LLM Watermarking: Sequential Detection

Motivating the Problem!

Watermark:

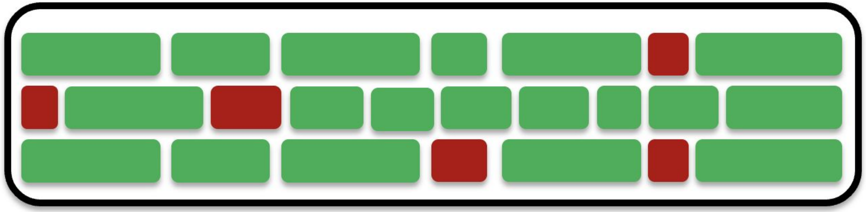
A hidden signal embedded in text generated by a language model (LM) to trace its origin.

Problem Setup:

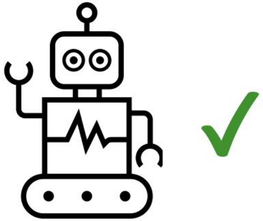
Given an LM ‘m’ and a user prompt ‘q’, embed a watermark in the output ‘y’ such that:

1. **Distortion-Free:** Output quality is unchanged (i.e., $P(x) \approx$ original distribution).
2. **Model Agnostic:** Detection works without access to ‘m’ or ‘q’.
3. **Robustness:** Watermark remains detectable even after adversarial modifications.

Kirchenbauer - Red/Green List Watermarking



Z-score = $\frac{(|s|_G - \gamma T)}{\sqrt{T\gamma(1-\gamma)}}$ = 4



Z-score > τ (say 3)

Detection

Overview of KGW

Solutions:

1. The technique is model/prompt agnostic and does not need the knowledge of model. (Although this a white-box approach)

Problems:

1. Induces distortion into the output by changing probability distribution.
2. The technique is susceptible to substitution attacks.

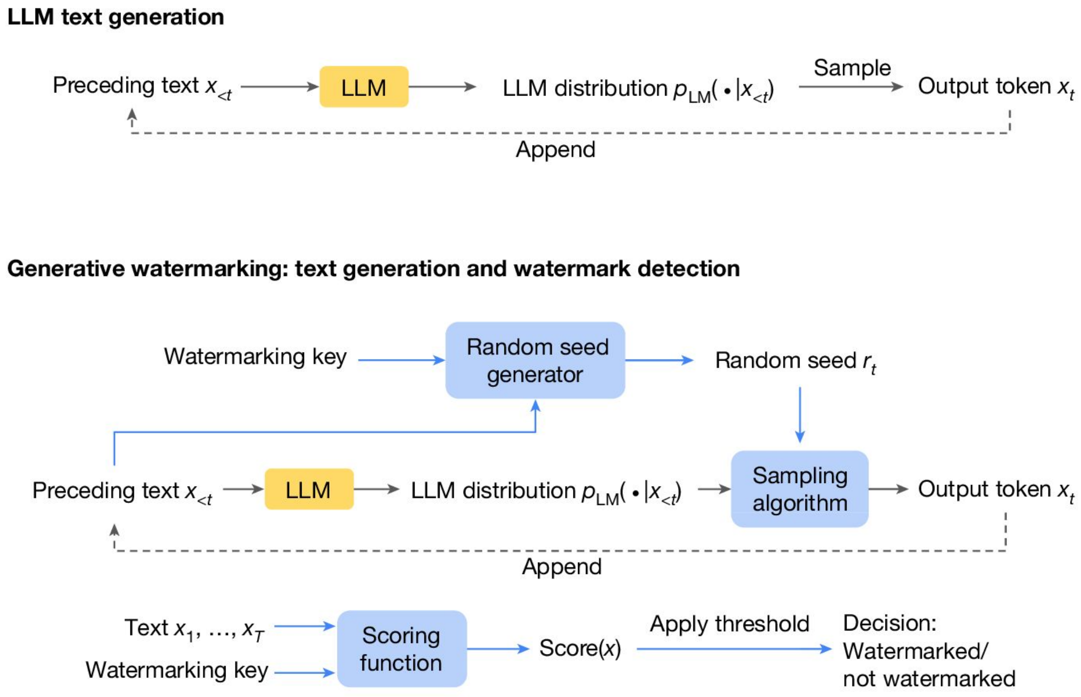
Kuditipudi - Statistical Watermarking

Solutions:

1. Model/Prompt Agnostic
2. Since we do not manipulate the output logits of the LM, we mitigate the distortion.
3. The technique is robust to majority of attacks.

Problems:

1. The algorithm has limitations on detection speeds, and is incapable of 'online' detection due to the design of the algorithm.



