forgery. Using a black-box detector as a stopping rule gives removal a few points of ASR at consistently lower distortion.

	Removal						Forgery					
	Rephrase (Add.)			Word edits (10%)			Rephrase (Add.)			Word edits (10%)		
Detector:	no-box	black-box	white-box	no-box	black-box	white-box	no-box	black-box	white-box	no-box	black-box	white-box
Gumbel-max K=1	70.4%	81.4%	99.9%	1.5%	1.6%	72.2%	0.0%	0.6%	20.8%	0.2%	0.3%	56.3%
Gumbel-max K=2	82.0%	92.9%	99.5%	6.7%	8.4%	97.8%	0.1%	0.3%	6.2%	0.1%	0.1%	48.3%
Maryland K=1	90.1%	96.1%	100.0%	8.6%	10.7%	15.9%	0.4%	1.1%	78.6%	0.1%	0.5%	1.3%
Maryland K=2	95.2%	98.9%	100.0%	34.4%	40.4%	98.7%	0.1%	0.9%	38.8%	0.1%	0.3%	1.9%
Distortion	0.054	0.046	0.063	0.019	0.016	0.024	0.094	0.094	0.096	0.021	0.021	0.024

on the detector c, or net after the joint tampering test, and attack detection rates (ADR) as:

$$\mathrm{ASR}_c = \Pr[y_c \neq y], \quad \mathrm{ADR} = \Pr[t = 1 \mid y_{\mathrm{fus}} \neq y], \quad \mathrm{ASR}_{\mathrm{net}} = \mathrm{ASR}_{\mathrm{fus}}\,(1 - \mathrm{ADR}). \tag{2}$$

$\mathrm{ASR}_{\mathrm{pub}}$ is the success the attacker observes, $\mathrm{ASR}_{\mathrm{fus}}$ the one the defender sees before the second stage, and $\mathrm{ASR}_{\mathrm{net}}$ what survives both stages. Where useful, we also report the watermark signal directly as $-\log_{10} p$, where p is the p-value returned by the watermark detection test, so a larger value means stronger evidence of a watermark. We measure the text *distortion* caused by the attack as 1 − BERT-F1 (Zhang et al., 2020) against the pre-attack text, which is 0 for untouched texts and grows with the amount of editing.

4.2 Vulnerability of Public Detection

We first measure how much a public detector helps an attacker performing the attacks of Sec. 2.3, in the no-box, black-box and white-box settings of Sec. 2.2. Watermarking research assumes that such access trivially enables tampering, which is the reason detectors stay private, yet the size of the advantage has not been systematically quantified. In Tab. 1 we report the results on the Gumbel-max and Maryland baselines with K keys. Gumbel-max with $K = 2$ keys is equivalent to TextSeal, while Maryland with $K = 2$ keys applies the same routing to two independent green/red lists.

First, the white-box attacks (fully transparent detector) reach far higher success rates than their uninformed counterparts. This is especially true for forgery, which otherwise requires a corpus (or a valid watermarked sample for the "piggyback" case). Rephrasing reaches about 100% removal ASR, and forgery reaches up to 79% ASR on Maryland, which rules out releasing a fully transparent detector. In the black-box setting (binary result), forgery remains at chance since the detector can only verify if a text has crossed the public detection threshold. For removal, the black-box setting improves the ASR-distortion tradeoff. The attacker tries candidates of increasing strength and keeps the first that flips the verdict, so it changes the text less than the uninformed attack. This increase costs one query per candidate, which a provider could control with rate limits.

4.3 Evaluating the Split-Key Defense

We now run the attacks of Sec. 2.3 against the split-key defense of Sec. 3, and report the two-stage effectiveness in Fig. 4. For removal (left), informed rephrasing and word edits reach near-100% public ASR, but they widen the public/private gap that the tampering test flags, so fewer attacks are successful after both stages. The informed attacker therefore succeeds more often than the no-box one only at the smallest budgets, where it needs fewer edits for the same effect, and less often at the larger ones, where the imbalance it leaves is rejected by the second stage. Forgery (right) is the opposite case: without access it succeeds only by chance, while public access lets the attacker raise the public scores. The $\mathrm{ASR}_{\mathrm{net}}$ stays above the near-zero no-box baseline even after the two stages, so public knowledge is useful for forgery. It is however bounded, since weak attacks do not move the fused score, while strong ones separate the two channels and the tampering test flags most of them. Full per-attack ASRs and text distortions are in App. D.1 (Tab. 6 and Tab. 7), while specific text examples before and after the attacks are provided in App. C.4.

Fig. 4 fixes each attack at one operating point, so in Fig. 5a we sweep removal over rephrasing temperatures and word edit fractions, plotting $\mathrm{ASR}_{\mathrm{pub}}$ and $\mathrm{ASR}_{\mathrm{net}}$ against median distortion. Informed word edits reach higher public ASR at lower distortion than rephrasing needs for comparable public

success, but these stronger attacks are mostly caught by the tampering test: their $\mathrm{ASR}_{\mathrm{net}}$ peaks and then falls below no-box rephrasing. Fig. 5b shows why, on the informed word edit sweep for removal and forgery: the attacker moves the public score freely (up for forgery, down for removal), the fused score partly follows, and the private score moves slower for removal and stays fixed for forgery. Therefore, the channel discrepancy the tampering test measures grows with attack strength. Another view of this tradeoff is to quantify the distortion required to flip the detector (App. D.6).

The black-box curve in Fig. 5a behaves differently from both. With word edits it falls between the other two tiers on the public detector at matched distortion, but comes last on the provider's verdict. Since it stops the moment the verdict flips, the text ends just past the public boundary, well below the level the fused test needs to reject it. Rephrasing behaves the same way on the provider's verdict, where it stays at the no-box level, while on the public detector it is the strongest tier at matched distortion. At this tier no tampering is yet detected (ADR 0%), because non-targeted edits weaken both channels by the same amount and leave $\mathbf{T}$ unchanged, so $\mathrm{ASR}_{\mathrm{net}} = \mathrm{ASR}_{\mathrm{fus}}$ and the first stage provides the whole defense. Unequal h-gram contexts do not close this gap (App. E).

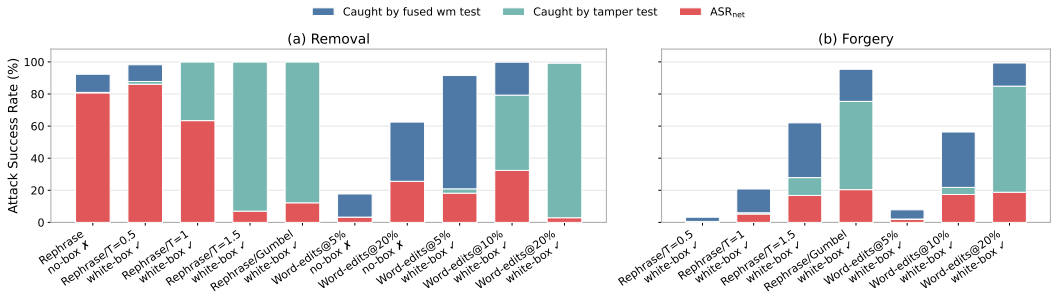
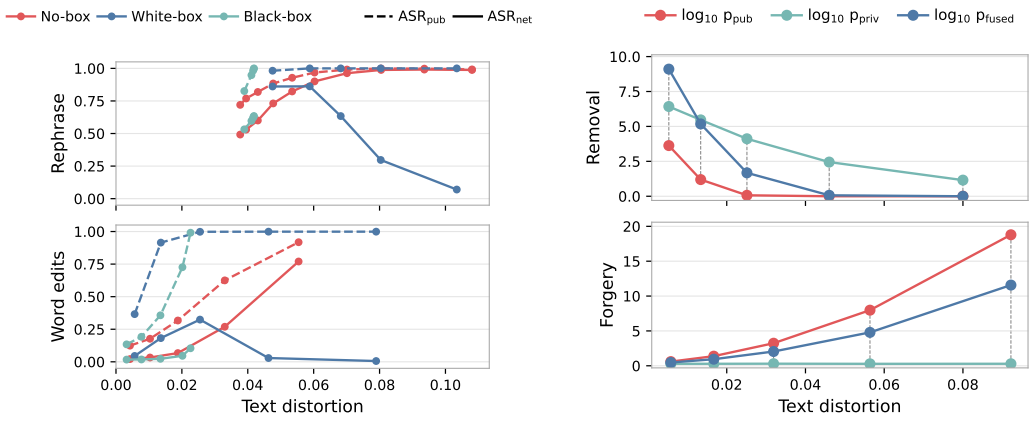


Figure 4: **Attack success rate under the two-stage test**, for removal (left) and forgery (right). Each bar represents the $\mathrm{ASR}_{\mathrm{pub}}$ the attacker sees from the public detector, split into the part the test on the fused watermark score rejects, the part the tampering test then catches, and the $\mathrm{ASR}_{\mathrm{net}}$ that survives both. Informed attacks fool the public detector, but are overall not more effective than uninformed paraphrasing for removal, and are often flagged for forgery.



(a) Attacker success at each tier.

(b) Signal per channel for white-box attacks.

Figure 5: *(Left)* $\mathrm{ASR}_{\mathrm{pub}}$ (dashed) and $\mathrm{ASR}_{\mathrm{net}}$ (solid) against median distortion, for removal by rephrasing (top) and word edits (bottom), at the **no-box**, **white-box** and **black-box** tiers. The swept parameter is temperature $T \in \{0.5, 0.8, 1.0, 1.2, 1.5\}$ or edit fraction $\{2, 5, 10, 20, 40\}\%$ (for **black-box**, the maximum it will spend). The informed attacks with word edits are the most harmful ones for the public detector, but get caught by the two-stage tests (lower $\mathrm{ASR}_{\mathrm{net}}$). Overall, informed attacks are stronger only at low distortion, while **no-box** rephrasing is more effective at higher distortion. *(Right)* Median $-\log_{10} p$ of the public, private and fused scores against median distortion, for the white-box word edit sweep, removal (top) and forgery (bottom). The private score drops slower than the public one for removal and stays fixed for forgery, so the gap the tampering test examines grows with attack strength.

Table 2: **Adaptive attacks** against the no-box baseline and their unaware white-box counterparts, at the same operating points (rephrasing $T{=}1$, word edits 20%). Bold marks the highest $\mathrm{ASR}_{\mathrm{net}}$ per family. Adaptive attacks increase the $\mathrm{ASR}_{\mathrm{net}}$ up to a point, with the private pipeline rejecting a smaller portion of them. However, they become detectable again at the large overshoot factor of 10^3. Despite the limited but real liability of the white-box detector in the adaptive attacks scenario, the $\mathrm{ASR}_{\mathrm{net}}$ with black-box adaptive attacks still matches no-box baselines (App. D.5).

	Removal								Forgery					
	Rephrase				Word edits				Rephrase			Word edits		
Metric	No-box	Adap. ($m{=}10$)	Adap. ($m{=}10^2$)	Adap. ($m{=}10^3$)	No-box	Adap. ($m{=}10$)	Adap. ($m{=}10^2$)	Adap. ($m{=}10^3$)	Adap. ($m{=}10$)	Adap. ($m{=}10^2$)	Adap. ($m{=}10^3$)	Adap. ($m{=}10$)	Adap. ($m{=}10^2$)	Adap. ($m{=}10^3$)
$\mathrm{ASR}_{\mathrm{pub}}\downarrow$	92.3%	99.5%	99.9%	100.0%	62.5%	99.9%	99.8%	99.8%	18.1%	19.6%	20.8%	99.8%	99.9%	99.9%
$\mathrm{ASR}_{\mathrm{fus}}\downarrow$	82.0%	88.5%	95.1%	99.4%	26.9%	8.8%	30.8%	99.3%	4.8%	7.5%	5.5%	45.8%	69.8%	85.1%
ADR$\uparrow$	0.4%	0.2%	0.2%	0.7%	0.0%	0.0%	0.3%	99.4%	0.0%	1.3%	5.5%	10.9%	36.0%	66.2%
$\mathrm{ASR}_{\mathrm{net}}\downarrow$	80.7%	88.1%	94.9%	**98.7%**	25.6%	8.8%	**30.7%**	0.6%	3.2%	**5.9%**	4.0%	40.8%	**44.7%**	28.8%
Distortion$\downarrow$	0.052	0.052	0.053	0.057	0.032	0.012	0.016	0.060	0.096	0.096	0.096	0.040	0.046	0.050

4.4 Adaptive Attacks

So far, we assumed attackers who only target the public detector and ignore the two-stage defense. We now consider a more conservative, "adaptive" attacker. As noted in Sec. 3, stopping at the detection boundary leaves enough private signal for the fused test, while pushing further widens the gap the tampering test identifies. The adaptive attack aims between the two and overshoots the stopping p-value by a factor m, targeting $p_{\mathrm{target}} \cdot m$ for removal and p_{target}/m for forgery. For word edits, the attack stops once the text reaches the target. For additive rephrasing, we scale the bias with the distance to the target and replace the fixed removal bias λ by $\max\left(0, \log\left(p_{\mathrm{target}}/p_{\mathrm{current}}\right)\right)$ (inverted ratio for forgery), where p_{current} is p_{pub} of the text generated so far. For white-box, Tab. 2 compares adaptive to unaware attacks at the same operating points. It shows that adaptive attacks evade the tampering test (low ADR), which raises net success against the unaware version. For rephrasing, removal slightly exceeds the no-box paraphrase at the same distortion, and it stays low for edits, while forgery stays low for rephrasing and rises for edits but remains partial.

