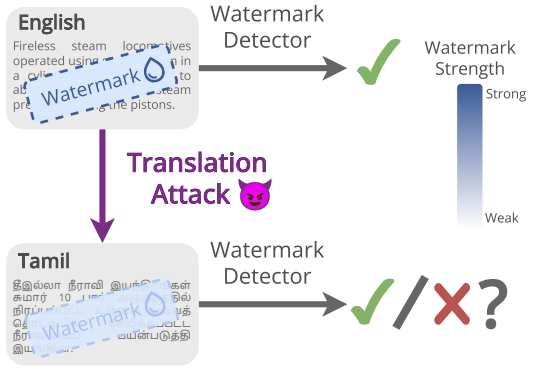
Is Multilingual LLM Watermarking Truly Multilingual? Scaling Robustness to 100+ Languages via Back-Translation

Asim Mohamed[1], **Martin Gubri**[2,3]

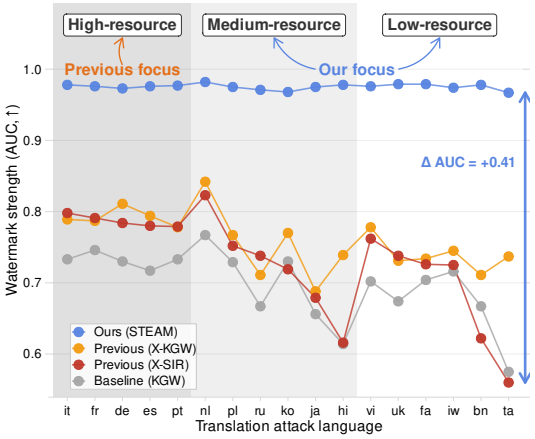
[1]African Institute for Mathematical Sciences, [2]Parameter Lab,
[3]École Polytechnique, Institut Polytechnique de Paris, CNRS
Correspondence: amohamed@aimsammi.org, martin.gubri@parameterlab.de

Abstract

Multilingual watermarking aims to make large language model (LLM) outputs traceable across languages, yet current methods still fall short. Despite claims of cross-lingual robustness, they are evaluated only on high-resource languages. We show that existing multilingual watermarking methods are not truly multilingual: they fail to remain robust under translation attacks in medium- and low-resource languages. We trace this failure to semantic clustering, which fails when the tokenizer vocabulary contains too few full-word tokens for a given language. To address this, we introduce STEAM, a detection method that uses Bayesian optimisation to search among 126 candidate languages for the back-translation that best recovers the watermark strength. It is compatible with any watermarking method, robust across different tokenizers and languages, non-invasive, and easily extendable to new languages. With average gains of +0.23 AUC and +37%p TPR@1%, STEAM provides a scalable approach toward fairer watermarking across the diversity of languages.



(a) Translation attack against LLM watermarking.



(b) Watermarking robustness across languages.

Figure 1: (a) **Our goal** is to evaluate the robustness of LLM watermarks against translation attacks. (b) **Our analysis** reveals that existing multilingual watermarks fail to generalise across languages, while our approach (STEAM) performs consistently better across a wide range of languages overlooked by previous work.

1 Introduction

Recent advances in multilingual watermarking claim to make large language model (LLM) outputs traceable across languages. Yet existing methods have been evaluated only on a small set of high-resource languages, leaving open the question of whether these techniques truly generalise to the world's linguistic diversity. In this work, we show that *current multilingual watermarking methods are not truly multilingual*. Their robustness weakens considerably for medium- and low-resource languages, revealing a major gap in current approaches to content provenance.

The limited robustness of multilingual watermarking has broad consequences. Watermarking was designed to identify LLM-generated text and to reduce the spread of misinformation on social media and synthetic content on the web. An adversary can exploit *translation attacks*, in which a model generates text in one language and the content is translated into another, effectively scrubbing the watermark and reducing its strength (He et al., 2024; Al Ghanim et al., 2025; Han et al., 2025; Luo et al., 2025; Chen et al., 2025). Figure 1a illustrates a translation attack. This threat is not theoretical: large-scale deployed systems such as Google's Syn-

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 23589–23616
October 24-29, 2026 ©2026 Association for Computational Linguistics

thID (Dathathri et al., 2024), used in Gemini, Veo, Imagen, and others, lose detectability after translation (Han et al., 2025). This vulnerability could enable undetectable synthetic content to spread in hundreds of languages, particularly in communities where moderation tools are less effective.

We focus specifically on translation attacks because they are the natural threat in multilingual deployment, where LLM-generated content frequently spreads across language communities via translation. Other attacks, such as paraphrasing or sample-based mixing, operate within the same language and preserve the overall token distribution. Several existing methods already target this regime through semantic invariance (Zhao et al., 2023; Liu et al., 2024a; Ren et al., 2024). Translation attacks are different: they shift text to a different language, causing the token distribution to change substantially, which degrades the watermark signal in ways that semantic invariance cannot address.

Semantic clustering has been proposed as a multilingual extension of watermarking. It groups semantically equivalent tokens (for example, 'house', 'maison', 'casa') into clusters and treats all tokens in a cluster identically regarding the watermark key (for instance, all green or all red). While this approach performs adequately for high-resource languages, we observe that it performs poorly for many others. Tokenizers allocate tokens according to language frequency in their training data, meaning that only high-resource languages contain enough whole-word tokens to be properly represented in semantic clusters. For medium- and low-resource languages, most words are split into subword units not represented in any cluster, which substantially weakens the watermark. These findings suggest that semantic clustering cannot scale effectively beyond high-resource languages.

To address the limited robustness of semantic clustering, we introduce **STEAM** (*Search-based Translation-Enhanced Approach for Multilingual watermarking*), a detection-time method that uses Bayesian optimisation to select the back-translation language that best recovers watermark strength. STEAM is non-invasive, model-agnostic, and compatible with any existing watermarking technique and tokenizer. STEAM supports 126 candidate languages covering high-, medium-, and low-resource settings. Bayesian optimisation caps the number of evaluations at 20 per input, making the search tractable at this scale. The results show large and consistent performance gains over semantic clustering, with *average improvements of +0.23 AUC and +37.4 percentage points (**%p**) in TPR@1%*. We perform an extensive robustness analysis and adaptive adversarial evaluation, all confirming STEAM's stability and effectiveness across diverse attack scenarios while maintaining a low false-positive rate.

Our contributions are:

1. **Extensive multilingual evaluation.** We conduct a large-scale evaluation of multilingual watermarking methods across high-, medium-, and low-resource languages, uncovering weaknesses overlooked in prior work, which has focused exclusively on high-resource languages.

