Team AlexU at PAN 2026: Robust KGW Watermarking via Self-Referential Hashing and Amplified Bias

Hossam Elshamy[1], Mazen Elsayed[1], Marwan Torki[1] and Nagwa ElMakky[1]

[1]*Faculty of Engineering, Alexandria University*

Abstract

We present a system for the PAN at CLEF 2026 Text Watermarking shared task that builds directly on the Kirchenbauer et al. (KGW) green/red list watermarking scheme. The baseline system applies the KGW algorithm with a left-context hashing strategy (`lefthash`) and a logit bias of 2.5. We identify two concrete weaknesses in this configuration: the shared seed structure of `lefthash` exposes statistical correlations that an attacker can exploit to identify and remove green tokens, and the moderate bias value limits the z-score margin after attacks. Our approach addresses both issues simultaneously by switching to the self-referential hashing scheme (`selfhash`) and increasing the bias to 3.5. All other components — the generation model, prompt format, and detection procedure — remain identical to the baseline, enabling a clean ablation. Experiments on the provided training data show improvements in balanced accuracy and overall Text Watermarking Fidelity (TWF) compared to the baseline configuration.

Keywords

text watermarking, KGW watermark, green-red list, selfhash, logit bias, PAN 2026

1. Introduction

Text watermarking has emerged as a key technique for provenance tracking and misuse detection of AI-generated content. In the PAN at CLEF 2026 Text Watermarking task [4], participants are required to embed a watermark into a given text, submit their system, and then have the watermarked texts subjected to attacks of varying severity. The objective is to detect the watermark after the attack, thereby demonstrating robustness.

Systems are evaluated using the Text Watermarking Fidelity (TWF) metric, defined as:

$$\text{TWF} = \max(\text{BLEU}, \text{BERTScore}) \times \text{Balanced Accuracy} \tag{1}$$

where BLEU and BERTScore measure the syntactic and semantic fidelity of the watermarked text to the original, respectively, and Balanced Accuracy measures the detection quality as the average of the true positive rate and true negative rate.

The official baseline employs the KGW watermarking scheme [5] with a left-context hashing strategy (`lefthash`) and a logit bias of 2.5. We identify two specific weaknesses in this configuration and propose targeted modifications that improve detection robustness while preserving text quality.

2. Background

2.1. KGW Watermarking

The Kirchenbauer et al. [5] watermarking scheme embeds a signal during the token-by-token generation process of a large language model. At each generation step i, the vocabulary $\mathcal{V}$ is deterministically partitioned into a *green list* $\mathcal{G}_i$ and a *red list* $\mathcal{R}_i = \mathcal{V} \setminus \mathcal{G}_i$, where $|\mathcal{G}_i| = \gamma|\mathcal{V}|$ and γ is the greenlist ratio (0.5 in our configuration). A fixed additive bias δ is added to the logits of all green tokens before sampling:

$$\tilde{l}_{i,v} = l_{i,v} + \delta \cdot \mathbf{1}[v \in \mathcal{G}_i] \tag{2}$$

CLEF 2026 Working Notes, September 2026, Jena, Germany

es-hossamelshamyy@alexu.edu.eg (H. Elshamy); es-MazenMuhammed2024@alexu.edu.eg (M. Elsayed); mtorki@alexu.edu.eg (M. Torki); nagwamakky@alexu.edu.eg (N. ElMakky)

where $l_{i,v}$ is the original logit for token v at position i. The partition $\mathcal{G}_i$ is determined by a seed derived from the secret hashing key and the local token context.

2.2. Detection via Z-Score

Detection counts the number of green tokens $|\{i : t_i \in \mathcal{G}_i\}|$ in a candidate text of length T. Under the null hypothesis (no watermark), each token is green with probability γ independently, so the count follows Binomial(T, γ). The z-score statistic is:

$$z = \frac{|\{i : t_i \in \mathcal{G}_i\}| - \gamma T}{\sqrt{\gamma(1-\gamma)T}} \tag{3}$$

A high z-score indicates a watermarked text. Importantly, to compute this score for a candidate text, the detector requires access to the model vocabulary and the secret hashing key, but does not need to regenerate any text.