The black-box attacks only observe the yes/no verdict, and stop just at the boundary. This usually gets flagged by the more powerful fused test. The adaptive attacker must thus keep editing after crossing the threshold. Since this tier returns no score to aim at, the same overshoot factor m is applied in edits instead of in p-value: if the verdict flips after e edits, the attacker continues to $m \cdot e$ edits, up to the full budget. $\mathrm{ASR}_{\mathrm{net}}$ increases with m and remains below the no-box attack. At matched distortion, a partial overshoot reaches a slightly higher $\mathrm{ASR}_{\mathrm{net}}$ than no-box (App. D.5).

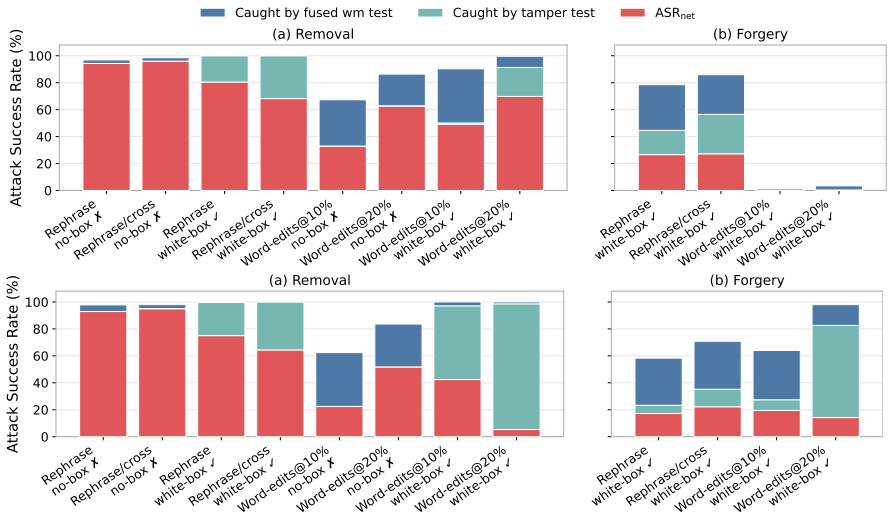


Figure 6: The split-key construction applies to a Maryland (top) and a SynthID watermark (bottom), where the fused watermark test and the tampering test both reject white-box attacks on the public detector. Rephrasing is done at temperature of 1 in each case, and we also include cross-tokenizer rephrasing. White-box rephrasing remains less successful than its no-box counterpart on both schemes, while white-box word edits are more successful at the smaller edit budgets.

Offline attacks that learn the watermark from a corpus are evaluated in App. D.8. They need no detector at all, and forge most carrier texts without moving the two channels apart, but only at h=2: at the h=3 used here they reach at most 3% ASR, so this threat is dependent on the context length.

4.5 Extension to Other Schemes

So far we have focused on TextSeal's Gumbel-max core, but dual-key routing extends to other watermarks. We implement Maryland (Kirchenbauer et al., 2023) and SynthID-Text (Dathathri et al., 2024) variants under the same construction of routing each position to the public key with probability α=0.5, and to the private key otherwise. In the Maryland variant, each key defines its own green list, and generation adds δ to the green tokens of the routed key. Similarly, with SynthID-Text each key defines a per-round partition, and the per-token score is the tournament total across rounds. In both cases, detection is a Binomial tail for both the public and fused detectors, and the tampering test remains the score difference against a channel permutation null.

The Maryland parameters are γ=0.25 and δ=2, and SynthID-Text uses tournament depth 30 at γ=0.5, while both use the same context of h=3 tokens per channel. Both use the same dataset, model, and generation settings as the main TextSeal runs. The effectiveness of the split construction is presented in Fig. 6, following Fig. 4. ADR is high for white-box attacks on both goals, showing that tampering using the public detector does not help the attacker and raises the chance of detection.

5 Related Work

LLM watermarking was introduced by Aaronson (2022), with the non-distortionary Gumbel-max method using pseudorandom next-token sampling, and Kirchenbauer et al. (2023), whose distortionary method boosts a green-list of logits (follow-ups achieved unbiased distributions by scaling this bias (Hu et al., 2024)). Other non-distortionary methods include inverse sampling (Kuditipudi et al., 2024), SynthID-Text (Dathathri et al., 2024), Permute-and-Flip (Zhao et al., 2024), TextSeal (Sander et al., 2026), etc. More details and methods are given in App. B.1.

Public detectability. Christ & Gunn (2024) introduced cryptography-based watermarking, later extended by Fairoze et al. (2025a) via payload-embedded cryptographic signatures. Alternatively, public security can rely on independent embedding and detection networks (Liu et al., 2024), ensuring unforgeability even with publicly shared detectors. Concurrent work by Feng et al. (2026) co-embeds robust and fragile watermarks with independent keys to detect uninformed tampering.

Removal and forgery. Corpus-stealing attacks reverse-engineer the watermark from collected outputs, from Maryland with known PRFs (Jovanović et al., 2024) to any PRF (Chen et al., 2025) and any logit reweighting (Zhang et al., 2026). Some works also fine-tune paraphrasers for these tasks: Huang et al. (2026) fine-tune a rephrasing model towards forgery, while Diaa et al. (2025) use preference optimization against known watermarking methods and greatly improve no-box removal. These attacks focus on the no-box scenario, which supports our claim since our baselines use plain paraphrasing or word-editing, which are weaker yet already close to the informed attacks.

6 Conclusion

This work examines to what extent the release of a public LLM watermark detector is a liability to the LLM provider. Given that no-box attacks are possible in any watermark deployment, we investigate whether a public detector can enable stronger watermark tampering. We build informed attacks that exploit a public detector carrying half of the watermark signal, and manipulate its detection result. However, the private half of the watermark can reveal the deviation in tampered texts, which we quantify in a statistical test for informed tampering. The split construction and the tampering test lower the net success rate of informed attacks, although not as effective against the careful attacker who keeps the channel imbalance small. Broadly, we show that in the case of watermark detection, a split construction can be used as a practical compromise between transparency and security, since it leaves removal close to what the no-detector setting already allows, and keeps the forgery it enables mostly detectable. Watermark detection could therefore be public: even the strongest attacks that a transparent detector enables leave the provider with a liability that is real but bounded by the key split. App. A discusses the scope and limitations of the construction.

AI Use Statement

In this work, we used generative AI to refine hypotheses and implement methods. We have not used AI tools for theoretical models, mathematical tasks, or interpreting results, and the rest of the required disclosure tasks are not applicable. Additionally, we used generative AI for creating software code, drafting parts of the paper, and identifying literature. Specifically, we formulated the research questions and the experimental methods to address them, then used AI tools to refine them and implement the code for experiments. We manually drafted the paper and revised it via generative AI for text compression and cohesion, and manually inserted all citations. We have reviewed all AI-assisted output and take full responsibility for the entire content of this work.

Ethics Statement

This work studies whether LLM watermark detection can be public. Regulations require AI-generated content to be marked and detectable (European Parliament and Council of the European Union, 2024; California State Legislature, 2024), yet most providers keep their detectors private because an exposed detector could help attackers. To make informed decisions on public detection, policymakers and providers need to know how large this risk is, which requires building and evaluating such attacks. We introduce informed removal and forgery attacks that exploit a public detector, which an actor could use to hide LLM use or to falsely attribute a text to a provider. These attacks need query access to a detector, so they do not apply to current deployments that keep it private. We run them only against our own watermark implementations on open-weight models.

We also introduce a way to publish detection results: the provider releases one of two watermark keys through a public detector and keeps the other private. A two-stage private test on both keys catches most informed forgeries that pass the public detector, including adaptive ones, and informed removal stays close to what is already possible without a detector.

We therefore expect the overall impact of this work to be positive.

Reproducibility Statement

Experiments build mainly on https://github.com/facebookresearch/textseal, which has a permissive license (Apache-2.0). We will release the experiment code with a permissive license for the camera-ready version. All models and datasets are publicly available, and no proprietary data is used. Qwen2.5-7B, SmolLM2, SmolLM3, RoBERTa, MiniLM, GPT-2 and CaLMQA have permissive licenses (Apache-2.0 or MIT). Qwen2.5-3B and Gemma 3 have custom licenses that allow research use (Qwen Research License, Gemma Terms of Use), and ELI5 consists of public Reddit posts.

References

Scott Aaronson. Watermarking of large language models. Talk, Simons Institute for the Theory of Computing, 2022.

Anthropic. How Claude marks AI-generated content, 8 2026. URL https://support.claude.com/en/articles/16266773-how-claude-marks-ai-generated-content. Accessed: 2026-08-13.

Toluwani Aremu, Noor Hussein, Munachiso Nwadike, Samuele Poppi, Jie Zhang, Karthik Nandakumar, Neil Gong, and Nils Lukas. Mitigating watermark forgery in generative models via randomized key selection. *arXiv preprint arXiv:2507.07871*, 2025.

Shane Arora, Marzena Karpinska, Hung-Ting Chen, Ipsita Bhattacharjee, Mohit Iyyer, and Eunsol Choi. Calmqa: Exploring culturally specific long-form question answering across 23 languages. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11772–11817, 2025.

Shariq Bashir. Asymmetric watermarking for large language models with public and private verification. *IEEE Access*, 2026.

California State Legislature. SB-942 California AI Transparency Act: Cal. Bus. & Prof. Code § 22757 – disclosure and AI detection tools. Enacted Sept. 19, 2024, effective Jan. 1, 2026, 2024. URL https://leginfo.legislature.ca.gov/faces/billNavClient.xhtml?bill_id=202320240SB942.

Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. De-mark: Watermark removal in large language models. In *International Conference on Machine Learning*, pp. 9316–9333. PMLR, 2025.

Yixin Cheng, Hongcheng Guo, Yangming Li, and Leonid Sigal. Revealing weaknesses in text watermarking through self-information rewrite attacks. In *International Conference on Machine Learning*, pp. 9982–10009. PMLR, 2025.

Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In *Annual International Cryptology Conference*, pp. 325–347. Springer, 2024.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

Abdulrahman Diaa, Toluwani Aremu, and Nils Lukas. Optimizing adaptive attacks against watermarks for language models. In *International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2410.02440.

Michael D. Ernst. Permutation methods: A basis for exact inference. *Statistical Science*, 19(4): 676–685, 2004. doi: 10.1214/088342304000000396.

European Commission. Code of Practice on Transparency of AI-Generated Content, 2026. URL https://digital-strategy.ec.europa.eu/en/faqs/code-practice-transparency-ai-generated-content.

European Parliament and Council of the European Union. Regulation (EU) 2024/1689 laying down harmonised rules on artificial intelligence – EU AI Act, Article 50 on machine-readable marking of AI-generated content, 2024. URL https://eur-lex.europa.eu/eli/reg/2024/1689/oj.

Gautier Evennou and Ewa Kijak. The forensic cost of watermark removal. In *Proceedings of the 2026 ACM Workshop on Information Hiding and Multimedia Security*, pp. 183–188, 2026.

Jaiden Fairoze, Sanjam Garg, Somesh Jha, Saeed Mahloujifar, Mohammad Mahmoody, and Mingyuan Wang. Publicly-detectable watermarking for language models. *IACR Communications in Cryptology*, 1(4), 2025a.

Jaiden Fairoze, Guillermo Ortiz-Jimenez, Mel Vecerik, Somesh Jha, and Sven Gowal. On the difficulty of constructing a robust and publicly-detectable watermark. In Yingzhen Li, Stephan Mandt, Shipra Agrawal, and Emtiyaz Khan (eds.), *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 1891–1899. PMLR, 03–05 May 2025b.

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, 2019.

Xiaoyan Feng, Yanjun Zhang, He Zhang, Leo Yu Zhang, and Shirui Pan. Tracing provenance and detecting tampering with complementary llm watermarks, 2026. URL https://arxiv.org/abs/2608.12713.

Siddhant Garg and Goutham Ramakrishnan. Bae: Bert-based adversarial examples for text classification. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6174–6181, 2020.

Gemma Team. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

Eva Giboulot and Teddy Furon. Watermax: breaking the llm watermark detectability-robustness-quality trade-off. *Advances in neural information processing systems*, 37:18848–18881, 2024.

Thibaud Gloaguen, Nikola Jovanović, Robin Staab, and Martin Vechev. Discovering spoofing attempts on language model watermarks. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 19562–19584. PMLR, 7 2025.