2. **Analysis of the limitations of semantic clustering.** We identify that the limitations of current multilingual watermarking stem from their core reliance on clusters of tokens.

3. **STEAM: a search-based, robust multilingual defence.** We introduce STEAM , which applies Bayesian optimisation to select the back-translation language that best recovers watermark strength. STEAM already supports 126 candidate languages and is retroactively extensible to new ones, compatible with any watermarking technique and tokenizer, and non-invasive to the model output.

4. **Robustness across diverse languages.** STEAM consistently outperforms existing multilingual watermarking methods, with improvements of up to **+0.41 AUC** and **+58.8%p TPR@1%**.

2 Related Work

LLM watermarking is generally classified by when the watermark is applied: training-time or inference-time (also known as logit-based) (Liu et al., 2024b). We focus on the latter.

Logit-based watermarking. Logit-based watermarking embeds a watermark by directly modifying the token probability distribution (logits) during text generation (Liu et al., 2024b). The seminal approach, KGW (Kirchenbauer et al., 2023), partitions the tokenizer vocabulary into green and red lists using a random seed derived from a fixed window of previous tokens and biases generation towards green tokens. Zhao et al. (2023) proposed Unigram Watermarking, an extension of KGW that

employs a fixed green/red partition to improve robustness against text editing and paraphrasing attacks. To maintain text quality, Hu et al. (2023) introduced an unbiased watermarking approach that integrates watermarks without altering the overall probability distribution of the output. Several works (Lee et al., 2024; Lu et al., 2024a; Liu and Bu, 2024; Wu et al., 2024) further improve robustness while preserving text quality.

Beyond these, ITS and EXP (Kuditipudi et al., 2024) offer model-agnostic, distortion-free watermarking schemes that remain robust to text manipulation attacks. Our work analyses the multilingual capabilities of these techniques and builds upon them to develop our defence, STEAM.

Watermarking robustness. Several studies, including SIR (Liu et al., 2024a), SemaMark (Ren et al., 2024), semantic-aware watermarking (Fu et al., 2024), and SempStamp (Hou et al., 2024), incorporate semantic information to improve the robustness of watermarks against text transformation attacks. To achieve a balanced and context-aware partitioning of the green and red token lists, Guo et al. (2024) leveraged locality-sensitive hashing (LSH) (Indyk and Motwani, 1998) to generate a semantic key from contextual embeddings. Inspired by the inherent redundancy of multimedia data, WatME (Chen et al., 2024) embeds mutual exclusion rules within the lexical space for text watermarking. Furthermore, Luo et al. (2025) identified watermark collision, where multiple watermarks interact in ways that distort statistical distributions and hinder detection. These approaches primarily target attacks within a single language, such as paraphrasing or text editing, where the overall token distribution remains broadly stable. They do not address translation attacks, which shift text to a different language and cause a substantially different token distribution. This is a distinct failure mode that requires a different defence.

Multilingual watermarking. While much of the initial research focused on monolingual English text, a growing body of work now addresses the unique challenges of cross-lingual watermarking. A foundational contribution in this area is X-SIR (He et al., 2024), a direct extension of the SIR framework designed to defend against translation attacks. Other works have focused on evaluating the cross-lingual robustness of existing methods. For example, Han et al. (2025) assessed the robustness of SynthID-Text (Dathathri et al., 2024)

Criterion	KGW & SIR	X-KGW & X-SIR	STEAM (ours)
Multilingual support			
Multilingual	✗	✓	✓
Multilingual robustness			
Medium-resource	✗	~	✓
Low-resource	✗	✗	✓†
Tokenizer-robust	–	✗	✓
Multilingual integration			
Non-invasive	–	✗	✓
Watermark-agnostic	–	✗	✓
Retroactive support	–	✗	✓

Table 1: Comparison of watermarking methods and their multilingual capabilities. Criteria defined in App. A.6.
✓ = Yes, ✗ = No, ~ = Limited, – = Not applicable
† Requires translator (low-quality translation sufficient)

to meaning-preserving transformations like back-translation. Similarly, Al Ghanim et al. (2025) conduct a comparative evaluation of four watermarking methods: KGW, Unigram, EXP, and X-SIR. Their analysis assesses robustness and text quality under various parameters and removal attacks in cross-lingual settings. Although these studies provide valuable insights, their scope is often limited to high-resource languages. Our work addresses this gap by providing a more comprehensive cross-lingual evaluation that includes an extensive set of low- and medium-resource languages.

3 Experimental Setup

Dataset. We base our evaluation on the English subset of the mC4 dataset (Raffel et al., 2023), following the setup introduced by He et al. (2024). We sample a test set of 500 texts for all experiments.

Attacks. We evaluate watermark robustness under two translation-based attacks. Unless specified otherwise, all translations are performed with Google Translate. The first, direct translation, converts English outputs into a target language and is used in the main experiments. The second applies multi-step translation through a pivot language (He et al., 2024) and is reported in Appendix B.

Multilingual models. We use the following multilingual language models: `Aya-23-8B` (Aryabumi et al., 2024), `LLaMA-3.2-1B` (Grattafiori et al., 2024), and `LLaMAX-8B` (Lu et al., 2024b).

Watermarking methods. We analyse three watermarking schemes. We use the standard KGW

(Kirchenbauer et al., 2023) as our primary non-multilingual baseline. Second, we evaluate X-SIR (He et al., 2024), a foundational work that proposes semantic clustering for cross-lingual robustness. Finally, we introduce X-KGW, a method that applies semantic clustering to KGW. This setup allows us to isolate and measure the precise impact of semantic clustering on watermark robustness (see Appendix A.4 for details about X-KGW).

Evaluation metrics. We assess the strength of the watermark using two standard binary classification metrics, the Area Under the ROC Curve (*AUC*) and the True Positive Rate at a false positive rate fixed at 1% (*TPR@1%*).

4 Semantic Clustering Fails in Diverse Multilingual Settings

We show that semantic clustering is not inherently multilingual: it lacks robustness in unsupported languages, and extending its coverage to more languages fails in medium- and low-resource settings. We then identify the structural cause of these failures.

4.1 Robustness Against Unsupported Languages

X-SIR is not robust when evaluated beyond its explicitly supported languages. The original evaluation covers only five supported languages (He et al., 2024), leaving robustness on unsupported ones unknown. Evaluating on ten unsupported languages (Italian, Spanish, Portuguese, Polish, Dutch, Croatian, Czech, Danish, Korean, and Arabic) shows that X-SIR is not robust in this setting (Appendix B.2). X-SIR achieves average AUC and TPR@1% of only 0.75 and 0.14 on Aya-23 8B, and 0.68 and 0.07 on LLaMA-3.2 1B. Some languages are more vulnerable than others: Arabic is the weakest for Aya-23 8B (AUC of 0.687, TPR@1% of 0.093), while Portuguese and Arabic are the weakest for LLaMA-3.2 1B (AUC of 0.650, TPR@1% of 0.055). Even a single poorly supported language is sufficient for an attacker to bypass detection. A hold-one-out experiment (Appendix B.1) confirms that the originally supported high-resource languages remain robust when one is withheld.