Statistical Setup of the Problem

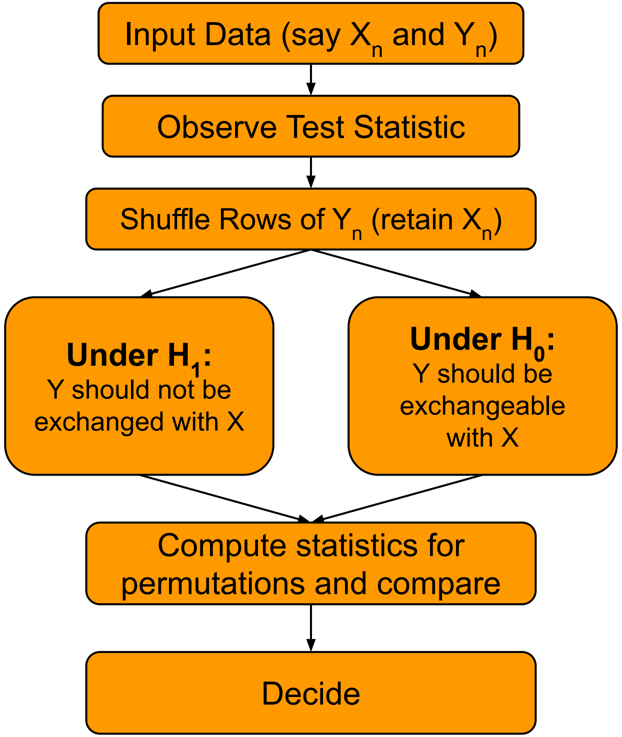
Objective: Detect whether a given sequence of tokens $Y_n = (Y_1, Y_2, \ldots, Y_n)$ was generated by a watermarked language model.

Setup: Given a stream of observations:
Sequence of keys $\xi = (\xi_1, \xi_2, \ldots \xi_n)$,
Output tokens $Y_n = (Y_1, Y_2, \ldots, Y_n)$

Decide between hypotheses:

$$H_0\text{: Text} \perp\!\!\!\perp \text{Key vs. } H_a\text{: Text} \not\perp\!\!\!\perp \text{Key}$$

Goal: For $\alpha \in (0, 1)$, construct a level-α sequential test of power one,
➔ Under H_0: continue forever w.p. $\geq 1 - \alpha$
➔ Under H_1: stop the test, and reject the null as soon as possible



For simplification, we modularise our watermarking scheme into 4 steps/algorithms:

- ➔ **Step-1: Generating the watermark**
- ➔ Step-2: Detecting the presence of watermark in a text
- ➔ Step-3: Test statistic evaluating the misalignment between the keys and the text

Algorithm 1: Watermarked text generation (`generate`)

Input : watermark key sequence $\xi \in \Xi^n$

Params: generation length m, language model p, decoder Γ

Output: string $y \in \mathcal{V}^m$

```
for i ∈ 1, . . . , m do
    y_i ← Γ(ξ_{i%n}, p(· | y_{:i−1}))
return y
```

For simplification, we modularise our watermarking scheme into 4 steps/algorithms:

- ➔ Step-1: Generating the watermark
- ➔ **Step-2: Detecting the presence of watermark in a text**
- ➔ Step-3: Test statistic evaluating the misalignment between the keys and the text

Algorithm 2: Watermarked text detection (`detect`)

Input : string $y \in \mathcal{V}^*$, watermark key sequence $\xi \in \Xi^n$
Params: test statistic ϕ; watermark key sequence distribution $\nu \in \Delta(\Xi^n)$; resample size T
Output: p-value $\widehat{p} \in [0, 1]$

for $t \in 1, \ldots, T$ **do**
  $\xi^{(t)} \sim \nu$
  $\phi_t \leftarrow \phi(y, \xi^{(t)})$
$\widehat{p} \leftarrow \frac{1}{T+1}\left(1 + \sum_{t=1}^{T} \mathbf{1}\{\phi_t \leq \phi(y, \xi)\}\right)$
return $\widehat{p}$

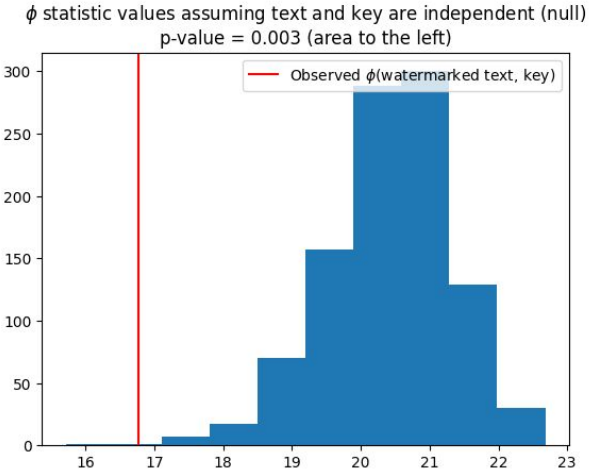


Fig: Distribution of test statistics (under the Null)

For simplification, we modularise our watermarking scheme into 4 steps/algorithms:

- ➔ Step-1: Generating the watermark
- ➔ Step-2: Detecting the presence of watermark in a text
- ➔ **Step-3: Test statistic evaluating the misalignment between the keys and the text**

Algorithm 3: Test statistic (ϕ)

Input : string $y \in \mathcal{V}^*$, watermark key sequence $\xi \in \Xi^n$

Params: alignment cost d, block size k

Output: test statistic value $\phi(y, \xi) \in \mathbb{R}$

for $i \in 1, \ldots, \texttt{len}(y) - k + 1$ **do**

    for $j \in 1, \ldots, n$ **do**

        $y^i \leftarrow \{y_{i+\ell}\}_{\ell=0}^{k-1}$, $\xi^j \leftarrow \{\xi_{(j+\ell)\%n}\}_{\ell=0}^{k-1}$

        $\widehat{d}_{i,j} \leftarrow d(y^i, \xi^j)$

return $\min_{i,j} \widehat{d}_{i,j}$

Proposed Solution

Main idea: Replace Kuditipudi et. al's permutation test with a *sequential* test

Sequential tests allow for gathering evidence against the null hypothesis in an online fashion and stop when it becomes significant. Compared to traditional ("batch") tests:

- ➔ They often reach decisions much earlier (saving resources), and
- ➔ can make the same guarantees on their false positive rates

"Testing by betting" is an increasingly popular framework for designing sequential tests.