Chenchen Gu, Xiang Li, Percy Liang, and Tatsunori Hashimoto. On the learnability of watermarks for language models. In *International Conference on Learning Representations*, volume 2024, pp. 38745–38768, 2024.

Frank Hartung and Bernd Girod. Fast public-key watermarking of compressed video. In *Proceedings of International Conference on Image Processing*, volume 1, pp. 528–531. IEEE, 1997.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. Unbiased watermark for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45408–45436, 2024.

Hanbo Huang, Xuan Gong, Yiran Zhang, Hao Zheng, and Shiyu Liang. Rlspoofer: A lightweight evaluator for llm watermark spoofing resilience. *arXiv preprint arXiv:2604.11546*, 2026.

Chloé Imadache, Eva Giboulot, and Teddy Furon. Evaluating the security of public surrogate watermark detectors. In *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1–5. IEEE, 2025.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 22570–22593, 2024.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International conference on machine learning*, pp. 17061–17084. PMLR, 2023.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *Transactions on Machine Learning Research*, 2024.

Erich L. Lehmann and Joseph P. Romano. *Testing Statistical Hypotheses*. Springer Texts in Statistics. Springer, 2022. ISBN 978-3-030-70578-7. doi: 10.1007/978-3-030-70578-7.

Hao Li, Yubing Ren, Yanan Cao, Yingjie Li, Fang Fang, Shi Wang, and Li Guo. Dualguard: Dual-stream large language model watermarking defense against paraphrase and spoofing attack. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 23338–23361, 2026.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. Bert-attack: Adversarial attack against bert using bert. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6193–6202, 2020.

Aiwei Liu, Leyi Pan, Xuming Hu, Shuang Li, Lijie Wen, Irwin King, and Philip Yu. An unforgeable publicly verifiable watermark for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 8291–8307, 2024.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Nils Lukas, Abdulrahman Diaa, Lucas Fenaux, and Florian Kerschbaum. Leveraging optimization for adaptive attacks on image watermarks. In *International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2309.16952.

Ministry of Electronics and Information Technology, Government of India. MeitY Advisory on AI-Generated Content Labelling and AI Governance Guidelines 2025 – mandatory watermarking and labelling proposals for AI-generated content and deepfakes. Roadmap for AI governance, IT Rules advisory 2025, 2025. URL https://www.meity.gov.in/.

Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. No free lunch in llm watermarking: Trade-offs in watermarking design choices. *Advances in Neural Information Processing Systems*, 37:138756–138788, 2024.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

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 and distillation protection. arXiv preprint arXiv:2605.12456, 2026.

Huanming Shen, Baizhou Huang, and Xiaojun Wan. Enhancing llm watermark resilience against both scrubbing and spoofing attacks. *Advances in Neural Information Processing Systems*, 38: 27957–27993, 2026.

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

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

Dor Tsur, Carol Long, Claudio Mayrink Verdun, Sajani Vithana, Hsiang Hsu, Richard Chen, Flavio Calmon, et al. Heavywater and simplexwater: Distortion-free llm watermarks for low-entropy distributions. *Advances in Neural Information Processing Systems*, 38:107204–107256, 2026.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. A resilient and accessible distribution-preserving watermark for large language models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53443–53470. PMLR, 21–27 Jul 2024.

Yihan Wu, Ruibo Chen, Georgios Milis, and Heng Huang. An ensemble framework for unbiased language model watermarking. In *International Conference on Learning Representations*, volume 2026, pp. 17899–17917, 2026.

Shuhao Zhang, Yuli Chen, Jiale Han, Bo Cheng, and Jiabao Ma. Beyond a fixed seal: Adaptive stealing watermark in large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 20678–20695, 2026.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.

Xuandong Zhao, Lei Li, and Yu-Xiang Wang. Permute-and-flip: An optimally stable and watermarkable decoder for llms. *arXiv preprint arXiv:2402.05864*, 2024. URL https://arxiv.org/abs/2402.05864.

Huadi Zheng, Qingqing Ye, Haibo Hu, Chengfang Fang, and Jie Shi. Bdpl: A boundary differentially private layer against machine learning model extraction attacks. In *European Symposium on Research in Computer Security*, pp. 66–83. Springer, 2019.

A Discussion and Limitations

On the channel split. A white-box public detector could in principle leak the channel routing: on watermarked text, tokens routed to the private key score low on the public detector, so an attacker could tell the two channels apart. For removal, using this routing means also editing the low-scoring tokens, but those edits get no detector guidance and amount to uninformed editing, which our no-box baselines already cover. For forgery there is nothing to leak, since non-watermarked carrier text scores are uniform on each key. In any case, a realistic deployment would be black-box and return no per-token scores; it can be further hardened with rate limits or a randomized band around the threshold that answers by coin flip, as done against model extraction (Zheng et al., 2019). We leave these defenses to future work and evaluate the fully transparent case as a liability upper bound.

On the value of a public detector. One objection is that a public detector is pointless, since no-box attacks are already effective. However, there are multiple reasons in favor of the release of a public detector. First, verification is no longer proprietary: anyone can examine a text for free without a prior relationship with the provider, which serves the interoperability that Article 50 requires of marking solutions, as far as is technically feasible (European Parliament and Council of the European Union, 2024). Second, large portions of LLM-generated text are produced by agentic tools and are never tampered, so the public detector is still an accessible tool helping against mass misinformation and fraud.

Regarding exploitation risk, anyone can repeatedly query the detector and edit their text against the returned result, which is the only signal most entities have access to. However, we have shown that such exploitation is itself detectable: steering with the detector's feedback moves the public channel away from the private one, which is identified in the full detection pipeline. The latter can still be licensed to trusted partners such as social media platforms, regulatory or investigative bodies, and plagiarism screening tools. This should deter the informed attack, as it leaves the attacker with rephrasing only, which is an already existing risk, and raises questions on the responsibility of the provider for marking a text that is no longer its original version[2].

On technical literacy and communication. Non-technical LLM users may misinterpret watermarking and how detection operates. Even if the provider's public detection tool explicitly states that exploiting it could leave a trace, this will not guarantee rational attack behavior. However, an intuitive public detection platform, with appropriate information for the user, would still be important towards achieving transparency.

B Extended Related Work

B.1 LLM watermarking

Early work such as Topkara et al. (2005; 2006) performed character-level, syntactic, or synonym substitutions for post-hoc text watermarking. Such methods have low capacity and robustness, now superseded by generation-time techniques that are compatible with autoregressive LLMs.

A recent research effort has been put into in-generation statistical watermarks. Aaronson (2022) pioneered the non-distortionary Gumbel-max method which samples via a pseudorandom rule instead of multinomial sampling for the next token selection. The Maryland method (Kirchenbauer et al., 2023), although distortionary, is a popular reference method since it boosts a portion of the logits at each generation step. Follow-up methods such as Hu et al. (2024); Wu et al. (2024) scaled the Maryland logit bias with the LLM distribution, achieving a non-distorted next-token distribution. Other non-distorting methods include inverse sampling (Kuditipudi et al., 2024), and SynthID-Text (Dathathri et al., 2024), which is based on multilayer tournament sampling. Permute-and-Flip decoding (Zhao et al., 2024) permutes the candidates with secret randomness and introduces a Gumbel-like watermark test, without changing the output distribution. Finally, TextSeal (Sander et al., 2026) extends Gumbel-max method by randomly routing to one of two secret keys, balancing detectability and sampling diversity, while also being compatible with speculative decoding where target model

[2]The philosophical paradox of the Ship of Theseus: does a ship fundamentally remain the same object after its original parts are replaced over time?

cannot use the same key as the draft. Using multiple keys has also been explored in the context of watermark ensembling, towards increasing the watermark signal (Wu et al., 2026).

Watermarking relies on imperceptibly altering the LLM distribution, thus it works better in high-entropy generations. Tsur et al. (2026) propose Pareto-optimal methods that are detectable even in low-entropy scenarios. Giboulot & Furon (2024) propose WaterMax, a best-of-N scheme that uses a watermarking method's randomness to sample text chunks that are both high-quality and strongly detectable.

B.2 Public detectability and unforgeable watermarks

Christ & Gunn (2024) introduced robust cryptography-based watermarking. Subsequently, Fairoze et al. (2025a) proposed a publicly-detectable scheme for language models by embedding cryptographic signatures in the payload, acknowledging the capacity bottleneck of text. Fairoze et al. (2025b) formalize the requirements for a robust, unforgeable and publicly-detectable watermark, showing existence in principle but intractability in practice without increase in embedding robustness. Other lines of work maintain security in the public setting without cryptographic guarantees, using independent embedding and detection constructions. For instance, Liu et al. (2024) propose a watermark that is embedded and detected by neural networks, and is thus unforgeable even with a publicly shared detector. Bashir (2026) propose a publicly detectable watermark based on the Maryland method that asymmetrically biases public and private green-listed tokens. Aremu et al. (2025) propose the random switching of watermark keys so that genuinely watermarked texts correlate with only one item from the key set, while forged texts produced from a watermarked corpus would correlate with multiple keys.

The concurrent work of Feng et al. (2026) co-embeds a robust and a fragile unbiased signal with independent keys, which can indicate tampering under light edits, including piggyback spoofing. The split score idea was also shared by Shen et al. (2026); Li et al. (2026), who proposed forgery-proof watermarks with two sources of signal. However, their constructions are distortionary and not strongly detectable, while their split construction is not oriented towards public detection.

B.3 Removal and forgery

A recent line of work aims at learning a text watermark purely through an AI platform's outputs (Gu et al., 2024). Jovanović et al. (2024) proposed a practical watermark stealing method, which reverse-engineers a Maryland scheme with a known pseudorandom function (PRF), further generalized to any PRF in De-Mark (Chen et al., 2025). A correlation-based test to classify such forged text is proposed by Gloaguen et al. (2025). Similarly, Zhang et al. (2026) use an adaptive method to steal any watermark that performs logit reweighting.

Pang et al. (2024) demonstrate that stealing a non-distortionary schema is possible given access to a watermark detector that provides a detection score per sentence. In practice, they assume a model API that provides top-k completions to considerably reduce the vocabulary search space. Huang et al. (2026) propose RLSpoofer, a forgery paraphraser for biased watermarks trained via reinforcement learning on unwatermarked and watermarked pairs. Diaa et al. (2025) optimize removal instead: they curate preference pairs with a surrogate key and tune open-weight paraphrasers with DPO against known watermark algorithms, evading all tested watermarks at negligible quality loss without detector queries, and the tuned paraphrasers transfer to unseen watermarks. This extends their optimization-based adaptive attacks on image watermarks (Lukas et al., 2024) to text.

C Additional Details

C.1 Distortion-free watermarking

Following Dathathri et al. (2024), we state the two levels of non-distortion used in Sec. 2.2. Let p_t be the LLM next-token distribution with $p_t(v)$ the mass on candidate v, x_t the sampled token, and $r_t(v)$ the seed value for v at position t. Let $\mathcal{V}$ be the vocabulary, $\mathcal{R}$ the seed space, and $\mathcal{K}$ the key space.

Single-token non-distortion (Dathathri et al., 2024, Def. 16). A sampling rule A mapping (p_t, r) to a token in $\mathcal{V}$ is *single-token non-distortionary* if for any p_t and any $v \in \mathcal{V}$:

$$\mathbb{E}_r \left[\Pr(A(p_t, r) = v)\right] = p_t(v),$$

where the expectation is over a uniform seed $r \in \mathcal{R}$. Marginalizing over the seed, each token is sampled with exactly its original LLM probability. This is a property of the sampling rule alone (e.g. Gumbel-max or SynthID-Text tournament sampling), independent of how the seed is produced at each step. In a deployed scheme the seed at t is the PRF output keyed by k, so averaging over k matches averaging over r when each context is fresh.

Single-sequence non-distortion (Dathathri et al., 2024, Def. 20, single-response case). A watermarking scheme p_{wm} is *single-sequence non-distortionary* if, for any prompt c and response $x_{1:n} \in \mathcal{V}^n$:

$$\mathbb{E}_k \left[p_{\text{wm}}(x_{1:n} \mid c, k)\right] = p_{\text{LM}}(x_{1:n} \mid c),$$

where $p_{\text{wm}}(x_{1:n} \mid c, k) = \prod_{t=1}^{n} q_k(x_t \mid c, x_{1:t-1})$, $q_k(\cdot \mid c, x_{1:t-1})$ is the law of $A(p_t, r_t(k))$ with $p_t = p_{\text{LM}}(\cdot \mid c, x_{1:t-1})$, and the expectation is over uniform $k \in \mathcal{K}$. This is strictly stronger: the *joint* distribution over a full response must match the original LLM, not just the per-token marginals. A single-token non-distortionary sampler with a sliding-window seed violates it whenever the same context repeats within a generation, since the PRF then returns identical values and re-selects the same token. *Repeated context masking* (Dathathri et al., 2024) restores it by falling back to unwatermarked sampling on repeated contexts. In practice, TextSeal and SynthID-Text reach answer-level non-distortion through this generation-time dedup: repeated h-grams are rare with $h \geq 3$, so masking them changes little while removing the deterministic repetitions. This scheme has no effect on model output quality and is used in production (Dathathri et al., 2024; Sander et al., 2026; Anthropic, 2026).