4.2 Failure to Support a Broad Range of Languages

Explicitly extending semantic clustering to more languages does not resolve its robustness failures.

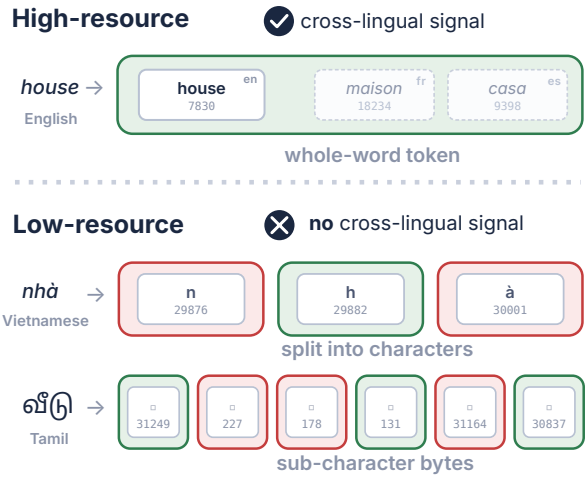


Figure 2: **Limitations of semantic clustering**. In high-resource languages (top), "house" and its translations ("maison", "casa") are tokenised as whole words and join a single green-token cluster. In low-resource languages (bottom), "house" splits into character-level tokens (Vietnamese) or UTF-8 byte tokens (Tamil). Without a shared cluster, tokens carry no cross-lingual watermark signal (green & red). LLaMA 3 tokeniser.

We extend the support of X-SIR and X-KGW to 17 languages spanning high-, medium-, and low-resource settings (Appendix A.5)[1]: French, German, Italian, Spanish, and Portuguese (high-resource); Polish, Dutch, Russian, Hindi, Korean, and Japanese (medium-resource); and Bengali, Persian, Vietnamese, Hebrew, Ukrainian, and Tamil (low-resource). Table 2 reports results for Aya-23 8B. Robustness decreases with the level of resource: X-SIR's average AUC drops from 0.786 (high) to 0.721 (medium) to 0.689 (low), and X-KGW follows the same trend (0.792, 0.753, 0.739). The degradation is most severe for Tamil, where X-SIR reaches an AUC of only 0.560 and TPR@1% of 0.049 on Aya-23 8B (0.561 on LLaMAX-3). Even with explicit support, semantic clustering fails to maintain reliable detection across all languages.

4.3 On the Fundamental Limitations of Semantic Clustering in Multilingual Watermarking

The failures of X-SIR and X-KGW in mid- and low-resource languages stem from a structural property of semantic clustering: the method relies on full-word tokens, whose coverage in tokeniser vocabularies varies sharply across languages.

[1]Our categorisation relies on the MUSE dictionary structure (Conneau et al., 2017): high-resource languages possess extensive non-English-centric bidirectional mappings; medium- and low-resource languages are English-centric only, distinguished by dictionary size.

Translation Attack		X-SIR (↑)		X-KGW (↑)	
Type	Lang.	AUC	TPR@1%	AUC	TPR@1%
High-resource	fr	0.791	0.149	0.787	0.280
	de	0.784	0.163	0.811	0.312
	it	0.798	0.152	0.789	0.354
	es	0.780	0.150	0.794	0.278
	pt	0.779	0.176	0.778	0.330
Medium-resource	pl	0.752	0.146	0.767	0.312
	nl	0.823	0.213	0.842	0.332
	ru	0.738	0.122	0.711	0.246
	hi	0.616	0.056	0.739	0.194
	ko	0.719	0.115	0.770	0.318
	ja	0.679	0.103	0.688	0.160
Low-resource	bn	0.622	0.055	0.711	0.180
	fa	0.726	0.131	0.734	0.242
	vi	0.762	0.157	0.778	0.308
	iw	0.725	0.115	0.745	0.220
	uk	0.738	0.148	0.731	0.222
	ta	0.560	0.049	0.737	0.172
Minimum		0.560 (ta)	0.049 (ta)	0.688 (ja)	0.160 (ja)

Table 2: **Semantic clustering robustness tends to decrease from high- to low-resource languages, even with explicit support.** Aya-23 8B generates text in English; the translation attack targets each of 17 supported languages. Minimum is the worst-case robustness (best attack language). Other models in App. B.3.

Semantic clustering assigns watermark signals using multilingual dictionaries to group semantically equivalent words across languages (He et al., 2024). However, the share of dictionary words that appear as whole-word tokens in tokeniser vocabularies varies sharply across languages (Appendix B.4). Low-resource languages have very few whole-word tokens, as low as 0.13% for Hebrew. BPE-based tokenisers allocate tokens by frequency in training data, which favours high-resource languages and fragments others into subword units with limited to no semantic meaning (Figure 2).

Figure 3 plots AUC against the number of whole-word tokens in the tokeniser vocabulary, for each language and model. A positive correlation emerges across all three models: languages with higher tokeniser coverage achieve stronger robustness, while those with lower coverage are far more vulnerable. This exposes three structural limitations. *(i)* When a language has no whole-word tokens, X-KGW collapses to KGW, as no clusters can be formed (Figure 2). *(ii)* Even multilingual tokenisers cannot resolve this, since BPE allocation inherently disadvantages underrepresented languages. *(iii)* The vulnerability extends to monolingual watermarking: text generated

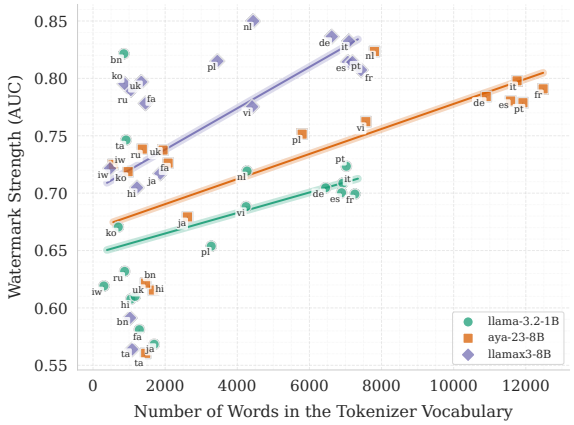


Figure 3: **Languages with larger tokenizer vocabularies have higher watermark robustness.** Average AUC per language and model across three seeds. Lines are least squared regressions.

in a high-resource language and translated into a low-resource one may lose its watermark entirely. These shortcomings are structural and cannot be overcome by expanding language support or retraining tokenisers.