Testing by Betting Framework

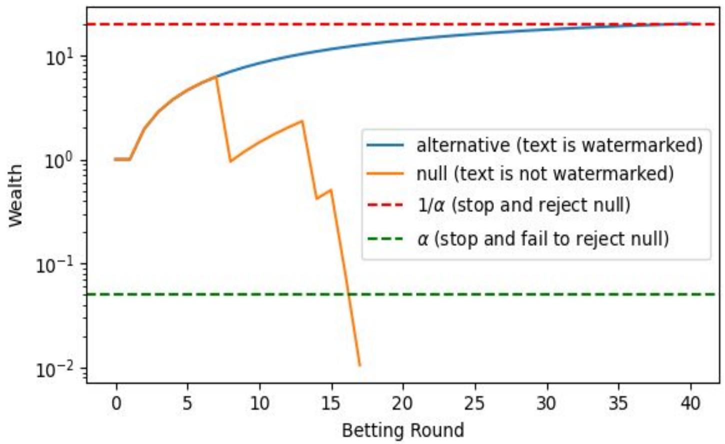
We track the wealth (a martingale by construction) of a gambler that bets against the null.

The betting function is designed such that the wealth (stochastic process):

- ➔ Is a martingale (remains constant in expectation) under the null
- ➔ Grows exponentially under the alternative

In our case,

- ➔ We are testing if the text is independent of the watermarking key
- ➔ Using a Monte Carlo permutation test, which computes T (expensive) test statistics
- ➔ Could before reaching T if we have enough evidence against the null?



➔ p-value = 1/(wealth process)

Intuition and Guarantees of the Strategy

We focus on a log-optimal betting strategy designed specifically for hypothesis testing under a given alternative hypothesis.

Main Properties:

- ➔ **Log-Optimality:** Maximizes the expected log wealth under the considered alternative, ensuring statistically efficient use of evidence.
- ➔ **Finite-Time Guarantee:** Achieves zero resampling risk after a finite number of permutations — no need for infinite resampling to maintain validity.
- ➔ **Any-Time Valid:** The method maintains type-I error control at any stage, enabling real-time, sequential analysis without needing a fixed sample size.

For simplification, we modularise our watermarking scheme into 4 steps/algorithms:

- ➔ Step-1: Generating the watermark
- ➔ Step-2: Detecting the presence of watermark in a text
- ➔ Step-3: Test statistic evaluating the misalignment between the keys and the text
- ➔ **Step-4: Sequentializing the Hypothesis Test using a Sequential-MC Test**

Algorithm 2′: Sequential Monte Carlo permutation test (`seq_mc_permutation_test`)

Input: tokens $y \in \mathcal{V}^*$, watermark key length n, block size k, test statistic function ϕ, watermark key sequence $\xi \in \Xi^n$, threshold α, slack parameter c

Output: p-value estimate $\hat{p} \in [0, 1]$, runtime $t \in \mathbb{N}$

begin
- $W \leftarrow 1$; `// initial wealth`
- $L \leftarrow 0$; `// success count`
- $\phi_0 \leftarrow \phi(y, n, k)$; `// flip sign of observed test statistic`
- **for** $t = 1$ **to** T **do**
  - $\xi^{(t)} \sim \nu$;
  - $\phi_t \leftarrow \phi(y, \xi^{(t)})$;
  - **if** $\phi_t \geq \phi_0$ **then**
    - $L \leftarrow L + 1$; `// increment success count`
  - $W \leftarrow \frac{1-\mathtt{BinomCDF}(L;t+1,c)}{c}$; `// update wealth using binomial tail`
  - **if** $W \geq \frac{1}{\alpha}$ ***or*** $W < \alpha$ **then**
    - **break**; `// early stop`
- $\hat{p} \leftarrow 1/\max(W, \epsilon)$; `// final p-value estimate`
- **return** $\hat{p}, t$

Based on Algorithm 3 from Fischer et al.'s *Sequential Monte-Carlo testing by betting*

Experiment setup

UNIVERSITY OF MICHIGAN

Model and Dataset:

- OPT-1.3B — A 1.3 billion parameter open-source language model developed by Meta.
- C4 dataset— A large-scale English-language dataset curated for language modeling tasks.

Watermark Generation Methods:

- ITS / ITS-edit: Inverse Transformed Sampling for watermarking and its sequential variant.
- EXP / EXP-edit: Exponential Sampling watermarking and its sequential variant.
- KGW-1.0: Kirchenbauer baselines for the sake of comparison.

Evaluation Metrics:

- Permutation p-value: Used to test statistical dependence between generated tokens and watermarking mechanism.
- Number of Permutations: Reflects computational efficiency and convergence behavior of the test.

Oracle Setup:

- We simulate an oracle setting where the watermark detection algorithm has access to the true distribution of the watermark signal under the null hypothesis (i.e., no watermarking).
- This setup enables us to isolate and evaluate the ideal performance of detection methods under best-case assumptions.

Results: Watermarking the Midterm Report

UNIVERSITY OF MICHIGAN

Midterm Report

As large language models (LLMs) continue to improve, traditional watermarking techniques—which previously depended on clear differences between machine-generated and human-written content—are becoming less reliable [9][3]. Alongside advancements in watermarking, more sophisticated detection techniques are also being developed [1][5][6].