C.2 Attacks

We provide additional details for the attack families described in Sec. 2.3. For the rephrasing attack, in all settings, we use the prompt:

```
f"Rephrase the following text while preserving its meaning.
Output only the rephrased version, nothing else.
Text: {text}
Rephrased:"
```

Since the word edits attack relies on an MLM for word replacement, which cannot capture math, we explicitly filter out the math-related samples from ELI5 for a fair comparison before isolating the first 1k samples. We detail the word edits attack in Algorithm 1. The method is similar to the self-information attack of Cheng et al. (2025) and the adversarial BERT attacks of Li et al. (2020) and Garg & Ramakrishnan (2020).

Adaptive white-box attacks. These are the attacks of Sec. 4.4, run at $p_{\text{target}}{=}10^{-3}$ with overshoot factor m. For word edits, Alg. 1 is unchanged except for the stopping test, which becomes $p_{\text{pub}} \geq m\, p_{\text{target}}$ for removal and $p_{\text{pub}} < p_{\text{target}}/m$ for forgery. For rephrasing, the fixed additive bias λ is replaced by $\max(0, \log(p_{\text{target}}/p_{\text{current}}))$, recomputed at every decoding step from p_{current}, the running public p-value of the text generated so far. Both read p_{pub}, so both need at least the gray-box tier.

Adaptive black-box attacks. In this setting, the public detector answers only $\mathbb{1}[p_{\text{pub}} < 10^{-3}]$.

For the black-box word edits attack, we first split the text in whitespace-delimited words, and draw uniformly words one by one without replacement. Replacements come from the same `roberta-large` MLM as in the white-box attack ($k{=}50$, minimum probability 10^{-4}). One word is replaced at a time and the detector is queried once per edit, so the number of queries equals the edit count. The attack stops at the first edit whose verdict flips, or at the edit budget. In the adaptive case, the overshoot factor m of Sec. 4.4 is applied in edits: we keep editing up to m times the number needed for the flip, while staying under the edit budget. For instance, if $m = 1.5$ and the 10th edit flipped the decision, we would edit 15 words if the budget is bigger than 15 and exactly the budget if the budget is lower than 15. $m{=}1$ is the stopping rule from before, and we compare with the no-box setting which spends the whole budget and never uses the verdict (equivalent to a large m).

Algorithm 1 Word edits for Watermark Removal (Forgery in comments).

Require: Text x, score threshold ζ, p-value goal θ, max edits B, max passes I, words per pass w

$n \leftarrow 0$, $i \leftarrow 0$ ▷ Edit count, pass count
repeat
  $p, \{r_v\} \leftarrow \textsc{Detector}(x)$ ▷ p-value and per-token scores (white-box detector)
  $i \leftarrow i + 1$
  $W \leftarrow$ the w words with highest-r tokens ▷ Expanded subword tokens (Forgery: lowest-r)
  for each word $w \in W$ **do** ▷ Right to left, keeping earlier scores valid
    $C \leftarrow \text{MLM}(x, [w])$ ▷ Mask w and get a replacement candidate set
    for each candidate $c \in C$ **do**
      $x_c \leftarrow x$ with c in place of w
      $_, \{r_t(c)\} \leftarrow \textsc{Detector}(x_c)$ ▷ Score by querying detector
      $\mathbf{S}(c) \leftarrow \sum_{t \in \text{tok}(c)} - \log(1 - r_t(c))$ ▷ $\text{tok}(c)$: tokens of x_c containing a part of c
    end for
    $C_\zeta \leftarrow \{c \in C : \max_{t \in \text{tok}(c)} r_t(c) < \zeta\}$ ▷ Select weak candidates (Forgery: $> \zeta$)
    if $C_\zeta \neq \emptyset$ **then**
      Replace w with $\arg\min_{c \in C_\zeta} \mathbf{S}(c)$ in x ▷ Best candidate (Forgery: $\arg\max$)
      $n \leftarrow n + 1$
    end if
  end for
until $p \geq \theta$ **or** $n \geq B$ **or** $i \geq I$ ▷ (Forgery: $p < \theta$)
return x

For the black-box rephrasing attack, the attacker increases the temperature $T \in \{0.2, 0.4, \ldots, T_{\max}\}$ and submits each paraphrase, keeping the first whose verdict flips.

C.3 Tampering Test

We use $R = 5 \times 10^4$ permutations to compute the p-values, which allows for a minimum p-value of 2×10^{-5}, covering the 10^{-3} detection threshold. We also show that the tampering test is calibrated on its nominal FPR. Fig. 7 shows that the empirical FPR closely follows its nominal value, indicating that the tampering test is calibrated on its null assumption of channel symmetry. Additionally, we verify each watermark test's calibration, since we use the fused test in the full detection pipeline, and use only one channel for the public detector.

Rate of the two stages together. The plots in Figs. 7 are per-stage marginals: each verifies one test against its own null in isolation. Both corpora in Fig. 7 probe the single null H_0 of Sec. 3, which is why one unwatermarked and one watermarked set both belong there. Write η for the per-stage FPR level, distinct from the routing fraction α of Sec. 3. Running the two stages in sequence at a common η gives unwatermarked text two chances to be flagged: the first stage may call it watermarked, and failing that the second may call it scrubbed. A genuine watermarked text, whose first-stage verdict is correct, faces only the forgery tail and so only one. The permutation relabels the channels: swapping the two labels at a position leaves $s_{\text{pub},t} + s_{\text{priv},t}$ unchanged, so the fused score, and with it p_{fus} and y_{fus}, is constant across every draw of the null. The tampering p-value is therefore exactly uniform conditionally on the first-stage verdict, which is what keeps each stage at its nominal level. This relies on the first stage reading the sum of the channels, not on it executing first: had it thresholded p_{pub}, selection would favor large $\mathbf{T}$, since $\text{Corr}(\mathbf{S}_{\text{pub}}, \mathbf{S}_{\text{pub}} - \mathbf{S}_{\text{priv}}) = 1/\sqrt{2}$ at equal variances, and the forgery tail would have increased FPR on untampered text. The rate at which unwatermarked text draws *any* flag is then $\eta + (1 - \eta)\eta \approx 2\eta$ instead of η. Tab. 3 measures this on 799 unwatermarked generations. The measured union is below 2η only because the fused watermark test is itself slightly conservative on this corpus. At 10^{-3} the corpus is too small to resolve the rate.

On watermarked untampered text the first stage is correct, so there is only one way to err: a false forgery verdict against a genuine user. That one stays on nominal, firing on 5.31%, 1.05% and 0.17% of the 5992 untampered generations at η of 0.05, 0.01 and 10^{-3}. The attack rates we report are likewise conditional on a stage-one verdict and so are governed by the single-stage level. The factor of two applies to the unconditional statement, i.e. the rate at which the pipeline flags an

Table 3: Flag rates of the two-stage pipeline on 799 unwatermarked generations, in percent. The second stage is the removal tail whenever the first stage returns *not watermarked*, and the forgery tail otherwise. Each stage holds its nominal level η, but unwatermarked text can fail either of them and so is flagged at roughly 2η.

η	Watermark test	Tampering test	Either stage	$\eta + (1-\eta)\eta$
0.05	3.88	5.01	8.51	9.75
0.02	1.13	2.38	3.50	3.96
0.01	0.63	0.88	1.50	1.99

Table 4: Average CPU cost of watermark detection and the tampering test, on the same 1k texts. Both tests share one fused K-key scoring pass (the 43.9 ms detection cost). The tampering test adds a permutation over R replicates of the already-computed score matrix, so the tampering columns report the permutation alone. Larger R lowers the smallest resolvable p-value but scales the cost linearly. We use $R = 5 \times 10^4$ by default.

	Detection	Tampering			
Permutations R	–	5×10^3	5×10^4	2×10^5	1×10^6
Resolvable $\min p$	analytic	2×10^{-4}	2×10^{-5}	5×10^{-6}	1×10^{-6}
Test duration (ms)	43.9	23.3	276.9	1105.1	5518.5

arbitrary clean text at all, and splitting the budget as $\eta/2$ per stage restores a joint η at a small cost in power.

Furthermore, we examine the computational cost of the tampering test compared to the watermark test. We present the results in Tab. 4.

A permutation instead of a binomial test. Exchangeability also admits a binomial test: count the positions with $d_t > 0$ against $\mathrm{Binomial}(n, \frac{1}{2})$, which needs no sampling but discards $|d_t|$. Neither dominates (Tab. 5). Rephrasing moves nearly every position, so the signs alone carry the evidence and the binomial test is more powerful. Forgery and word edits concentrate their damage in a few

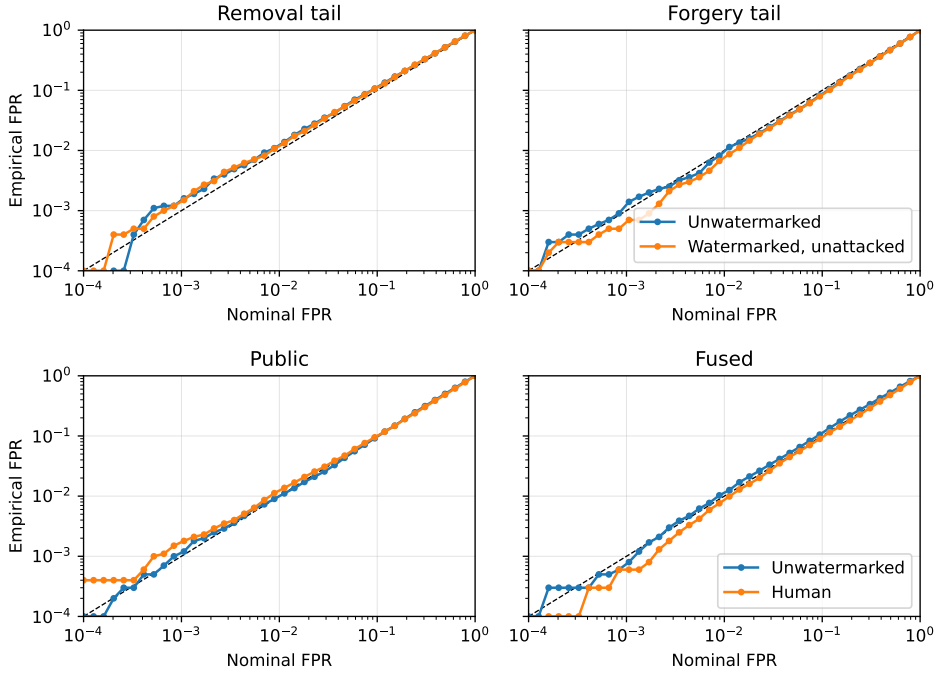


Figure 7: *(Top)* Null calibration of the tampering tests on two sets of 10k untampered texts, both generated by Qwen2.5-7B. One is unwatermarked and the other is watermarked but untampered, covering both tampering nulls. *(Bottom)* Null calibration of the watermark tests for both detectors. We use two sets of 10k unwatermarked texts, one generated from Qwen2.5-7B, and the other being the human-written answers in ELI5.

Table 5: Attack detection rate (%) at $p < 10^{-3}$ for the two exact tests, symmetric $(2, 2)$, white-box attacks, 1000 documents per row. Bold marks the better test per attack. The binomial is more powerful on rephrasing, the permutation on the concentrated attacks.

Attack	Binomial	Permutation
Untampered (FPR)	0.3	0.1
Rephrase removal	**86.9**	82.4
Rephrase removal (additive)	**40.1**	25.1
Word-edit removal	0.6	**1.2**
Rephrase forgery	12.1	**25.2**
Word-edit forgery	3.2	**9.5**

positions, and the permutation doubles the detection rate. We keep the permutation because its worst case is the better of the two (1.2% against 0.6%) and forgery is the threat the test exists to catch.

Routing must stay uniform. Exchangeability holds because each token goes to either key with probability $\frac{1}{2}$, so relabelling the two is legitimate. Routing them unequally breaks it: a token is then more likely to have come from one key, the sign of d_t is a biased coin, and the sign-flip null no longer applies.

C.4 Attack Examples

We visualize how the different attack goals manipulate the watermark score distributions in the score space, and how the tampering test covers different areas than the fused watermark test in Fig. 8. We qualitatively show how our white-box attacks manipulate actual texts to remove the watermark in Fig. 9, and to forge the watermark in Fig. 10.