5 STEAM: Scaling Defence to 100+ Languages via Back-Translation

To address the limitations of semantic clustering for multilingual watermarking, we propose **Search-based Translation-Enhanced Approach for Multilingual watermarking (STEAM)** , a novel, model-agnostic defence method that uses Bayesian optimisation to search for the back-translation language that best recovers watermark strength.

5.1 STEAM Description

STEAM recovers a watermark strength degraded by translation attacks via multilingual back-translation. For each suspect text, STEAM searches for the candidate language that, when used to back-translate the text, best recovers the watermark strength. Each candidate is evaluated using a watermark detector with a language-specific null hypothesis (described below). Finally, the highest corrected z-score across all evaluated candidates serves as the final test statistic. Figure 4 provides an overview of the STEAM pipeline.

Bayesian optimisation for back-translation language selection. We use Bayesian Optimisation (BO) to search for the back-translation language that best recovers watermark strength. Each language is characterised by a 131-dimensional feature vector with syntactic and phonological prop-

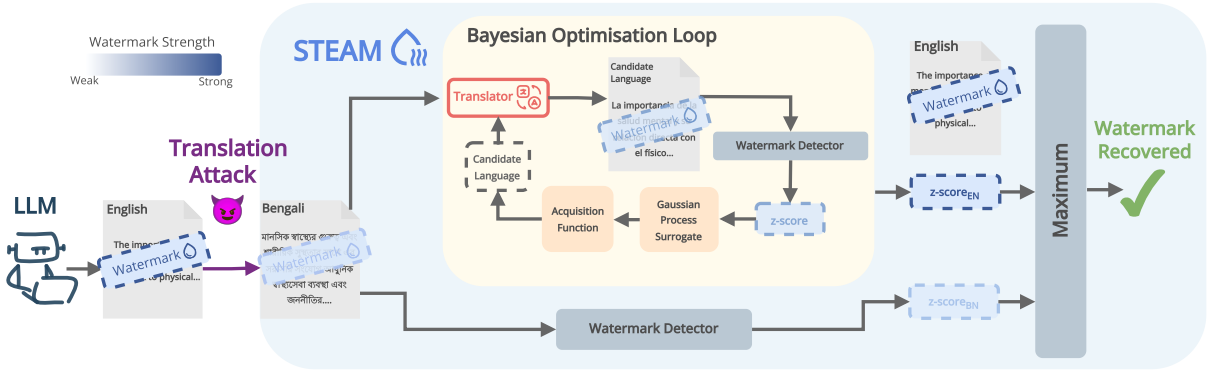


Figure 4: **Overview of STEAM** . A suspect text (Bengali) is back-translated into candidate languages selected by Bayesian optimisation from a pool of 126 languages. Each language is represented by continuous syntactic and phonological features, which guide the search. A standard watermark detector computes a z-score for each candidate text. STEAM selects the back-translation language that yields the highest z-score, here English, the language of the watermarked text before the translation attack.

erties sourced from URIEL (Khan et al., 2025), a knowledge base of linguistic properties. BO first evaluates a small set of randomly selected candidates, then fits a Gaussian process surrogate that models the relationship between linguistic features and observed z-scores, allowing it to predict which unevaluated languages are likely to yield high z-scores based on their similarity to already evaluated ones. At each subsequent iteration, the next back-translation language is chosen by maximising the expected improvement over all unevaluated candidates. The process repeats until a fixed budget of 20 evaluations is exhausted, enabling STEAM to scale to 126 candidate languages at low cost.

Language-specific null hypothesis. The standard z-score formula (Kirchenbauer et al., 2023) tests against a null hypothesis of a fixed green token fraction γ (e.g. $\gamma = 50\%$), assumed uniform across all languages. This assumption is violated in low-resource languages, where tokenizers fragment single UTF-8 characters into sub-character tokens that are reused across many characters, making them disproportionately frequent. For instance, a single token accounts for 21.5% of all tokens in our Tamil texts under Llama 3 tokenizer (see Appendix B.5). If such a frequent token falls in the green list, the green token fraction exceeds γ and the z-score inflates; when it falls in the red list, the z-score deflates. In both cases, the shift is independent of any watermark strength. Since STEAM optimises for the highest z-score across candidate languages, such bias would cause it to systematically favour certain languages regardless of the watermark presence.

We address language token bias by replacing γ with a language- and key-specific γ_ℓ. Concretely, γ_ℓ is the empirical green token fraction measured on a calibration set of 500 human-written texts per candidate language, sourced from mC4 (Raffel et al., 2023), for a fixed watermark key:

$$z_\ell = (n_g - \gamma_\ell n)/\sqrt{n\gamma_\ell(1-\gamma_\ell)} \qquad (1)$$

where n_g is the number of green tokens of the suspect text and n its total number of tokens. Ablating this correction causes only a modest AUC drop of 0.049, but correct language identification falls from 83.5% to 38.6% (App. C.9), confirming that normalisation is critical for unbiased detection.

5.2 STEAM Evaluation

Comparison to semantic clustering. We evaluate STEAM against semantic clustering methods to assess its robustness under translation attacks. As in §4.2, all methods are tested on the same set of 17 attack languages. For back-translation, STEAM chooses from a pool of 126 candidate languages covering a wide range of language families (Appendix A.3). STEAM achieves consistently strong results, maintaining an average AUC above **0.965** across all language categories, including medium- and low-resource ones (Table 3, Appendix C). Compared with semantic clustering approaches (X-SIR and X-KGW), STEAM shows large gains: on average, +0.25 AUC and +44.0%p TPR@1% relative to X-SIR, and +0.216 AUC and +30.7%p TPR@1% relative to X-KGW. The largest improvements are observed for Tamil and Hindi, with up to +0.41 AUC and +58.8%p TPR@1%, respectively. These gains confirm that STEAM generalises reliably beyond high-resource settings.