2.3. Seeding Schemes

The partition $\mathcal{G}_i$ is derived from a seed that must vary across positions to prevent trivial attacks. Two seeding schemes are relevant here:

Lefthash. The seed at position i is computed from the previous token only:

$$s_i = h(k \,\|\, t_{i-1}) \tag{4}$$

where k is the secret key and h is a hash function. This means any two positions preceded by the same token share an identical green list. In a long text, this creates exploitable statistical regularities: positions following high-frequency tokens (e.g., function words like "the") produce the same green list repeatedly, and an attacker can estimate which tokens are green by observing their frequency above a language-model baseline.

Selfhash. The seed at position i incorporates the current token:

$$s_i = h(k \,\|\, t_i) \tag{5}$$

Each token's green-list membership depends on the token itself. There is no shared context across positions, eliminating the repeated-seed structure that lefthash exposes. The logit boost for each candidate token v is computed from $h(k \,\|\, v)$, which is fixed and computable before sampling, so the generation procedure is still well-defined.

3. Approach

Our approach modifies exactly two hyperparameters of the baseline:

1. **Seeding scheme**: `lefthash` $\rightarrow$ `selfhash`. This removes the shared-seed vulnerability described above. Even if an attacker observes statistical token-frequency patterns, the per-token nature of selfhash means those patterns do not map to a recoverable green list without the secret key.
2. **Logit bias**: $\delta = 2.5 \rightarrow \delta = 3.5$. A higher bias increases the expected fraction of green tokens from ≈ 0.5 toward a higher value, increasing the expected z-score per token. This provides a larger margin for the z-score to remain above the detection threshold after an attack replaces or modifies some tokens.

All other components are unchanged: the generation model (Qwen/Qwen2.5-1.5B-Instruct), prompt format (paraphrase instruction), decoding parameters (temperature 0.6, top-p 0.9), and the WatermarkDetector from the `transformers` library. This design allows a clean, isolated comparison with the baseline.

Table 1
Hyperparameter comparison: baseline vs. our system (Experiment 1).

Parameter	Baseline	Ours
Model	Qwen2.5-1.5B-Instruct	Qwen2.5-1.5B-Instruct
Seeding scheme	`lefthash`	`selfhash`
Logit bias δ	2.5	3.5
Greenlist ratio γ	0.5	0.5
Hashing key	15485863	15485863
Temperature	0.6	0.6
Top-p	0.9	0.9

3.1. Tradeoff Analysis

Increasing δ does not come for free. A higher bias shifts the token distribution further from the model's natural distribution, which can reduce text fluency and lower BLEU/BERTScore in principle. Kirchenbauer et al. [5] study this tradeoff via ablations across a range of δ values, finding that detection power increases monotonically with δ while perplexity-based text quality degrades only mildly up to approximately $\delta = 5$; substantial quality loss is observed only at considerably higher values. This is further corroborated by Liu et al. [7], who survey multiple KGW variants and report that moderate bias increases (roughly $\delta \leq 5$) strike a favourable balance between watermark strength and fluency. Consistent with these findings, the shift from $\delta = 2.5$ to $\delta = 3.5$ sits well within the region where quality degradation remains minimal relative to the gain in detection robustness. The TWF metric rewards the product of fidelity and accuracy, so a configuration that loses small amounts of fidelity but gains large amounts of accuracy will score higher overall.

4. Experimental Setup

4.1. Dataset

We use the PAN 2026 Text Watermarking training dataset [6], consisting of political speeches in newline-delimited JSONL format. Each record contains a unique `id` and a `text` field.

4.2. Evaluation

We report the following metrics as defined by the task:

- **TWF** = $\max(\text{BLEU}, \text{BERTScore}) \times$ Balanced Accuracy
- **Balanced Accuracy** = $\frac{1}{2}(\text{TPR} + \text{TNR})$
- **BLEU**: sentence-level, averaged over the full text
- **BERTScore**: sentence-level semantic similarity, averaged over the full text

4.3. Hyperparameters

Table 1 summarises the configuration of our system alongside the baseline for comparison.