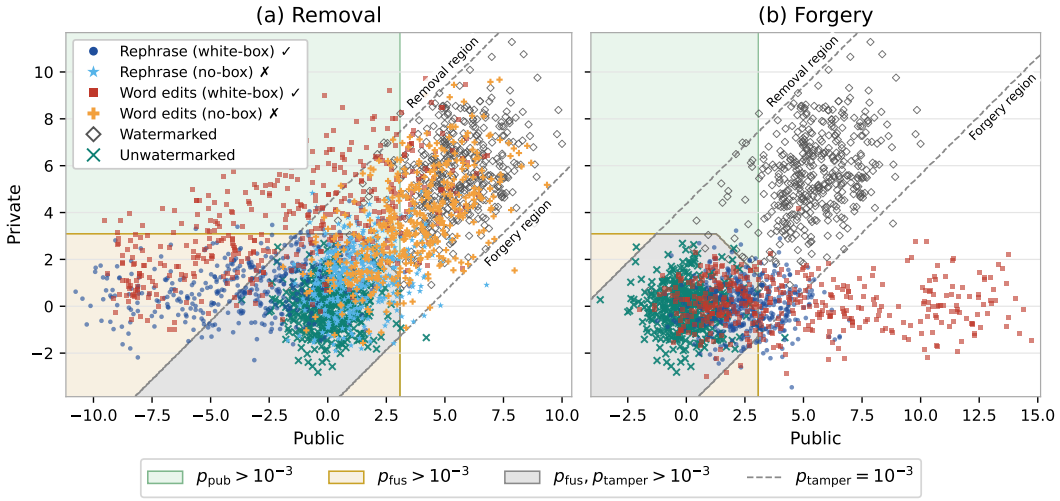


Figure 8: The scores (standardized using their mean and variance) of each channel and the areas where each test fails to reject its null hypothesis (e.g. $p_{\text{pub}} > 10^{-3}$ means that the public p-value is high enough for the text to be labeled as unwatermarked). The attacked data points are stratified samples at different strength levels from our main experiments on Qwen2.5-7B (additive rephrasing with a temperature sweep, and word edits with a replacement rate sweep). The public detector can capture genuinely watermarked text, but is far less sensitive to tampered text than the fused detector and the tampering test (diagonal band, signifying large score deviations). White-box attacks tend to move the scores outside of the tampering test band, into either the removal or the forgery region, whereas the no-box removal attacks move the scores only within the balanced score band. The white-box removal attacks also have similar private scores to their no-box counterparts, but are denser towards lower public score areas.

Removal caught by fused:

Attack: Removal, Word edits/white-box, Edit rate = 0.05

Prompt: How the fuck does Facebook know about people I know?!

	Public detector	Full detector	
		Fused	Tamper
p-value	$1.5\times10^{-6} \rightarrow \mathbf{1.3\times10^{-1}}$	$7.6\times10^{-13} \rightarrow \mathbf{8.7\times10^{-5}}$	$\mathbf{9.8\times10^{-1}}$
Verdict @ 0.1% FPR	**Unwatermarked**	**Watermarked**	**None**

Source (223 words)

Facebook's ability to understand who you know is based on several factors, primarily involving how users interact with the platform and the information they provide. Here are the key ~~mechanisms:~~ 1. **Tagging**: When you tag someone in a photo or post, you directly inform Facebook that you know them. 2. **Mutual Friends**: If you and someone else have mutual friends who are also on Facebook, the system can infer that you know them. 3. **Interactions**: Facebook looks at your interactions, such as ~~messages,~~ comments, and ~~shares,~~ to see if you have a history of interacting with someone. 4. **Friend List**: If you manually add someone to ~~your~~ friend list, Facebook [...]

Attacked (223 words)

Facebook's ability to understand who you know is based on several factors, primarily involving how users interact with the platform and the information they provide. Here are the key ones 1. **Tagging**: When you tag someone in a photo or post, you directly inform Facebook that you know them. 2. **Mutual Friends**: If you and someone else have mutual friends who are also on Facebook, the system can infer that you know them. 3. **Interactions**: Facebook looks at your interactions, such as your comments, and events to see if you have a history of interacting with someone. 4. **Friend List**: If you manually add someone to a friend list, Facebook [...]

Removal caught by tampering:

Attack: Removal, Rephrase/matched/white-box/additive, T = 1

Prompt: How do we know all the money the government is getting from bank settlements is going back to the people?

	Public detector	Full detector	
		Fused	Tamper
p-value	$5.7\times10^{-4} \rightarrow \mathbf{1.0\times10^{0}}$	$2.3\times10^{-9} \rightarrow \mathbf{8.4\times10^{-1}}$	$\mathbf{8.0\times10^{-5}}$
Verdict @ 0.1% FPR	**Unwatermarked**	**Unwatermarked**	**Removal**

Source (330 words)

~~Ensuring~~ that funds collected ~~by the government~~ from bank settlements are effectively ~~used~~ for the ~~benefit of the people involves a combination of transparency, accountability,~~ and ~~oversight mechanisms. Here are some key ways this is typically managed: 1. **Transparency~~ and Reporting**: ~~Government agencies often~~ publish detailed ~~reports on~~ how funds are collected and ~~spent.~~ These ~~reports include~~ the ~~amounts received, the sources,~~ and ~~the intended use~~ of these ~~funds.~~ This ~~helps to ensure that the~~ public ~~can see where their money is going.~~ 2. ~~**Public~~ Audits and Oversight**: ~~Independent auditors or the~~ public ~~can review financial statements~~ and ~~transactional details to verify the accuracy~~ of financial ~~records.~~ This helps ~~catch~~ [...]

Attacked (246 words)

By integrating these measures, governments can enhance the likelihood that funds collected from bank settlements are used effectively for the public benefit: 1. **Increasing Transparency and Regular Reporting**: Governments publish detailed records detailing how funds are collected and utilized. These accounts detail the quantities, origins, and designated uses of these resources. This promotes public oversight and trust. 2. **Independent Audits and Oversight**: The public and regulatory bodies conduct reviews of financial records to ensure accuracy. This helps in detecting discrepancies or mismanagement promptly. 3. **Legislation and Oversight Bodies**: Government entities and regulatory bodies scrutinize financial practices to ensure compliance with laws and regulations, guaranteeing funds are allocated accordingly. 4. **Public [...]

Successful removal:

Attack: Removal, Rephrase/matched/no-box/additive, T = 1

Prompt: in football whats the point of wasting the first two plays with a rush - up the middle - not regular rush plays i get those

	Public detector	Full detector	
		Fused	Tamper
p-value	$2.5\times10^{-12} \rightarrow \mathbf{5.3\times10^{-1}}$	$1.9\times10^{-15} \rightarrow \mathbf{8.8\times10^{-2}}$	$\mathbf{6.5\times10^{-2}}$
Verdict @ 0.1% FPR	**Unwatermarked**	**Unwatermarked**	**None**

Source (333 words)

In football, ~~you're right that~~ rush plays are ~~a common~~ and ~~strategic use of the first few plays, especially when a team is attempting to establish~~ the run or ~~set up their offense. However, rushing~~ up the ~~middle with no specific objective~~ is less common and ~~may~~ be ~~seen~~ as ~~"wasting" plays in the sense that it doesn't have~~ a clear tactical purpose. The term "rush up the middle" without a specific plan can be misleading. Rushing up the middle ~~is~~ a ~~term often~~ used ~~for running plays where~~ the ~~ball carrier moves directly towards the opponent's middle. This can be part of a broader strategy, even if it's not~~ [...]

Attacked (207 words)

In football, rush plays are indeed frequent and strategic, particularly for establishing the run or setting the offense's pace. Rushing straight up the middle, however, is less common and can be viewed as "wasteful," lacking a clear tactical purpose. The term "rush up the middle" without a specific plan can be misleading. Typically, this involves running plays directly towards the opponent's middle, which may serve several broader strategic objectives: 1. **Establishing a Running Game**: Rushing up the middle can help a team begin by running the ball, potentially controlling the game's tempo and forcing adjustments from the defense. 2. **Feinting**: This play might be used to mislead the defense, drawing [...]

Figure 9: Watermark removal examples with white-box attacks, either rephrasing or word edits. While the public score may be lowered enough to indicate unwatermarked text, the fused watermark test along with the tampering test can discover that the text is actually either watermarked or tampered, indicating key correlation with the provider's LLM.

Forgery caught by fused:

Attack: Forgery, Word edits/white-box, Edit rate = 0.05

Prompt: How does ring re-sizing work? How is it possible to do it without breaking or warping the metal?

	Public detector	Full detector	
		Fused	Tamper
p-value	$1.6\times10^{-1} \rightarrow \mathbf{1.7\times10^{-5}}$	$1.2\times10^{-1} \rightarrow \mathbf{1.6\times10^{-3}}$	$\mathbf{1.0\times10^{0}}$
Verdict @ 0.1% FPR	**Watermarked**	**Unwatermarked**	**None**

Source (202 words)

Goldsmith here, not from de US so ill try my best to explain it but im not really used to explaining this in english. There are different ways ~~to~~ resize a ring. If its a weddingband without stones or ornaments etc you can just stretch it on a machine. When you stretch it the band does get thinner but you normally dont really see any difference. You can also shrink the ring by pressing the ring in a V-shape so it gets smaller. If you try to stretch a ring ~~with~~ a stone it will break ~~cause~~ it is weaker ~~in~~ the place ~~with~~ the stone. So if you cant [...]

Attacked (202 words)

Goldsmith here, not from de US so ill try my best to explain it but im not really used to explaining this in english. There are different ways that resize a ring. If its a weddingband without stones or ornaments etc you can just stretch it on a machine. When you stretch it the band does get thinner but you normally dont really see any difference. You can also shrink the ring by pressing the ring in a V-shape so it gets smaller. If you try to stretch a ring round a stone it will break because it is weaker through the place below the stone. So if you cant [...]

Forgery caught by tampering:

Attack: Forgery, Rephrase/matched/white-box/Gumbel, T = 1

Prompt: What's wrong with a flat tax? From a middle-class perspective.

	Public detector	Full detector	
		Fused	Tamper
p-value	$5.9\times10^{-1} \rightarrow \mathbf{7.2\times10^{-7}}$	$7.0\times10^{-1} \rightarrow \mathbf{9.3\times10^{-5}}$	$\mathbf{2.4\times10^{-4}}$
Verdict @ 0.1% FPR	**Watermarked**	**Watermarked**	**Forgery**

Source (198 words)

~~So let's start with~~ a simple ~~fact. Changing~~ how we ~~levy tax doesn't change~~ the ~~total~~ amount ~~we levy, changing~~ how we ~~spend taxes does. There's a lot of argument surrounding whether we should spend less, but~~ those ~~arguments~~ are separate from who ~~should pay~~ what ~~we pay now.~~ If the 'flat tax' is ~~set~~ at ~~exactly what you pay right now (on average), then~~ the ~~people~~ who ~~are richer than you will pay a lot~~ less, ~~and the people~~ who ~~are poorer than you will~~ have to pay a lot more. ~~Those~~ people ~~just don't actually~~ have the ~~money~~ to ~~make up for a~~ tax ~~break~~ for the ~~rich~~ [...]

Attacked (176 words)

To begin, a simple fact is that altering how we collect taxes does not affect the overall amount collected; it depends on how we allocate those taxes. Debates about reducing government spending are separate from determining who pays what taxes. If the 'flat tax' is imposed at the average amount currently paid, those who earn more would end up paying significantly less, while those who earn less would have to pay a lot more. This is impractical because people who earn less do not have the financial means to support tax breaks for the wealthy. Consequently, the tax rate for the 'flat tax' would likely be higher than what you [...]

Successful forgery:

Attack: Forgery, Word edits/white-box, Edit rate = 0.05

Prompt: As a non-gamer, can someone please explain what everyone's problem is with No Man's Sky.

	Public detector	Full detector	
		Fused	Tamper
p-value	$2.2\times10^{-2} \rightarrow \mathbf{1.8\times10^{-6}}$	$3.1\times10^{-3} \rightarrow \mathbf{1.0\times10^{-5}}$	$\mathbf{7.7\times10^{-3}}$
Verdict @ 0.1% FPR	**Watermarked**	**Watermarked**	**None**

Source (194 words)

It's like if you saw a commercial for a burger joint that said "Buy one super deluxe burger meal, get another for half off *." The * would point ~~to~~ some small print that said it was only a supported promotion if you ~~got~~ the super deluxe burger without lettuce, onion, or cheese, ~~and that~~ you only got 1 order of fries and 1 drink, despite the common assumption that you'd get all these things with how the promotion is worded. No Man's Sky was hyped up before launch with all kinds of claimed features, but the complaints against ~~it~~ are basically that it hasn't lived up to the ~~hype,~~ [...]

Attacked (194 words)

It's like if you saw a commercial for a burger joint that said "Buy one super deluxe burger meal, get another for half off *." The * would point at some small print that said it was only a supported promotion if you ordered the super deluxe burger without lettuce, onion, or cheese, meaning essentially you only got 1 order of fries and 1 drink, despite the common assumption that you'd get all these things with how the promotion is worded. No Man's Sky was hyped up before launch with all kinds of claimed features, but the complaints against launch are basically that it hasn't lived up to the claims [...]