Source Language		AUC (↑)				TPR@1% (↑)			
Type	Language	KGW	X-KGW	X-SIR	STEAM	KGW	X-KGW	X-SIR	STEAM
High-resource	fr	0.746	0.787	0.791	**0.976**	0.224	0.280	0.149	**0.582**
	de	0.730	0.811	0.784	**0.973**	0.224	0.312	0.163	**0.622**
	it	0.733	0.789	0.798	**0.978**	0.202	0.354	0.152	**0.530**
	es	0.717	0.794	0.780	**0.976**	0.232	0.278	0.150	**0.580**
	pt	0.733	0.778	0.779	**0.977**	0.246	0.330	0.176	**0.538**
Medium-resource	pl	0.729	0.767	0.752	**0.975**	0.228	0.312	0.146	**0.526**
	nl	0.767	0.842	0.823	**0.982**	0.290	0.332	0.213	**0.612**
	ru	0.667	0.711	0.738	**0.971**	0.158	0.246	0.122	**0.576**
	hi	0.614	0.739	0.616	**0.978**	0.120	0.194	0.056	**0.644**
	ko	0.730	0.770	0.719	**0.968**	0.210	0.318	0.115	**0.460**
	ja	0.656	0.688	0.679	**0.975**	0.114	0.160	0.103	**0.498**
Low-resource	bn	0.667	0.711	0.622	**0.978**	0.068	0.180	0.055	**0.604**
	fa	0.704	0.734	0.726	**0.979**	0.196	0.242	0.131	**0.664**
	vi	0.702	0.778	0.762	**0.976**	0.186	0.308	0.157	**0.626**
	iw	0.716	0.745	0.725	**0.974**	0.172	0.220	0.115	**0.578**
	uk	0.674	0.731	0.738	**0.979**	0.210	0.222	0.148	**0.542**
	ta	0.575	0.737	0.560	**0.967**	0.082	0.172	0.049	**0.504**

Table 3: **STEAM consistently and substantially outperforms semantic clustering.** Watermark strength (AUC and TPR@1%) of multilingual watermarking techniques with 17 supported languages and Aya-23 8B. Red indicates robustness lower than the KGW baseline. Bolded is best. Other models and all supported languages in Appendix C.

STEAM robustness to unsupported languages. We next evaluate whether STEAM remains robust when the best language is not in its pool of candidate languages. To ensure a fair comparison, we remove the language of the watermarked text (before the translation attack) from all methods: for semantic clustering, we exclude the corresponding dictionary; for STEAM, we remove the language from the back-translation pool. In this setup, STEAM maintains strong detection across all evaluated languages, with an average AUC of 0.967 (Table 4). It consistently and substantially outperforms semantic clustering, with average gains of +0.19 AUC and +20.6%p TPR@1% over X-KGW and +0.22 AUC and +31.0%p TPR@1% over X-SIR. Unlike semantic clustering methods, which lose coverage when the language is absent from their dictionaries, STEAM remains resilient: linguistically similar languages in its back-translation pool are sufficient to recover the watermark strength.

Robustness across 126 languages. We extend the evaluation to 126 attack languages, covering a broad range of language families and resource levels. Figure 5 plots the AUC of STEAM against KGW for each language. STEAM achieves an average AUC of 0.976, and only two languages fall below 0.965. It improves detection over KGW on every attack language, confirming consistent robustness at scale.

New Language	AUC (↑)			
	KGW	X-KGW	X-SIR	STEAM
it	0.733	0.772	0.796	**0.966**
es	0.717	0.807	0.754	**0.967**
pt	0.732	0.792	0.775	**0.971**
pl	0.730	0.762	0.749	**0.960**
nl	0.768	0.808	0.776	**0.966**
hr	0.706	0.757	0.726	**0.965**
cs	0.717	0.754	0.773	**0.974**
da	0.713	0.764	0.734	**0.971**
ko	0.732	0.754	0.729	**0.961**
ar	0.689	0.765	0.687	**0.971**

Table 4: **STEAM remains robust on unsupported languages and consistently outperforms multilingual methods.** Bolded is best. Red indicates lower than the KGW baseline. Full table in Appendix C.

Robustness to the generation language. STEAM remains robust when the watermarked text is not generated in English. Our other experiments watermark English text and attack it by translation out of English. We invert this setting, generating the watermarked text directly in each of the 17 languages and attacking it by translation into a language drawn at random from a pool of 40 languages. STEAM achieves an average AUC of 0.971 and TPR@1% of 0.504 across source languages (Appendix C.3), in line with the English-origin results.

5.3 Robustness Analysis

Robustness to translator mismatch. The robustness of STEAM should not depend on the spe-

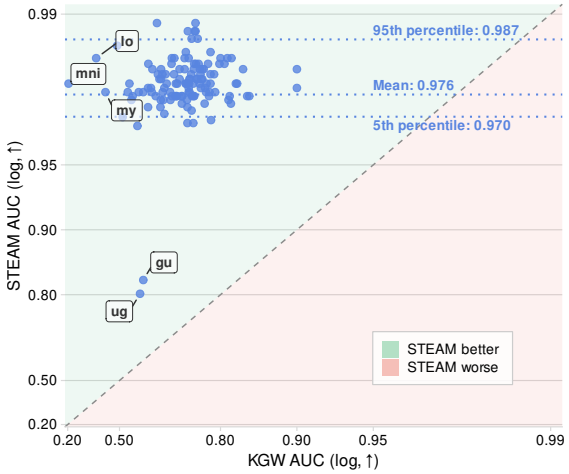


Figure 5: **STEAM generalises to 126 languages.** AUC of STEAM (y-axis) and KGW (x-axis) for 126 attack languages (Aya-23 8B). STEAM always improves the detection of KGW-watermarked texts.

cific translation service used. An adversary could try to bypass our defence by using a different translation system for their attack. We examine whether the performance of STEAM remains robust when the attacker and defender use different translators. Table 5 reports the average AUC across German, Hindi, and Hebrew for all nine combinations of three translation services (Google Translate, DeepSeek-V3.2-Exp, and GPT-4o-mini). All pairs achieve an AUC above 0.94, and the diagonal (matched translator) does not consistently outperform mismatched settings. This suggests that our method genuinely recovers watermark strength independently of the specific translation service rather than relying on translator-specific artefacts. Detection is also possible locally with a small open-weights translator: using NLLB-200 distilled 600M keeps detection reliable, with a mean AUC of 0.907, though the TPR@1% drops (App. C.5).

Adaptive evaluation: multistep translation attack. To assess the robustness of our defence under adaptive attack, we introduce a stronger multistep translation attack that adds an extra translation step beyond the single-hop setup. This design prevents STEAM from relying on direct back-translation to recover the watermark strength. In this two-step attack, the text is first translated using the full set of languages from §4.2, and the resulting output is then translated again through one of three pivot languages: German (high-resource), Korean (medium-resource), or Bengali (low-resource) (Appendix C). Despite this adaptive setup, STEAM remains robust, maintaining an average AUC of

Attacker	Defender		
	G. Translate	DS-V3.2	GPT-4o-mini
Google Translate	0.976	0.974	0.975
DS-V3.2	0.979	0.952	0.943
GPT-4o-mini	0.976	0.973	0.964

Table 5: **STEAM is robust to translator mismatch.** Average AUC (German, Hindi, Hebrew) for all attacker and defender translator combinations. Attacker translator is used for translation attack. Defender translator is used by STEAM. DS-V3.2 is DeepSeek-V3.2-Exp.