Recently, watermarking strategies have explored statistical embedding and detection mechanisms. Notably, Kuditipudi [8] introduced a distortion-free watermarking approach coupled with a dependable detection method; however, its dependency on batch processing limits its practicality for real-time use.

To address this limitation, our project introduces a sequential watermark detection algorithm [2]. We propose an anytime-valid e-process/p-process framework that allows for real-time detection with early stopping, live access to test statistics, and reduced computational requirements.

p-value: ~ 0.95 ⇒ Likely written by a human

Watermarked Report

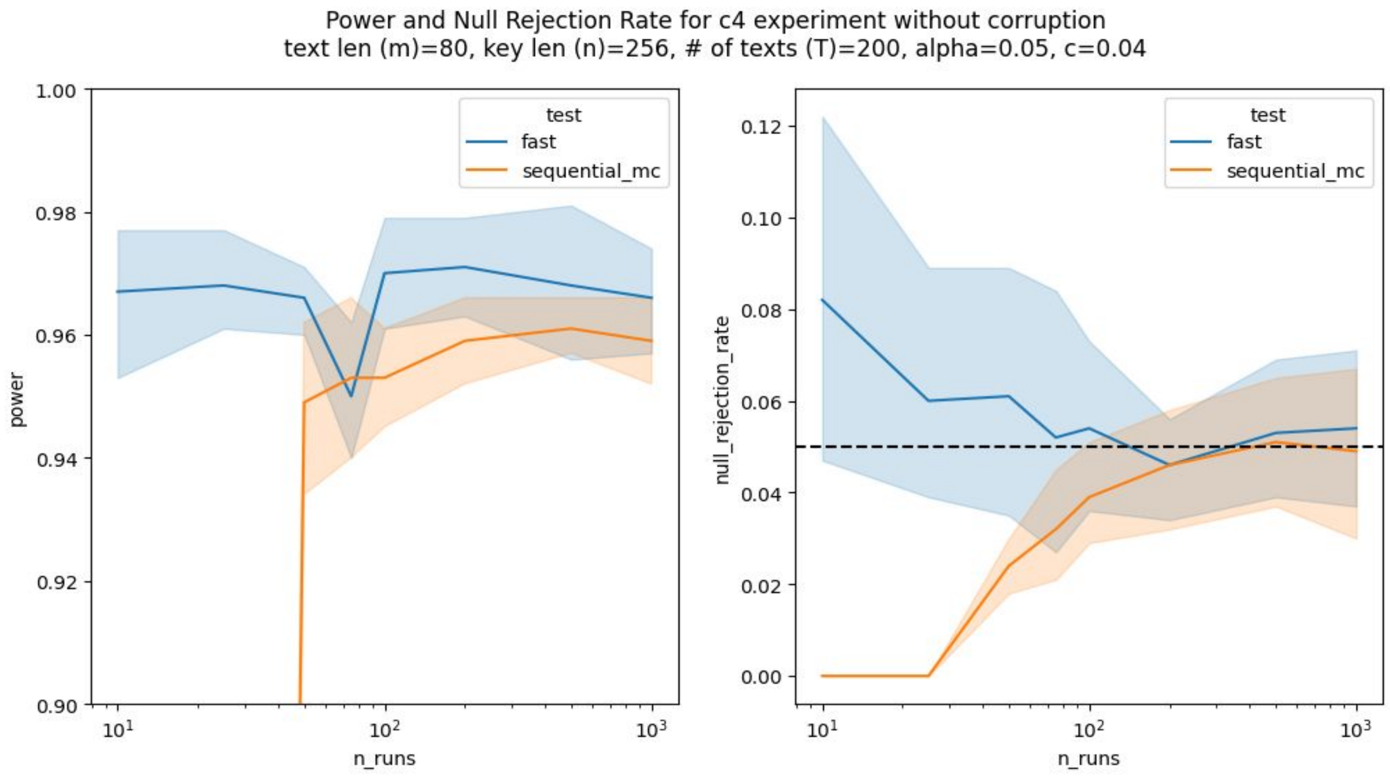
As large language models (LLMs) continue to advance, the effectiveness of traditional watermarking methods—which often rely on observable distinctions between human-authored and machine-generated content—has diminished [9][3]. In parallel with improvements in watermarking strategies, researchers have also made progress in the development of more robust detection methodologies [1][5][6].

Recent approaches in watermarking have focused on statistical techniques for both embedding and detection. One such method, proposed by Kuditipudi [8], offers a distortion-free watermarking strategy alongside a reliable detection mechanism. However, this approach is limited by its reliance on batch processing, which constrains its applicability in real-time environments.