Figure 10: Watermark forgery examples with white-box attacks, either rephrasing or word edits. While the public score may be increased enough to forge a watermark signal, the full detector is harder to deceive, while the tampering test may explicitly indicate forgery.

D Additional Results

D.1 Extended Numbers of the Main Results

Tab. 6 and Tab. 7 give the per-attack numbers behind Fig. 4, for removal and forgery respectively. For each attack we report the median $-\log p$ of the public, private, and fused scores, the attacker-visible public success rate $\mathrm{ASR}_{\mathrm{pub}}$, the success rate after the fused-score test $\mathrm{ASR}_{\mathrm{fus}}$, the attack detection rate of the tampering test ADR, the effective threat $\mathrm{ASR}_{\mathrm{net}}$ that survives both stages, and the text distortion $1-$BERT-F1. We note that the tables report the rates unconditionally. However, the bar heights in Fig. 4 (with a full bar equal to $\mathrm{ASR}_{\mathrm{pub}}$), are conditioned on the public success, since only the texts that fool the public detector are examined. The two views are almost equivalent and only differ on rare occasions where texts do not evade the public detector but evade the fused one. From the attacker's perspective, we follow the per-channel score presentation of Fig. 5b and examine the strength λ as an additional attack parameter, showing its effect in Fig. 11.

Regarding distortion quantification, we revisit the white-box attack sweeps of Fig. 5b and measure their distortion using additional text quality metrics in Fig. 12 for the removal goal, and Fig. 13 for the forgery goal. From left to right, we present the distortion with: $1 -$ BERT-F1 (the default metric used throughout), the cosine distance using MiniLM (Wang et al., 2020), the log perplexity ratio from GPT-2, and word overlap distance.

D.2 Generalization to Other Models

We repeat the main experiments with Gemma-3-4B-IT (Gemma Team, 2025) as the text generator, on the same dataset and settings, with Gemma-3-1B-IT as the *matched* proxy rephrasing model. How much watermark signal a text carries depends on the generator's output entropy, and Gemma-3-4B is less entropic than Qwen2.5-7B, so at a shared temperature its texts start with far less signal to remove and no attack would be comparable. We therefore raise its sampling temperature to 1.5 to share approximately the same p-value per 100 tokens as on Qwen, and attack from there.

Tab. 8 shows that, once the two generators start from the same signal, the defense behaves the same way on a different model family: word edits and forgery reach the same rates as on Qwen, informed and uninformed alike. Rephrasing is the only exception because of the rephrasing model which is Gemma-1B in the case of Gemma-4B. It rewrites much more heavily than the 3B proxy used on Qwen, and an attack that rewrites that much stops steering and starts behaving like an uninformed paraphrase (same mechanism as App. D.4). The table also includes SmolLM2-1.7B-Instruct, a smaller generator with bigger entropy, whose watermark is stronger at the same number of tokens. For this model, we use SmolLM2-360M-Instruct for the rephrasing attack. Its watermark is the strongest of the three, so its absolute removal rates are the lowest, but the ordering between attacks stays unchanged.

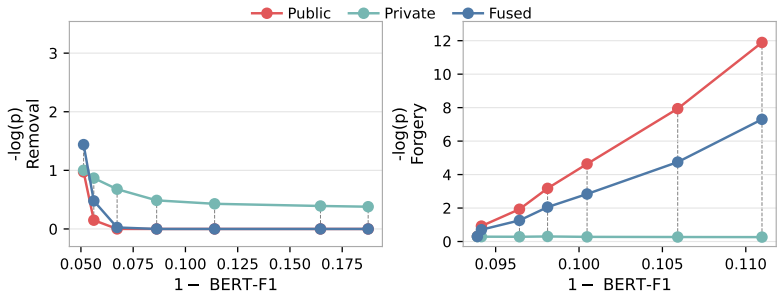


Figure 11: Additive rephrasing with the same model and dataset at $T{=}1$, but with a strength sweep $\lambda \in \{0, 0.5, 1, 1.5, 2, 3, 4\}$. The strength is similarly effective to temperature at manipulating the public score, and the fused score alongside it. However, extreme strengths also cause large channel deviations for forgery, which are detected by the tampering test.

D.3 Watermark Strength Before Attack

Tab. 9 reports the three p-values the detector exposes: public, private and fused, before any attack, on the first 100 tokens of each generation. As expected, the two channels are interchangeable, while the fused p-value is lower. The scores are modest because they are measured on a 100-token prefix, whereas a full generation averages 360 tokens and the fused test then flags 98.3% of untampered watermarked texts at the same threshold (Tab. 6). Across generators the watermark is much stronger on SmolLM2 than on Gemma 3 at equal length, as it directly depends on the entropy of the model (App. D.2). Raising Gemma's sampling temperature recovers most of the gap, which is why we generate with it at T=1.5 throughout.

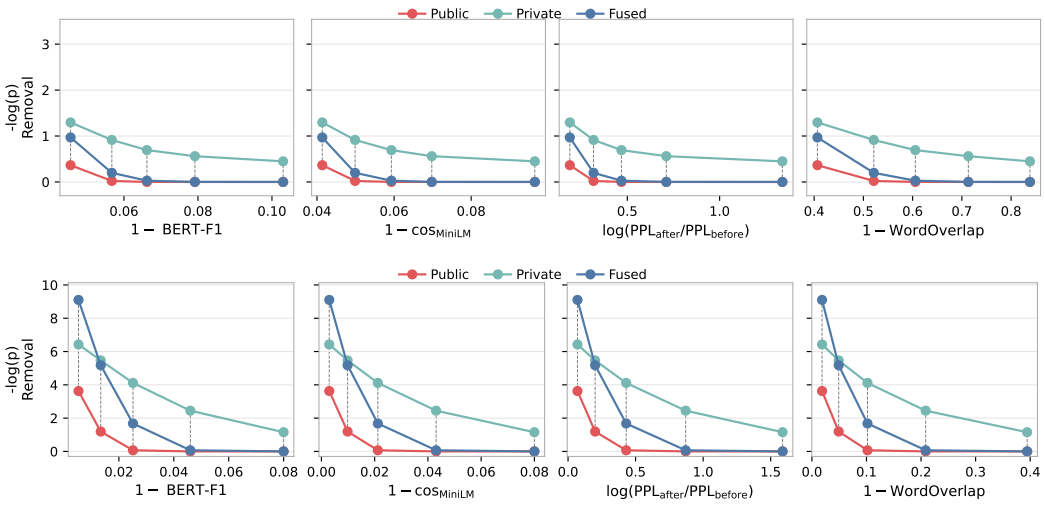


Figure 12: Removal watermark score per channel over quality degradation with the rephrasing attack (top) and word edits (bottom). The settings are the same as Fig. 5, namely temperature $T \in \{0.5, 0.8, 1.0, 1.2, 1.5\}$ and edit fractions $\{2, 5, 10, 20, 40\}\%$. The vertical axes are the same across all panels, but the horizontal axes show the different distortion measurements. Rephrasing causes stronger distortion on pure word overlap compared to word edits (which cause their nominal edit rate), but weaker distortion on fluency as quantified with the perplexity ratio, and similar distortions to the LM metrics (BERT and MiniLM) that capture both.

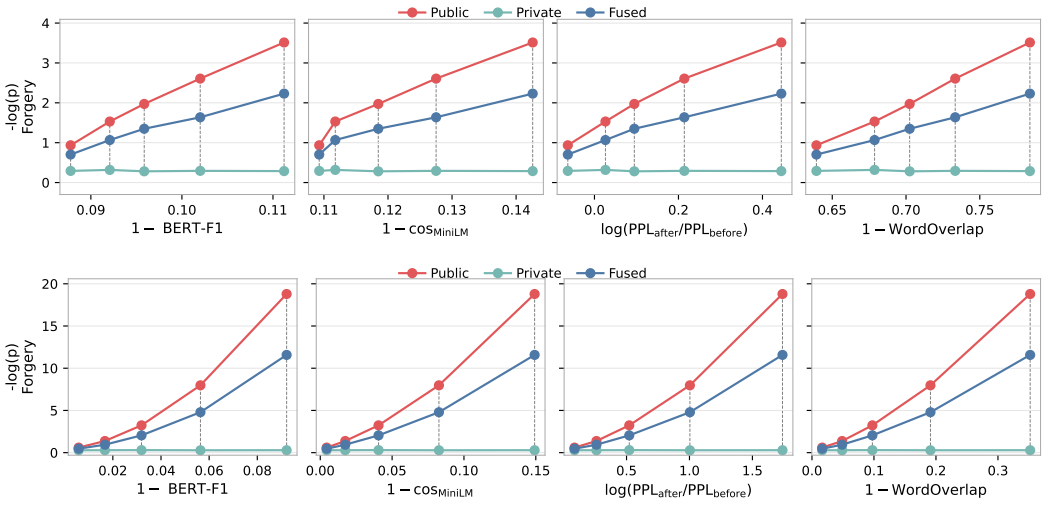


Figure 13: Forgery watermark score over quality degradation with rephrasing (top) and word edits (bottom).

Table 6: Per-attack removal results behind Fig. 4, over 1k attacked texts. Success rates ASR are lower is better for the defender, and ADR and BERT-F1 are higher is better. Light blue marks the representative operating points reported in the main text (Fig. 4).

	Median $-\log p\uparrow$			Attacker	Defender			Distortion
Attack	Public	Private	Fused	$\text{ASR}_{\text{pub}}\downarrow$	$\text{ASR}_{\text{fus}}\downarrow$	ADR↑	$\text{ASR}_{\text{net}}\downarrow$	$1-\text{BERT-F1}\downarrow$
Watermarked	6.8	7.0	12.9	9.0%	1.7%	0.0%	1.7%	0.000
Rephrase (matched, no-box, add, T=0.5)	1.5	1.7	2.5	79.1%	58.8%	0.2%	58.7%	0.039
Rephrase (matched, no-box, add, T=0.8)	1.2	1.3	1.8	89.2%	71.3%	0.3%	71.1%	0.045
Rephrase (matched, no-box, add, T=1)	0.9	1.0	1.5	92.3%	82.0%	0.4%	81.7%	0.052
Rephrase (matched, no-box, add, T=1.2)	0.8	0.8	1.1	97.5%	93.4%	0.1%	93.3%	0.060
Rephrase (matched, no-box, add, T=1.5)	0.5	0.6	0.7	99.2%	98.3%	0.4%	97.9%	0.074
Rephrase (matched, white-box, add, T=0.5)	0.4	1.3	1.0	98.2%	88.0%	1.9%	86.3%	0.046
Rephrase (matched, white-box, add, T=0.8)	0.0	0.9	0.2	100%	99.0%	12.9%	86.2%	0.057
Rephrase (matched, white-box, add, T=1)	0.0	0.7	0.0	100%	100%	36.6%	63.4%	0.066
Rephrase (matched, white-box, add, T=1.2)	0.0	0.6	0.0	100%	100%	70.3%	29.7%	0.079
Rephrase (matched, white-box, add, T=1.5)	0.0	0.4	0.0	100%	100%	93.0%	7.0%	0.103
Rephrase (matched, white-box, Gumbel, T=0.5)	0.0	1.2	0.4	99.4%	95.5%	13.7%	82.4%	0.063
Rephrase (matched, white-box, Gumbel, T=0.8)	0.0	0.9	0.0	100%	99.7%	57.3%	42.6%	0.073
Rephrase (matched, white-box, Gumbel, T=1)	0.0	0.7	0.0	100%	100%	87.8%	12.2%	0.081
Rephrase (matched, white-box, Gumbel, T=1.2)	0.0	0.6	0.0	100%	100%	97.8%	2.2%	0.093
Rephrase (matched, white-box, Gumbel, T=1.5)	0.0	0.5	0.0	100%	100%	99.8%	0.2%	0.145
Word edits (no-box, 2%)	6.1	6.3	11.6	12.4%	2.1%	0.0%	2.1%	0.004
Word edits (no-box, 5%)	5.1	5.3	9.7	17.7%	3.3%	0.0%	3.3%	0.010
Word edits (no-box, 10%)	4.0	4.1	7.4	31.7%	6.7%	0.0%	6.7%	0.019
Word edits (no-box, 20%)	2.5	2.6	4.4	62.5%	26.9%	0.0%	26.9%	0.032
Word edits (no-box, 40%)	1.2	1.2	1.8	91.8%	77.0%	0.0%	77.0%	0.055
Word edits (white-box, 2%)	3.6	6.4	9.1	36.6%	4.7%	4.3%	4.5%	0.005
Word edits (white-box, 5%)	1.2	5.5	5.2	91.5%	20.9%	12.9%	18.2%	0.013
Word edits (white-box, 10%)	0.1	4.1	1.7	99.8%	79.3%	59.1%	32.4%	0.025
Word edits (white-box, 20%)	0.0	2.4	0.1	99.9%	99.2%	97.1%	2.9%	0.046
Word edits (white-box, 40%)	0.0	1.2	0.0	99.9%	99.2%	99.4%	0.6%	0.080
Average (25 attacks)	0.2	1.4	0.7	83.9%	72.2%	33.8%	39.6%	0.053

Table 7: Per-attack forgery results behind Fig. 4, over 1k attacked texts. Columns follow Tab. 6, and light blue again marks the representative operating points from the main text (Fig. 4).