0.884 across all conditions. The lowest result occurs with a Korean pivot on high-resource languages (AUC of 0.833), while all other settings remain above 0.860. Although such multi-hop attacks can weaken other defences, they also tend to reduce overall translation quality, limiting their practical impact.

Additional robustness evaluation. We confirm STEAM's robustness across a broader range of scenarios. Under a combined paraphrase-then-translate attack, it retains AUC above 0.967 while X-SIR collapses to 0.492 (App. C.8). Against a fully informed oracle adversary selecting the worst-case attack language, STEAM still achieves AUC 0.802 (App. C.7). Detection takes approximately 17.85 s per text, but is free with Google Translate and requires no GPU (App. C.2). A BO budget of 5 already achieves AUC 0.903, providing an option for faster detection (App. C.6). Performance is stable across text lengths (App. C.4).

5.4 False Positive Analysis

Calibrating the max-over-search statistic. Searching over candidate languages inflates the false-positive rate. STEAM reports the highest corrected z-score among the languages it evaluates, so its null distribution is that of a maximum rather than of a single test. A naive single-test cutoff is therefore invalid: at $z = 2.33$, the nominal 1% threshold, the empirical FPR reaches 25.8% for a pool of 126 languages (Table 6). We instead calibrate the decision threshold τ^* on the null distribution of the full max statistic. All results reported in this paper use this calibration.

Search budget governs the inflation. The inflation is driven by the number of evaluations the search performs, not by the size of the candidate pool. We run the full Bayesian optimisation search on held-out human-written texts, disjoint from the calibration texts used to estimate γ_ℓ, and mea-

Budget	Pool Size	FPR (z=2.33) (↓)	τ^*
Fixed	33	29.2%	3.94
	66	25.8%	3.87
	126	25.8%	3.90
Proportional	33	17.7%	3.60
	66	25.8%	3.87
	126	41.2%	4.07

Table 6: **Search budget governs STEAM's false-positive rate**. Empirical FPR of the max-over-search statistic on held-out human texts. We report FPR at the naive single-test 1% cutoff ($z = 2.33$) and at τ^*, a threshold calibrated to the 99th percentile of the null max-score distribution. A *fixed* budget evaluates 20 candidate languages whatever the pool size, as done throughout this paper, while a *proportional* budget evaluates 30.3% of the pool.

sure the empirical FPR averaged over 17 null languages. Under the fixed budget of 20 evaluations used throughout this paper, FPR is stable as the pool grows, from 29.2% at $P = 33$ to 25.8% at $P = 126$, and τ^* stays close to 3.9 (Table 6). Under a budget scaled with the pool (30.3% of P), FPR instead rises from 17.7% to 41.2%. A fixed budget therefore keeps detection calibrated as language coverage grows.

Why the inflation stays slow. Two properties of the corrected scores bound how fast the null maximum grows. First, the language-specific null hypothesis (Eq. 1) removes the per-language bias that would otherwise let some languages win the maximum regardless of watermark presence (§5.1). Second, the corrected null scores are approximately standard Gaussian, and by extreme-value theory the maximum of P such scores concentrates around $\sqrt{2 \ln P}$, which grows only from ≈ 2.6 at $P = 33$ to ≈ 3.1 at $P = 126$. This bound is conservative, since typologically related languages yield positively correlated scores and thus fewer effective independent tests.

Pool size trades detection power for coverage. Enlarging the candidate pool costs detection power, but far too slowly to erase STEAM's advantage. We measure TPR@1% for pools of 33, 66 and 126 languages, sampled at random from the full set (Table 7). TPR@1% generally decreases as the pool grows, since a fixed budget of 20 evaluations then covers a smaller fraction of the candidates. Yet at $P = 126$, STEAM still outperforms X-KGW and X-SIR on every language.

	TPR@1% (↑)					
Lang.	KGW	X-KGW	X-SIR	STEAM		
				(P=33)	(P=66)	(P=126)
fr	0.224	0.280	0.149	**0.830**	0.696	0.582
de	0.224	0.312	0.163	**0.792**	0.710	0.622
it	0.202	0.354	0.152	**0.710**	0.612	0.530
es	0.232	0.278	0.150	**0.758**	0.622	0.580
hi	0.120	0.194	0.056	**0.808**	0.580	0.644
ko	0.210	0.318	0.115	**0.708**	0.700	0.460
ja	0.114	0.160	0.103	**0.622**	0.508	0.498
bn	0.068	0.180	0.055	0.464	**0.634**	0.604

Table 7: **STEAM remains above all baselines across all pool sizes.** TPR@1% for KGW, X-KGW, X-SIR, and STEAM with back-translation pools of size 33, 66, and 126. Supporting more languages requires a larger pool, which trades increased robustness in new languages for a slight increase in the false positive rate. Even at P=126, STEAM consistently outperforms all baselines across all languages. Red indicates values fall below the undefended KGW baseline. Bolded is best.

6 Conclusion

Current multilingual watermarking methods fail to remain robust under translation attacks, especially in medium- and low-resource languages. If watermarking lacks robustness in a given language, online content in that language may be disproportionately affected by synthetic or undesirable content. This risk is especially serious for low- and medium-resource languages, which already face a shortage of high-quality digital resources and often lack effective moderation systems.

To address this, we introduced STEAM , a watermark-agnostic method that applies Bayesian optimisation to select the back-translation language that best recovers watermark strength. Extensive experiments scaling to 126 languages and diverse attack scenarios show that STEAM achieves consistently stronger robustness and fairness than existing multilingual watermarking methods, particularly in medium- and low-resource settings.

Our findings highlight the need for watermarking research to treat linguistic diversity as a core requirement, ensuring that the security and trust of large language models extend to all languages, not only those with abundant digital resources.

Limitations

STEAM is designed for translation attacks only. Multilingual robustness failures stem from structural properties of tokeniser vocabularies, which underrepresent low- and medium-resource languages (§4.3). This specificity has motivated dedicated multilingual watermarking techniques, confirming that multilinguality constitutes a distinct research challenge. Other threat models, such as paraphrasing, are addressed by separate methods (Zhao et al., 2023; Liu et al., 2024a; Ren et al., 2024). A key property of translation attacks is their invertibility: STEAM recovers watermark strength by translating back into a language close to the original. Paraphrasing offers no such inverse operation, which means STEAM addresses translation attacks only.