To overcome this limitation, the present work introduces a sequential watermark detection algorithm [2]. Our method leverages an anytime-valid e-process/p-process framework, which facilitates real-time detection through early stopping mechanisms, access to intermediate test statistics, and reduced computational demands.

p-value: 0.0494 ⇒ Likely written by LM
***(LM = OPT-1.3B here)**

Results: Power & Null Rejection Rate



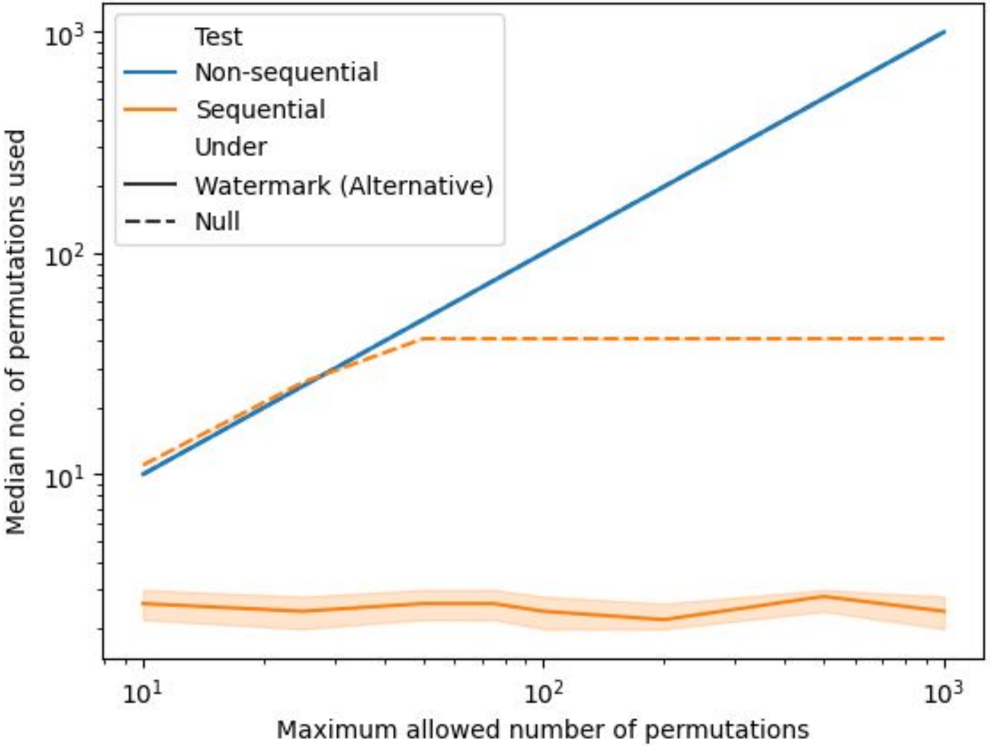
Power = $\mathbf{P}_{\text{Alternate}}$ (Reject Null) = 1 - β