	Median $-\log p$			Attacker	Defender			Distortion
Attack	Public	Private	Fused	$\text{ASR}_{\text{pub}}\downarrow$	$\text{ASR}_{\text{fus}}\downarrow$	ADR↑	$\text{ASR}_{\text{net}}\downarrow$	$1-\text{BERT-F1}\downarrow$
Unwatermarked	0.3	0.3	0.3	0.1%	0.0%	0.0%	0.0%	0.000
Rephrase (matched, white-box, add, T=0.5)	0.9	0.3	0.7	3.2%	1.0%	0.0%	1.0%	0.088
Rephrase (matched, white-box, add, T=0.8)	1.5	0.3	1.1	9.5%	3.8%	0.0%	3.8%	0.092
Rephrase (matched, white-box, add, T=1)	2.0	0.3	1.3	20.8%	7.5%	10.7%	6.7%	0.096
Rephrase (matched, white-box, add, T=1.2)	2.6	0.3	1.6	38.2%	12.6%	15.9%	10.6%	0.102
Rephrase (matched, white-box, add, T=1.5)	3.5	0.3	2.2	62.0%	29.5%	37.6%	18.4%	0.111
Rephrase (matched, white-box, Gumbel, T=0.5)	1.9	0.3	1.2	21.5%	8.1%	11.1%	7.2%	0.087
Rephrase (matched, white-box, Gumbel, T=0.8)	4.2	0.3	2.6	74.0%	40.2%	42.5%	23.1%	0.090
Rephrase (matched, white-box, Gumbel, T=1)	7.1	0.3	4.3	95.4%	75.4%	72.9%	20.4%	0.093
Rephrase (matched, white-box, Gumbel, T=1.2)	11.5	0.3	7.2	99.9%	94.2%	91.5%	8.0%	0.097
Rephrase (matched, white-box, Gumbel, T=1.5)	24.1	0.3	15.4	99.9%	99.7%	99.7%	0.3%	0.109
Word edits (white-box, 2%)	0.6	0.3	0.5	0.5%	0.5%	0.0%	0.5%	0.006
Word edits (white-box, 5%)	1.4	0.3	0.9	7.8%	3.2%	6.2%	3.0%	0.017
Word edits (white-box, 10%)	3.2	0.3	2.0	56.3%	22.8%	19.3%	18.4%	0.032
Word edits (white-box, 20%)	8.0	0.3	4.8	99.3%	84.9%	77.9%	18.8%	0.056
Word edits (white-box, 40%)	18.8	0.3	11.6	99.9%	99.7%	100%	0.0%	0.092
Average (15 attacks)	3.2	0.3	2.1	52.5%	38.9%	39.0%	9.3%	0.088

D.4 Cross-Tokenizer Rephrasing

The rephrasing attacks so far use a proxy (the rephrasing model) sharing the generator's tokenizer. An attacker has no particular reason to own one: the tokenizer belongs to a model it does not have, and the obvious choice is whatever open model it already runs. We call the attack *matched* when the two tokenizers agree and *cross* when they do not, and run two cross proxies against the Qwen2.5-7B generator. At each decoding step, for every candidate v in its nucleus, the attacker decodes v to a string, appends it to what it has written so far, and submits the whole string. The detector re-tokenizes it in the generator's vocabulary and returns r for the last generator token, under the h generator tokens before it. The attacker then biases the logit of v by $\log r$, exactly as in the matched case. Tab. 10 shows that the token overlap decides the outcome. A proxy that tokenizes similarly to the generator can steer the score more effectively, and the tampering test then flags it. A proxy that

Table 8: Other generators compared to Qwen2.5-7B, at $p < 10^{-3}$ on 1k texts. Gemma-3-4B generations have approximately the same watermark signal as Qwen per 100 tokens, so its rows are directly comparable. SmolLM2-1.7B is not, and carries a stronger watermark. The two-stage defense behaves the same way on all three.

Attack	Generator	$\mathrm{ASR}_{\mathrm{fus}}$	ADR	$\mathrm{ASR}_{\mathrm{net}}$	Distortion
Rephrase, no-box, T=1	Qwen2.5-7B	82.0%	0.4%	81.7%	0.052
	Gemma-3-4B	93.7%	0.2%	93.5%	0.107
	SmolLM2-1.7B	91.7%	0.0%	91.7%	0.097
Rephrase, white-box, T=1	Qwen2.5-7B	100%	36.6%	63.4%	0.066
	Gemma-3-4B	98.8%	12.2%	86.7%	0.115
	SmolLM2-1.7B	100%	50.3%	49.7%	0.116
Word edits, no-box, 10%	Qwen2.5-7B	6.7%	0.0%	6.7%	0.019
	Gemma-3-4B	8.0%	0.0%	8.0%	0.019
	SmolLM2-1.7B	3.4%	0.0%	3.4%	0.017
Word edits, white-box, 10%	Qwen2.5-7B	79.3%	59.1%	32.4%	0.025
	Gemma-3-4B	68.6%	53.1%	32.2%	0.027
	SmolLM2-1.7B	23.7%	26.2%	17.5%	0.023
Forgery, edits, white-box, 10%	Qwen2.5-7B	22.8%	19.3%	18.4%	0.032
	Gemma-3-4B	20.1%	20.4%	16.0%	0.032
	SmolLM2-1.7B	22.9%	14.0%	19.7%	0.031

Table 9: Watermark strength before any attack, at $p < 10^{-3}$ over 1k texts. Both the median $-\log_{10} p$ and the TPR are measured on the first 100 tokens of each generation, so that generators of different verbosity are compared at equal length. Texts shorter than 100 tokens are excluded from those columns. T is the sampling temperature used to generate, and "Answer tokens" the length of the full generation (mean $\pm$ std) in the generator's answers.

			On the first 100 tokens					
			Median $-\log_{10} p$			TPR		
Generator	T	Answer tokens	p_{pub}	p_{priv}	p_{fus}	p_{pub}	p_{priv}	p_{fus}
Qwen2.5-7B	1.0	364 ± 61	2.0	2.0	3.3	24.6%	24.5%	57.7%
Gemma-3-4B	1.5	398 ± 20	1.4	1.5	2.2	10.3%	12.3%	30.2%
Gemma-3-4B	1.0	398 ± 22	0.9	0.9	1.3	1.6%	2.8%	6.4%
SmolLM2-1.7B	1.0	268 ± 97	4.6	4.5	8.3	75.7%	75.4%	96.3%

tokenizes differently cannot steer the score, so the attack reduces to an uninformed paraphrase in the extreme case of little overlap, the one case the tampering test cannot detect, and it thus reaches the highest net removal rate.

Table 10: Matched against cross-tokenizer rephrasing on the Qwen2.5-7B generator, removal at T=1 on 1k texts. "Token overlap" is the fraction of the generator's token spans that the proxy reproduces, measured on the generator's own generations. Both the attack steering and the token score-based tampering detection are applicable only at this overlap.

Attacker	Proxy	Token overlap	$\mathrm{ASR}_{\mathrm{fus}}$	ADR	$\mathrm{ASR}_{\mathrm{net}}$	Distortion
Uninformed	Matched (Qwen2.5-3B)	100%	82.0%	0.4%	81.7%	0.052
	Cross (SmolLM3-3B)	99.3%	98.5%	0.0%	98.5%	0.137
	Cross (Gemma-3-4B)	90.3%	99.1%	0.0%	99.1%	0.079
Informed	Matched (Qwen2.5-3B)	100%	100%	36.6%	63.4%	0.066
	Cross (SmolLM3-3B)	99.3%	99.9%	65.8%	34.2%	0.143
	Cross (Gemma-3-4B)	90.3%	100%	4.9%	**95.1%**	0.081

Table 11: Black-box removal via word edits, at $p < 10^{-3}$ over 1k texts, grouped by edit budget. Within a group, m is the overshoot factor: after the verdict flips at e edits the attacker continues to $m \cdot e$, capped by the budget, so m=1 stops on the crossing and no-box spends the budget without using the verdict. Bold marks the highest $\mathrm{ASR}_{\mathrm{net}}$. ADR is 0.0% everywhere: no stopping rule leaves the channel imbalance the tampering test looks for, so the fused score alone is what limits the attacker.

Edit budget	5%			10%			20%			40%			
Metric	m=1	m=2	No-box	m=1	m=2	No-box	m=1	m=2	No-box	m=1	m=2	m=3	No-box
$\mathrm{ASR}_{\mathrm{pub}}\downarrow$	19.2%	16.9%	17.7%	35.7%	31.5%	31.7%	72.6%	64.5%	62.5%	99.1%	93.4%	94.9%	91.8%
$\mathrm{ASR}_{\mathrm{fus}}\downarrow$	1.8%	1.8%	3.3%	2.3%	3.0%	6.7%	4.6%	16.3%	26.9%	10.4%	54.5%	69.3%	77.0%
ADR$\uparrow$	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
$\mathrm{ASR}_{\mathrm{net}}\downarrow$	1.7%	1.7%	3.3%	2.3%	3.0%	6.5%	4.3%	15.6%	25.6%	10.3%	53.7%	68.4%	75.3%
Distortion$\downarrow$	0.008	0.008	0.010	0.015	0.016	0.019	0.022	0.029	0.032	0.022	0.042	0.052	0.055

D.5 Adaptive Black-Box Attacks

We present the adaptive black-box attacks, as done for the white-box attacks in Sec. 4.4. A black-box detector only answers "watermarked or not", so the attacker cannot use it to choose which words to edit or what to replace them with. They make both choices blindly, as in the no-box attack, and use the verdict only to decide when to stop. As a reminder, the adaptive attacker edits until the verdict flips, and with an overshoot factor m it keeps editing until it has made m times the number of edits needed for the flip, capped by the edit budget (full details in App. C.2). Tab. 11 sweeps m for each edit budget, with the same metrics as Tab. 2 on the same 1k texts.

First, overshooting past the verdict flip only helps when the budget is large enough: e.g. at 5% the three stopping rules give equivalent results, because the attack reaches the budget before the verdict flips. Second, at every budget the most harmful attack is no-box, which spends the full budget, but it is also the one that introduces the most distortion. However, at matched distortion a partial overshoot can beat uninformed attacks. Adaptive black-box attacks therefore improve the ASR-distortion tradeoff over the non-adaptive ones, but only slightly. ADR stays at 0.0%: none of these attacks touches one channel more than the other, so the tampering test never flags them, and the only useful defense is the fact that the test on the fused score is more powerful.

D.6 Distribution of Flipping Distortion

As an alternative to the per-knob averages of Fig. 5 (left), we plot the empirical distribution of flipping distortion, following Imadache et al. (2025). For each text i we sweep the attack strength ($T \in \{0.5, 0.8, 1.0, 1.2, 1.5\}$ for rephrasing and edit fractions $\{2\%, 5\%, 10\%, 20\%, 40\%\}$ for word edits), and measure the text distortion d as in the main text. Let d_i^c be the smallest distortion over the sweep at which decision c flips ($d_i^c = \infty$ if it never flips), for $c \in \{\mathrm{pub}, \mathrm{net}\}$ where net additionally requires passing the tampering test. We plot $\mathrm{CDF}_c(d) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}[d_i^c \le d]$, the share of all N texts flipped at distortion budget d, with texts that never flip keeping the curve below 1. Results for removal are in Fig. 14, which gives the distortion needed to reach a given success rate. Informed rephrasing and the uninformed baseline curves nearly coincide, so detector access gives the attacker little advantage there. Informed word edits instead reach a public ASR near 100% at distortions where rephrasing achieves almost nothing, but the two-stage test limits their net rate near 40%.

D.7 A Gray-box Public Detector

We now assume a gray-box detector that only returns the public-key p-value of the text. We can directly adapt our white-box attack suite to read the score of a token from two detector queries. The returned p-value is a decreasing function of the accumulated score $\mathbf{S}$ of the text, so the attacker inverts the watermark's null to recover $\mathbf{S}$, and takes $\mathbf{S}_{\text{w/ } v} - \mathbf{S}_{\text{w/o } v}$ as the score of token v. We examine the difference in the score space, not the p-values, because p depends on the number n of scored positions, which differs between the two texts. This estimate is noisy for word edits, since removing v also re-seeds the h positions that follow it: their tokens are unchanged but their contexts are not, so their scores are redrawn and the difference carries that change along with the score of v. On watermarked text the redrawn positions lose the watermark, but for small h over large n the

effect on the total scores is small. This noise even vanishes for rephrasing, where candidates are appended and the preceding context is fixed.