STEAM requires access to a translation service for the target language. Such services may not exist for the most under-resourced languages. However, LLMs themselves tend to perform poorly in languages without translation support, so the practical scope of the limitation is narrow. Additionally, STEAM does not require high-quality translation: our robustness experiments show that different translation services yield comparable results (§5.3).

STEAM involves a trade-off between detection cost and detection accuracy. Each suspect text requires several back-translations, which introduce latency and, when a paid translation service is used, a monetary cost. This overhead affects detection only, not generation: users experience no added latency, since STEAM does not modify the generation. The search budget controls both the cost and the accuracy of detection, so a smaller budget yields cheaper but less accurate detection (Appendices C.6, C.2). A comparable trade-off applies to language coverage, where a larger candidate pool extends protection to more attack languages but reduces detection power (§5.4). STEAM is therefore appropriate for offline detection, such as provenance auditing or post-hoc moderation, but not for real-time or streaming verification.

Finally, despite extensive testing, the implementation of our experiments may contain implementation errors.

Ethical Considerations

This work has potential dual-use implications. On one hand, studying adversarial attacks against watermarking could inform malicious actors about possible strategies to weaken watermark defences. However, we believe the benefits outweigh these risks.

First, our contribution is not only an analysis but also a concrete defence (STEAM) that achieves a high level of robustness, substantially exceeding prior multilingual watermarking methods. Our results demonstrate that STEAM provides consistently strong robustness against translation attacks across a wide range of languages.

Second, by explicitly addressing low- and medium-resource languages, our method promotes fairness: watermarking becomes more reliable across diverse linguistic settings, rather than being limited to a handful of high-resource languages.

Robust multilingual watermarking is an important safeguard against misuse of large language models, such as the generation and dissemination of fake news or disinformation in less-resourced languages where moderation tools are often weaker. We view this work as a step toward improving the security and trustworthiness of multilingual AI systems.

STEAM translates the suspect text at detection time. Using a third-party translation API therefore exposes that text to an external provider, which is undesirable when detection is applied to sensitive material. STEAM makes no assumption about the translator, so detection can run entirely locally with an open-weights translation model or a self-hosted LLM, and no text needs to leave the detector's infrastructure. Our translator mismatch results confirm that no specific service is required (§5.3), and we further find that a small open-weights translation model run locally is sufficient for reliable detection. We recommend local translation whenever the text under examination is sensitive.

Acknowledgments

This work was supported by Parameter Lab, which provided the compute resources and covered the API costs of large language models.

References

Mansour Al Ghanim, Jiaqi Xue, Rochana Prih Hastuti, Mengxin Zheng, Yan Solihin, and Qian Lou. 2025. Evaluating the robustness and accuracy of text watermarking under real-world cross-lingual manipulations. In *Findings of the Association for Computational Linguistics: EMNLP 2025*. Association for Computational Linguistics.

Viraat Aryabumi, John Dang, Dwarak Talupuru, Saurabh Dash, David Cairuz, Hangyu Lin, Bharat Venkitesh, Madeline Smith, Kelly Marchisio, Sebastian Ruder, Acyr Locatelli, Julia Kreutzer, Nick Frosst, Phil Blunsom, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. 2024. Aya 23: Open weight releases to further multilingual progress. *Preprint*, arXiv:2405.15032.

Maximilian Balandat, Brian Karrer, Daniel R. Jiang, Samuel Daulton, Benjamin Letham, Andrew Gordon Wilson, and Eytan Bakshy. 2020. BoTorch: A Framework for Efficient Monte-Carlo Bayesian Optimization. In *Advances in Neural Information Processing Systems 33*.

Liang Chen, Yatao Bian, Yang Deng, Deng Cai, Shuaiyi Li, Peilin Zhao, and Kam fai Wong. 2024. Watme: Towards lossless watermarking through lexical redundancy. *Preprint*, arXiv:2311.09832.

Ruibo Chen, Yihan Wu, Junfeng Guo, and Heng Huang. 2025. Improved unbiased watermark for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20587–20601, Vienna, Austria. Association for Computational Linguistics.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Costa-jussà, NLLB Team, Marta R., James Cross, Onur Çelebi, Maha Elbayad, Kenneth Heafield, Kevin Heffernan, Elahe Kalbassi, Janice Lam, Daniel Licht, Jean Maillard, Anna Sun, Skyler Wang, Guillaume Wenzek, Al Youngblood, Bapi Akula, Loic Barrault, Gabriel Mejia Gonzalez, and 21 others. 2022. No language left behind: Scaling human-centered machine translation. *Preprint*, arXiv:2207.04672.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, Jamie Hayes, Nidhi Vyas, Majd Al Merey, Jonah Brown-Cohen, Rudy Bunel, Borja Balle, Taylan Cemgil, Zahra Ahmed, Kitty Stacpoole, and 5 others. 2024. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823.

Yu Fu, Deyi Xiong, and Yue Dong. 2024. Watermarking conditional text generation for ai detection: Unveiling challenges and a semantic-aware watermark remedy. *Preprint*, arXiv:2307.13808.

Jacob R. Gardner, Geoff Pleiss, David Bindel, Kilian Q. Weinberger, and Andrew Gordon Wilson. 2021. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Preprint*, arXiv:1809.11165.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yuxuan Guo, Zhiliang Tian, Yiping Song, Tianlun Liu, Liang Ding, and Dongsheng Li. 2024. Context-aware watermark with semantic balanced green-red lists for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22633–22646, Miami, Florida, USA. Association for Computational Linguistics.

Xia Han, Qi Li, Jianbing Ni, and Mohammad Zulkernine. 2025. Robustness assessment and enhancement of text watermarking for google's synthid. *Preprint*, arXiv:2508.20228.

Zhiwei He, Binglin Zhou, Hongkun Hao, Aiwei Liu, Xing Wang, Zhaopeng Tu, Zhuosheng Zhang, and Rui Wang. 2024. Can watermarks survive translation? on the cross-lingual consistency of text watermark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4115–4129, Bangkok, Thailand. Association for Computational Linguistics.

Abe Hou, Jingyu Zhang, Tianxing He, Yichen Wang, Yung-Sung Chuang, Hongwei Wang, Lingfeng Shen, Benjamin Van Durme, Daniel Khashabi, and Yulia Tsvetkov. 2024. SemStamp: A semantic watermark with paraphrastic robustness for text generation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4067–4082, Mexico City, Mexico. Association for Computational Linguistics.