a.k.a Type-2 error rate

Null Rejection Rate = $\mathbf{P}_{\text{Null}}$ (Reject Null) = α

a.k.a Type-1 error rate

Results: Number of Permutations to Decision

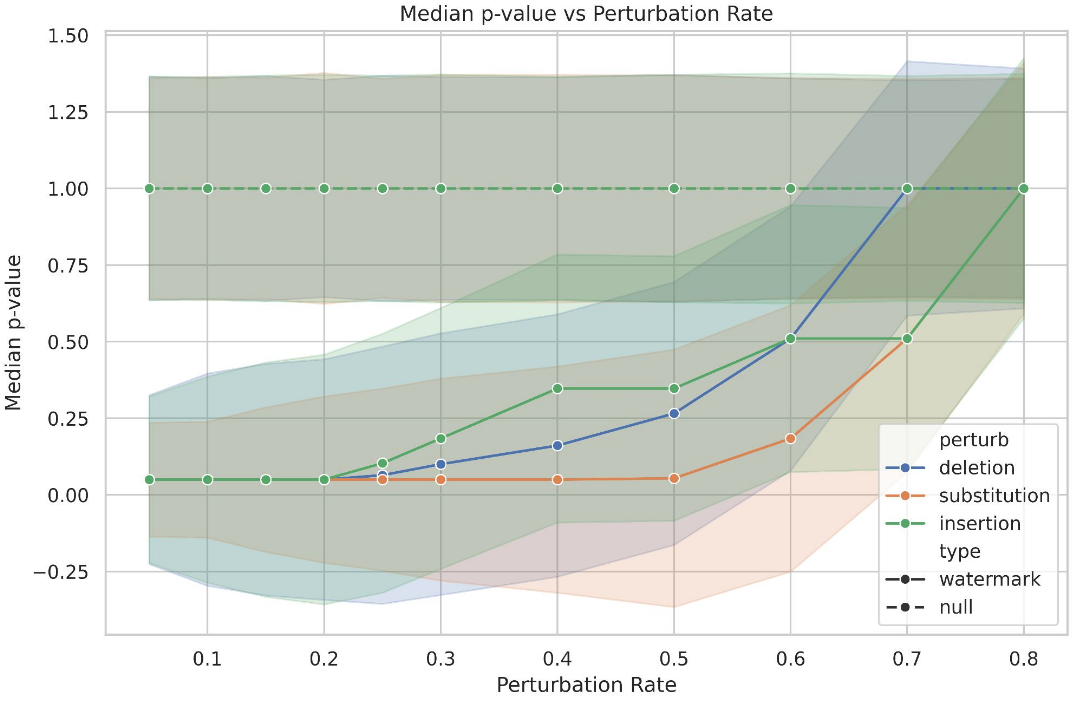


Less Permutations ⇒ Less Time to Decision ⇒ Early Stopping ⇒ Less Computations

Experiment Setup: Testing Robustness

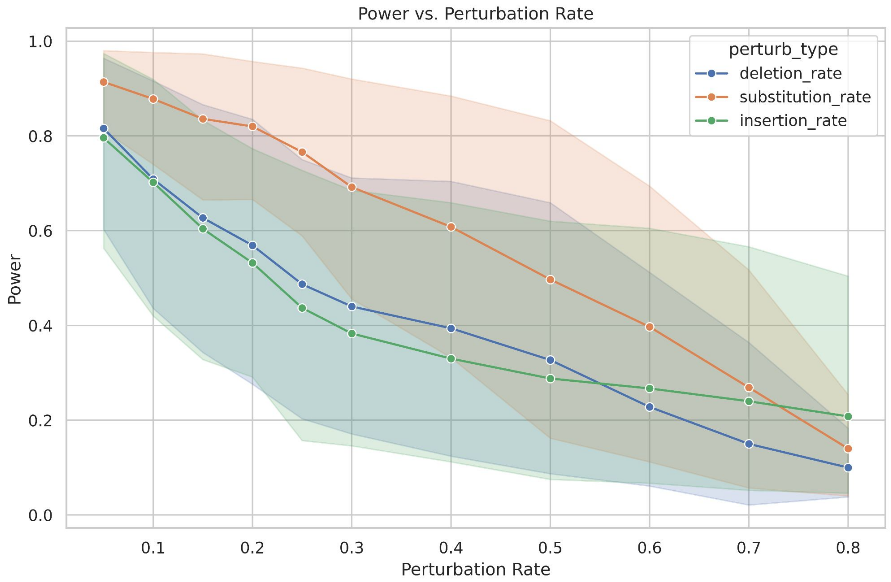
- ➔ Do T = 200 permutations on a text of length m = 80
- ➔ Remove/Insert/Substitute tokens from the generated output at rates ranging from 0.05 to 0.8 randomly.
- ➔ For each corruption rate, report:
  - the Average of Median p-values
  - Null Rejection Rate
  - Empirical Power
  - Median Permutation to Decision across Runs

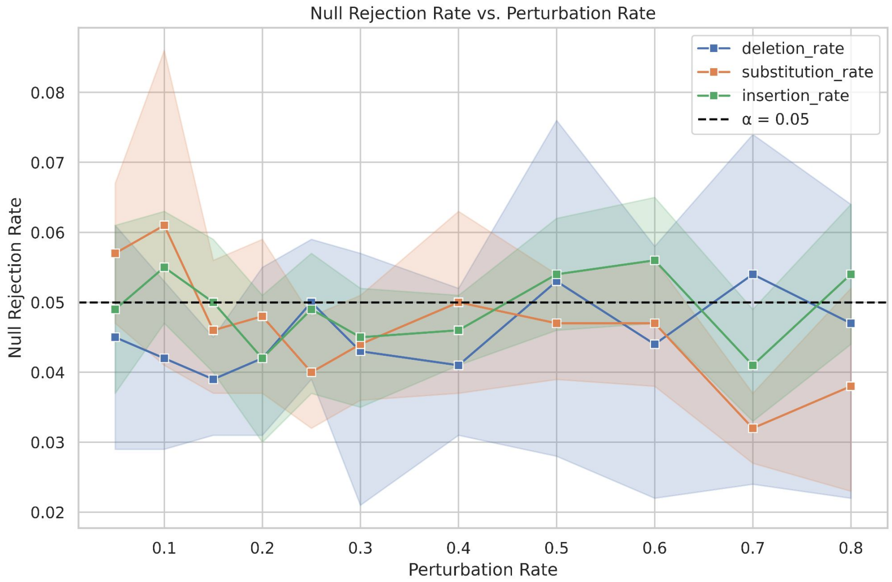
Results: Robustness of Watermark



Median p-values remain low under perturbations ⇒ Watermark is Robust to attacks

Results: Power & Null Rejection Rate under Attack

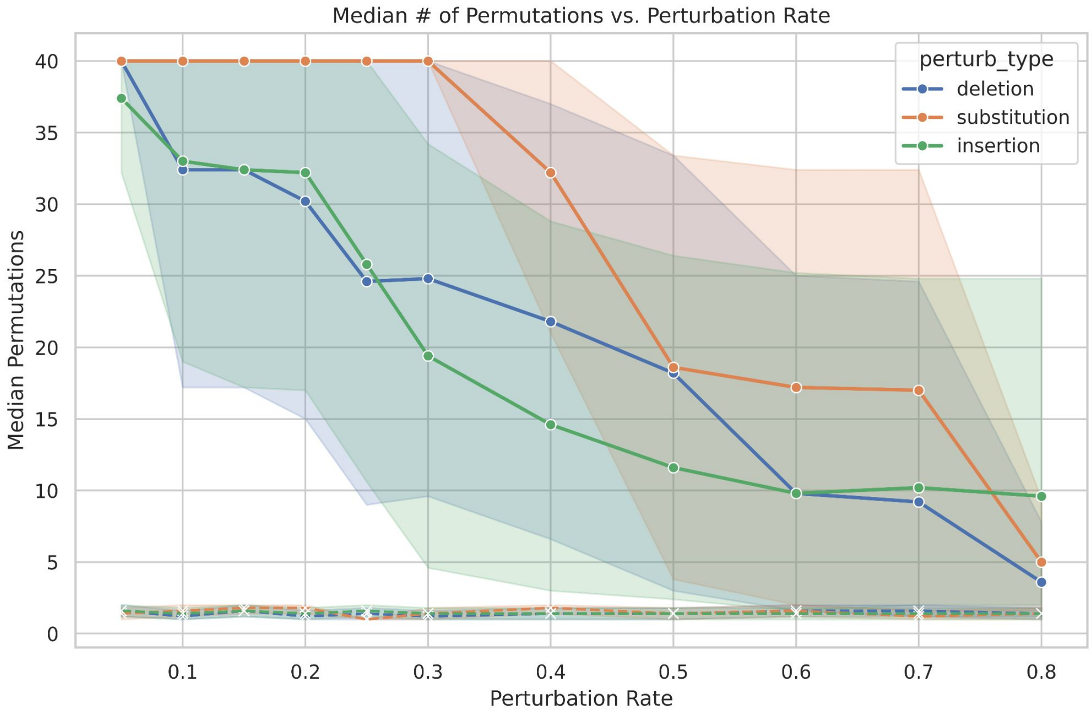




Power close to 1 + Null Rejection Rate below 0.05 ⇒ Test remains valid under attack