However, exploiting the gray-box detector means submitting numerous queries to it, which can be practically defended with rate-limiting. Rephrasing attacks need to select the current token based on its score, so at every generation step they can conservatively query the detector at the top k probable tokens. Avoiding full regeneration with word edits results in fewer queries, since only a few words are going to be replaced. However, this is more expensive in the case of gray-box, where the attacker needs to probe each token's importance by removing it from the text and comparing the difference in overall p-value.

D.8 Watermark Stealing

So far, our experiments exploit the public detector in an online setting, querying it during tampering. In this section, we examine watermark stealing methods (Jovanović et al., 2024; Chen et al., 2025), which learn an approximate watermark rule from a corpus, then perform offline attacks based on the reverse-engineered watermark. We evaluate how advantageous the public detector can be for

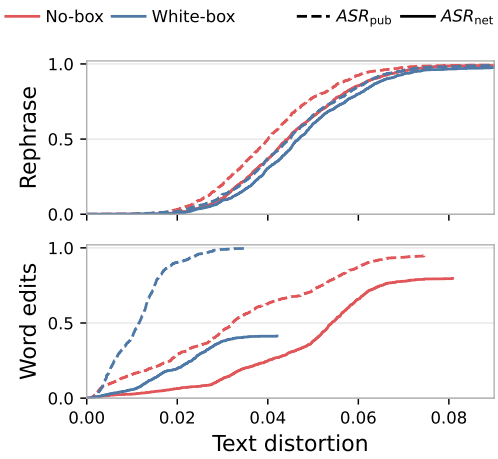


Figure 14: Same sweep as the left panel of Fig. 5, viewed per text: empirical CDF of the smallest flipping distortion for removal, no-box vs. white-box, rephrasing (top) and word edits (bottom). White-box rephrasing barely improves over no-box at equal distortion, while white-box word edits flip the public detector at low distortion but rarely survive the two-stage test.

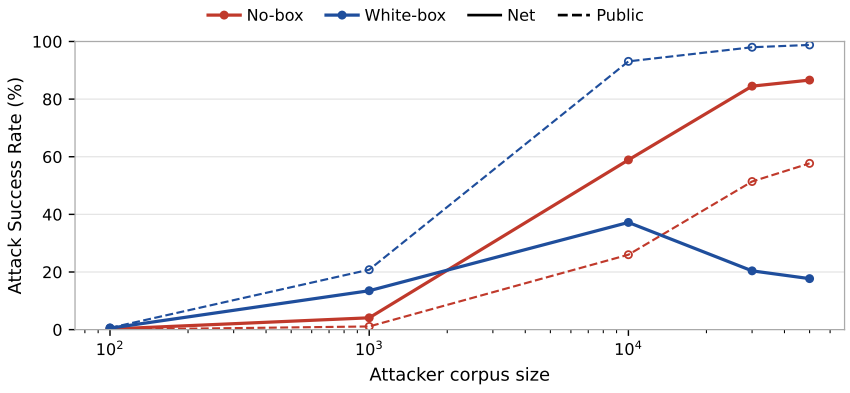


Figure 15: Watermark stealing. Using the public detector increases the public ASR but not $\mathrm{ASR}_{\mathrm{net}}$, which stays no higher than the no-box attack, since supervision reaches only the public channel. The generation parameters are identical to previous experiments, with the exception of using $h = 2$ as a context window, since our experiments with $h = 3$ achieved at most 3% ASR.

Table 12: CaLMQA by language at $p < 10^{-3}$. The median $-\log_{10} p$ columns are measured on unattacked watermarked text, and the attack columns report white-box word edits at 20%, the strongest arm of Sec. 2.3. $\mathrm{ASR}_{\mathrm{pub}}$ is what the attacker sees, ADR the percentage of tampering detections on texts that evade the fused detector, and $\mathrm{ASR}_{\mathrm{net}}$ what evades both stages. We see that both the split watermark construction and the tampering test remain effective in low-resource languages.

			Median $-\log_{10} p \uparrow$			Removal				Forgery			
Language	Script	N	Public	Private	Fused	$\mathrm{ASR}_{\mathrm{pub}}\downarrow$	$\mathrm{ASR}_{\mathrm{fus}}\downarrow$	ADR$\uparrow$	$\mathrm{ASR}_{\mathrm{net}}\downarrow$	$\mathrm{ASR}_{\mathrm{pub}}\downarrow$	$\mathrm{ASR}_{\mathrm{fus}}\downarrow$	ADR$\uparrow$	$\mathrm{ASR}_{\mathrm{net}}\downarrow$
Higher-resource													
Hungarian	Latin	72	17.5	18.0	34.2	98.6%	19.4%	42.9%	11.1%	86.1%	52.8%	47.4%	27.8%
Chinese	Han	72	10.2	10.7	20.5	100%	98.6%	94.4%	5.6%	100%	97.2%	90.0%	9.7%
German	Latin	72	9.9	10.7	19.2	100%	90.3%	84.6%	13.9%	94.4%	77.8%	69.6%	23.6%
Arabic	Arabic	72	9.4	10.2	18.3	100%	87.5%	77.8%	19.4%	91.7%	80.6%	77.6%	18.1%
Hebrew	Hebrew	65	8.5	9.5	17.1	100%	93.8%	68.9%	29.2%	95.4%	84.6%	85.5%	12.3%
Korean	Hangul	72	8.9	9.0	17.0	100%	87.5%	63.5%	31.9%	94.4%	75.0%	64.8%	26.4%
Japanese	Japanese	72	8.1	8.6	15.3	100%	100%	95.8%	4.2%	100%	97.2%	92.9%	6.9%
English	Latin	71	7.3	8.3	15.2	100%	100%	98.6%	1.4%	97.2%	95.8%	92.6%	7.0%
Spanish	Latin	72	7.2	6.7	13.1	100%	100%	90.3%	9.7%	100%	87.5%	90.5%	8.3%
Russian	Cyrillic	72	6.8	5.9	11.6	100%	97.2%	75.7%	23.6%	94.4%	65.3%	66.0%	22.2%
Hindi	Devanagari	72	3.1	3.8	6.1	100%	97.2%	34.3%	63.9%	37.5%	5.6%	25.0%	4.2%
Lower-resource													
Hiligaynon	Latin	48	22.0	21.8	42.4	100%	27.1%	61.5%	10.4%	97.9%	81.2%	74.4%	20.8%
Kirundi	Latin	39	15.1	17.1	35.1	59.0%	33.3%	61.5%	12.8%	71.8%	35.9%	42.9%	20.5%
Tswana	Latin	48	17.3	17.1	34.4	95.8%	43.8%	14.3%	37.5%	77.1%	60.4%	51.7%	29.2%
Pashto	Arabic	56	15.8	15.6	32.4	94.6%	32.1%	38.9%	19.6%	69.6%	50.0%	28.6%	35.7%
Samoan	Latin	18	8.0	8.4	16.6	100%	55.6%	40.0%	33.3%	77.8%	66.7%	58.3%	27.8%
Papiamento	Latin	7	4.4	5.5	7.3	100%	85.7%	83.3%	14.3%	71.4%	57.1%	50.0%	28.6%
Average	–	–	9.1	9.4	17.5	97.8%	77.3%	74.4%	19.8%	87.6%	70.9%	74.9%	17.8%

stealing attacks. We use the purely statistical method of Jovanović et al. (2024) as a baseline in the no-box scenario, which uses the frequency ratio of `(context, token)` pairs between a watermarked and unwatermarked corpus to estimate a per-token context-based bias. We create a white-box adaptation, only changing the frequency estimation by using exactly the score for that pair instead of counting every appearance in the watermarked corpus equally. The rephrasing attacks with the stolen watermark are performed with the same settings as the main experiments.

We present the stealing results in increasing sample budget in Fig. 15, showing that while detector exploitation accelerates the learning, it leads to very low $\mathrm{ASR}_{\mathrm{net}}$. This happens because while no-box stealing extracts information from both the public and the private channels, exact supervision from the public detector only targets the former, causing channel score imbalance. Using uninformed stealing without querying the detector is therefore more advantageous to an attacker who intends to reverse-engineer the watermark. The stealing liability still exists, but is caused merely by the deployment of the watermark itself, not the public detector.

D.9 Other Datasets

We test whether the two-stage pipeline survives a change of language and script on CaLMQA (Arora et al., 2025), a multilingual and culturally-specific question answering dataset. We prompt Qwen2.5-7B with 1k questions from the dataset to generate watermarked responses towards removal, and unwatermarked responses towards forgery (since there are no human-written reference answers). We modify our attacks by using a rephrasing prompt translated in the target language. Because the 1k questions are split across languages, each row examines less than 100 documents, so per-language rates are expected to be noisier than the main experiments. Results are given in Tab. 12.

E Asymmetric Contexts

So far both keys share one h-gram context. The tampering test then uses a single signal, the public/private gap $\mathbf{T} = \mathbf{S}_{\mathrm{pub}} - \mathbf{S}_{\mathrm{priv}}$, which moves only when an attack treats the two keys differently. A no-box paraphrase degrades both channels alike and leaves $\mathbf{T}$ near zero, which is why the uninformed attacker remains the stronger removal threat (Fig. 4).

Unequal contexts are a natural way to close that gap, however, we report them as a negative result. A channel keeps its signal at a token only if the whole $(h{+}1)$-gram seeding its pseudorandom function survives the edit, with probability $q_h = (1-\rho)^{h+1}$ at edit rate ρ, so a wide context on the public key decays faster than a short one on the private key and uninformed editing does move $\mathbf{T}$ (Fig. 16).

Table 13: $\mathrm{ASR}_{\mathrm{net}}$ (%, lower is better for the provider) against the number of tokens scored, with both tests run over all of it, at $p < 10^{-3}$. Bold marks the better configuration per column. The tampering test does flag some of the asymmetric runs, with ADR rising to about 40%, but it recovers less than the fused test gives up.

	$T = 0.5$		$T = 0.8$		$T = 1.2$	
Scored tokens	$(3, 3)$	$(10, 3)$	$(3, 3)$	$(10, 3)$	$(3, 3)$	$(10, 3)$
296	**65.3**	78.8	**75.2**	89.5	**93.8**	96.2
592	**31.7**	55.4	**51.5**	74.6	**83.4**	93.2
1.5k	**2.1**	16.4	**10.3**	37.5	**50.9**	84.2
3.0k	**0.0**	0.8	**0.2**	7.0	**14.8**	56.5
7.4k	**0.0**	0.0	**0.0**	0.0	**0.1**	8.5
14.8k	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0

The contrast becomes visible where equal contexts leave it at zero, and the sign-flip null of Sec. 3 calibrates it unchanged. However, the wide public channel carries half the fused score and rephrasing destroys it, so the fused test gives up more detection than the tampering test recovers, at every amount of text we scored. The asymmetry advantage is merely forensic, the ability to say that a corpus was rephrased. Concurrent work by Feng et al. (2026) similarly pairs a robust with a fragile signal, fixing its tamper threshold at the first percentile of the fragile score on intact watermarked text, where our sign-flip null needs no such corpus.

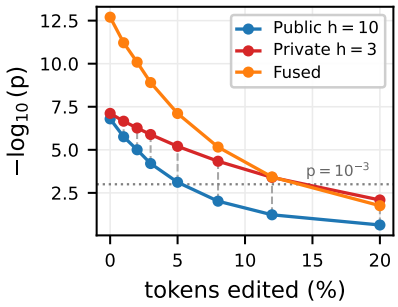


Figure 16: Idealised i.i.d. token edits at $(h_{\mathrm{pub}}, h_{\mathrm{priv}})=(10, 3)$: the wide public context decays faster, so the gap grows with editing. Not enough, though: net success stays worse than at equal contexts.

Placement. The wide context goes on the public key. An edit lowers the public score and also re-seeds the h positions after it, so both effects hit the same channel. Reversing the placement sends the second effect to the private channel, where it cancels part of the first.

Calibration. At detection, a position is kept only if its context is new in every channel, so that all channels are scored on the same positions. On 5992 untampered generations at $(10, 2)$ the pooled z is then -0.001 ± 0.013, and the per-document FPR is 0.0017, 0.011 and 0.053 for nominal 10^{-3}, 0.01 and 0.05.

Uninformed attacker. We apply the no-box rephrasing of Sec. 2.3 to $(3, 3)$ and to $(10, 3)$, 1000 documents per temperature, at matched quality (BERT-F1 0.952 at $T = 0.8$ for both). At 300 scored tokens the asymmetric design is already worse: the public channel is wide, rephrasing destroys it, and it carries half the fused score.

The contrast is a sum, so $|z|$ grows with the square root of the text scored. Over the removed text it reaches -24.3, against -0.6 for the control. At $T = 0.8$ it flags about 1% at 300 scored tokens, 29% at 3×10^3 and 99% at 1.6×10^4, while the control stays at its nominal rate.

That sensitivity does not lower the net rate. Tab. 13 runs both tests over increasing amounts of scored text, showing that equal contexts are better at every size. Rephrasing destroys the wide public channel, so the fused test loses more detection than the contrast recovers, and the asymmetry adds forensic evidence alone.