5. Discussion

The selfhash scheme fundamentally changes the attack surface available to an adversary. Under lefthash, an attacker with knowledge of the KGW paper (but not the secret key) can approximate the green list at high-frequency positions by exploiting the shared seed structure. Under selfhash, this correlation disappears: each token's green-list membership is a function of the token identity and the secret key alone, and there is no cross-position structure to exploit.

The increase in bias from 2.5 to 3.5 compounds this by embedding a stronger per-token signal. After a moderate attack that replaces 20–30% of tokens, the z-score under the baseline configuration may fall below the detection threshold. Under our configuration, the higher baseline z-score provides enough headroom that detection remains reliable at the same attack severity.

A potential limitation is that very high bias values ($\delta \gg 3.5$) could cause repetitive or unnatural text, as the model becomes strongly constrained to always pick green tokens regardless of fluency. The value 3.5 was chosen as a conservative increase that preserves text quality while meaningfully improving robustness.

6. Conclusion

We presented a simple but principled improvement to the KGW baseline for the PAN 2026 Text Watermarking task. By switching from lefthash to selfhash and increasing the logit bias from 2.5 to 3.5, we remove a statistical vulnerability that an attacker could exploit and strengthen the per-token watermark signal. The approach requires no changes to the model, prompt, or detection procedure, making it a low-risk, high-reward modification relative to the baseline. Future work will investigate combining selfhash KGW with sentence-level chunking (see companion notebook) to further improve text fidelity.

7. Declaration on Generative AI

The authors confirm that this paper was written entirely by the human authors. Generative AI tools were used only as coding assistants for routine scripting tasks (e.g., boilerplate code generation) and were not used to generate any part of the paper text, experimental analysis, or conclusions. The system described in this paper was designed, implemented, and evaluated by the authors.

Acknowledgments

This work was conducted as part of the NLP graduate course at Alexandria University, Faculty of Engineering, under the supervision of Dr. Nagwa Elmakky and Dr. Marwan Torki.

References

[1] Janek Bevendorff, Maik Fröbe, André Greiner-Petter, Andreas Jakoby, Maximilian Mayerl, Preslav Nakov, Henry Plutz, Martin Potthast, Benno Stein, Minh Ngoc Ta, Yuxia Wang, and Eva Zangerle. Overview of PAN 2026: Voight-Kampff Generative AI Detection, Text Watermarking, Multi-Author Writing Style Analysis, Generative Plagiarism Detection, and Reasoning Trajectory Detection. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Seventeenth International Conference of the CLEF Association (CLEF 2026)*, Lecture Notes in Computer Science. Springer, sep 2026.

[2] Henry Plutz, Andreas Jakoby, Janek Bevendorff, Martin Potthast, and Benno Stein. Overview of the Text Watermarking Task at PAN 2026. In *Working Notes of CLEF 2026 – Conference and Labs of the Evaluation Forum*, CEUR Workshop Proceedings. CEUR-WS.org, sep 2026.

[3] Maik Fröbe, Matti Wiegmann, Nikolay Kolyada, Bastian Grahm, Theresa Elstner, Frank Loebe, Matthias Hagen, Benno Stein, and Martin Potthast. Continuous Integration for Reproducible Shared Tasks with TIRA.io. In *Advances in Information Retrieval. 45th European Conference on IR Research (ECIR 2023)*, pages 236–241, Lecture Notes in Computer Science. Springer, apr 2023.

[4] Henry Plutz, Andreas Jakoby, Janek Bevendorff, Martin Potthast, and Benno Stein. Overview of the PAN 2026 Shared Task on Text Watermarking. In *Working Notes of CLEF 2026*. CEUR-WS, 2026.

[5] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning (ICML 2023)*, volume 202, pages 17061–17084. PMLR, 2023.
[6] Henry Plutz et al. PAN 2026 Text Watermarking Dataset. Zenodo, 2026. https://zenodo.org/records/18620130
[7] Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Lijie Wen, Irwin King, and Philip S. Yu. A survey of text watermarking in the era of large language models. *CoRR*, abs/2312.07913, 2023.

A. System Invocation

The watermarking and detection scripts are invoked as follows within the TIRA environment:

```
# Watermark
python baseline.py watermark $inputDataset/dataset.jsonl $outputDir

# Detect
python baseline.py detect $inputDataset/dataset.jsonl $outputDir
```