***Although the statistical confidence of the decision is impacted**

Results: Number of Permutations to Decision under Attack



Perturbation Increase ⇒ Watermark becomes weak ⇒ Algorithm initiates Early Stopping

***Still taking lesser time than permutations to make a decision**

Analysis

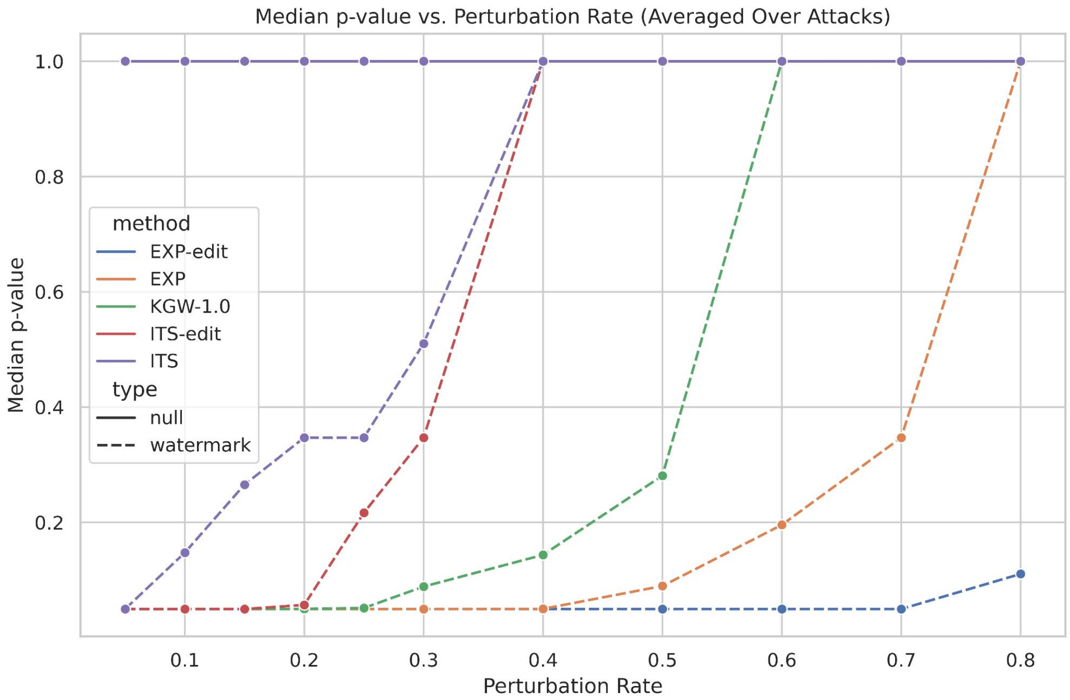
Substitution is most robust to corruption

Null rejection rate stays close to $\alpha = 0.05$, indicating that the false positive rate is well-controlled.

Require far less permutations

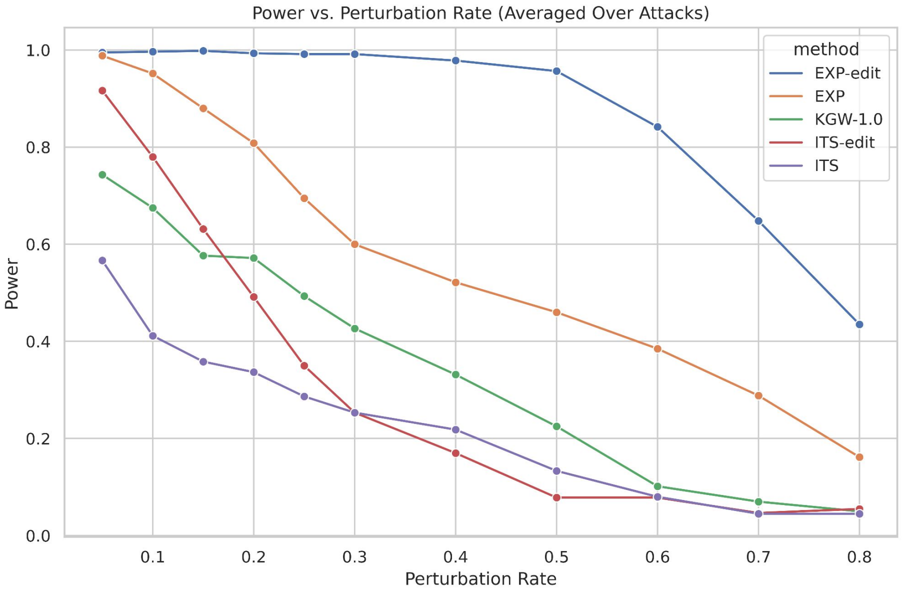
Require less to test null output

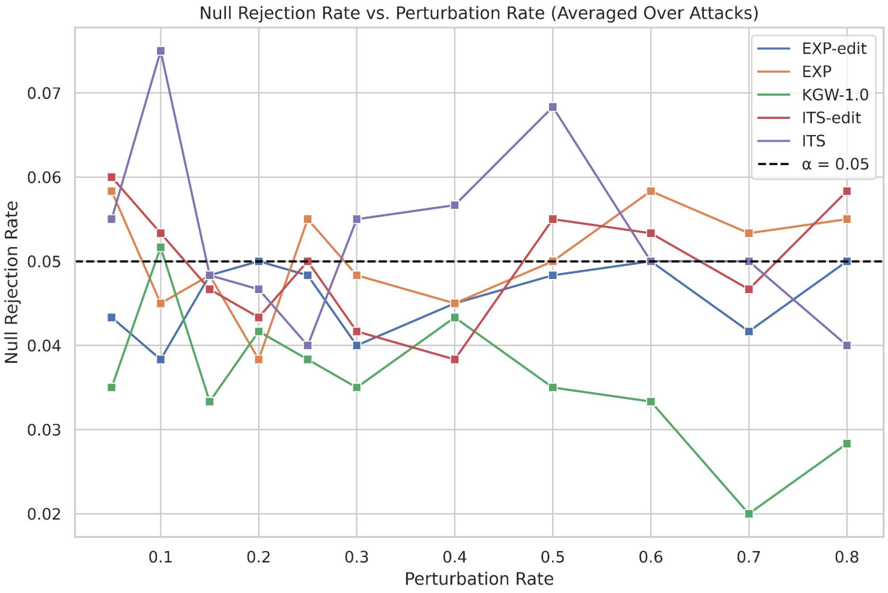
Results: Comparing different generate algorithms



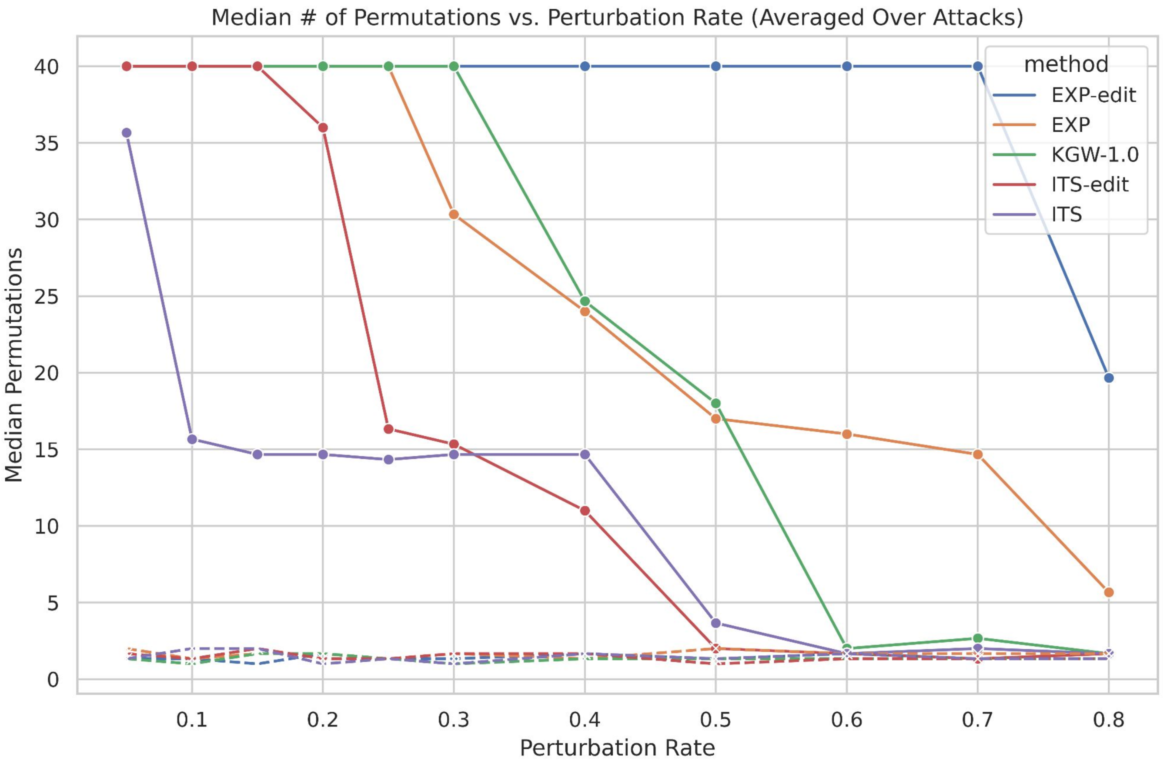
EXP-edit method is most robust to attack outperforming EXP method!

Results: Performance under Attack





Results: Performance under Attack



Conclusion

➔ **Robust Sequential Monte Carlo Test Implemented**: Developed a reliable and scalable SMC-based framework for watermark detection under real-world conditions.

➔ **Outperform the Permutation Tests**: Demonstrates significantly higher efficiency and greater robustness, especially in limited-sample or corrupted data settings.

➔ **High Statistical Power**: Power evaluations show strong and consistent detection capability, confirming the test's reliability across diverse scenarios.

➔ **Robust to Corruptions**: Maintains performance under token-level corruptions such as substitutions and deletions, making it practical for noisy or adversarial text.