Zhengmian Hu, Lichang Chen, Xidong Wu, Yihan Wu, Hongyang Zhang, and Heng Huang. 2023. Unbiased watermark for large language models. *Preprint*, arXiv:2310.10669.

Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, page 604–613, New York, NY, USA. Association for Computing Machinery.

Aditya Khan, Mason Shipton, David Anugraha, Kaiyao Duan, Phuong H. Hoang, Eric Khiu, A. Seza

Doğruöz, and En-Shiun Annie Lee. 2025. Uriel+: Enhancing linguistic inclusion and usability in a typological and multilingual knowledge base. *Preprint*, arXiv:2409.18472.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 17061–17084. PMLR.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2024. Robust distortion-free watermarks for language models. *Preprint*, arXiv:2307.15593.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2024. Who wrote this code? watermarking for code generation. *Preprint*, arXiv:2305.15060.

Patrick Littell, David R. Mortensen, Ke Lin, Katherine Kairis, Carlisle Turner, and Lori Levin. 2017. URIEL and lang2vec: Representing languages as typological, geographical, and phylogenetic vectors. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 8–14, Valencia, Spain. Association for Computational Linguistics.

Aiwei Liu, Leyi Pan, Xuming Hu, Shiao Meng, and Lijie Wen. 2024a. A semantic invariant robust watermark for large language models. In *The Twelfth International Conference on Learning Representations*.

Aiwei Liu, Leyi Pan, Yijian Lu, Jingjing Li, Xuming Hu, Xi Zhang, Lijie Wen, Irwin King, Hui Xiong, and Philip S. Yu. 2024b. A survey of text watermarking in the era of large language models. *Preprint*, arXiv:2312.07913.

Yepeng Liu and Yuheng Bu. 2024. Adaptive text watermark for large language models. *Preprint*, arXiv:2401.13927.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. 2024a. An entropy-based text watermarking detection method. *Preprint*, arXiv:2403.13485.

Yinquan Lu, Wenhao Zhu, Lei Li, Yu Qiao, and Fei Yuan. 2024b. LLaMAX: Scaling linguistic horizons of LLM by enhancing translation capabilities beyond 100 languages. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10748–10772, Miami, Florida, USA. Association for Computational Linguistics.

Yiyang Luo, Ke Lin, Chao Gu, Jiahui Hou, Lijie Wen, and Luo Ping. 2025. Lost in overlap: Exploring logit-based watermark collision in LLMs. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 620–637, Albuquerque, New Mexico. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. *Preprint*, arXiv:1912.01703.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Jie Ren, Han Xu, Yiding Liu, Yingqian Cui, Shuaiqiang Wang, Dawei Yin, and Jiliang Tang. 2024. A robust semantics-based watermark for large language model against paraphrasing. *Preprint*, arXiv:2311.08721.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Yihan Wu, Zhengmian Hu, Junfeng Guo, Hongyang Zhang, and Heng Huang. 2024. A resilient and accessible distribution-preserving watermark for large language models. *Preprint*, arXiv:2310.07710.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. 2023. Provable robust watermarking for ai-generated text. *Preprint*, arXiv:2306.17439.

Appendix

The appendices contain the following sections:

- Appendix A details the experimental settings,
- Appendix B contains additional experimental results of Section 4 about semantic clustering,
- Appendix C contains additional experimental results of Section 5 about Steam,
- Appendix D contains our usage of AI assistants.
- Appendix E contains the details of our artifacts.

For transparency and reproducibility, our code is available on GitHub at `https://github.com/asimzz/steam`

A Experimental Setting

A.1 Hyperparameters

To ensure reproducibility, we detail the hyperparameters used for both the neural network training and the watermark generation/detection phases of our experiments.

X-SIR neural network training. The neural network component of X-SIR, which inherits its architecture from SIR, was trained using the following hyperparameters:

- Architecture. The model consists of 4 layers, with an input dimension of 1024, a hidden dimension of 500, and an output dimension of 300.
- Optimization. We used *Stochastic Gradient Descent (SGD)* with a *learning rate* of 0.006 and a *weight decay* of 0.2. A `StepLR` scheduler with a *step size* of 200 and a gamma of 0.1 was employed to adjust the learning rate during training.
- Training. The model was trained for 2000 epochs with a batch size of 32.

Watermarking scheme parameters. For the watermark generation and detection phases, the following parameters were used for each scheme:

- KGW. We used the default parameters recommended by Kirchenbauer et al. (2023): a green list proportion (gamma) of 0.25, a logit bias (delta) of 2.0, and the `minhash` seeding scheme.
- X-KGW. To create a direct comparison with XSIR, we set the context width to 1. The gamma and delta values were kept consistent with KGW at 0.25 and 2.0, respectively.
- X-SIR. We followed the original implementation, setting the window size to 5, the chunk size to 10, and the logit bias (delta) to 1.0. The multilingual sentence embeddings were generated using the `paraphrase-multilingual-mpnet-base-v2` model.

A.2 Computational Resources & Softwares

All experiments were conducted on a Google Cloud Platform instance of type `n1-standard-4`, equipped with 4 vCPUs, 15 GB of RAM, and two NVIDIA T4 GPUs.

All translations use the Google Translate service accessed through the `deep_translator` Python library, which provides a unified interface to various translation APIs. The translator mismatch experiment in §5.3 employs the DeepSeek API for back-translation.

We used Pytorch as our deep learning framework (Paszke et al., 2019), with CUDA support for GPU acceleration. In addition, we employed Hugging Face Transformers library[2] (Wolf et al., 2020) to access pretrained models and tokenizers.

A.3 Back-Translation Candidate Languages

STEAM searches over a pool of candidate back-translation languages using Google Translate as the default translation service. Table 8 lists all candidate languages used in our experiments for § 5.2 § 5.3. These languages span diverse language families, scripts, and resource levels, covering high-resource (e.g., English, French, German), medium-resource (e.g., Hungarian, Romanian, Thai), and low-resource (e.g., Tigrinya, Xhosa, Yoruba) settings.

A.4 Description of X-KGW

X-KGW (Cross-lingual KGW) is a hybrid watermarking approach we introduce to combine the hash-based mechanism of KGW with the semantic clustering strategy of X-SIR. Unlike KGW, which partitions individual tokens, X-KGW operates at the cluster level. The process consists of three distinct phases:

[2] `https://huggingface.co/`

Code	Language	Code	Language	Code	Language
af	Afrikaans	iw	Hebrew	fa	Persian
sq	Albanian	hi	Hindi	pl	Polish
am	Amharic	hu	Hungarian	pt	Portuguese
ar	Arabic	is	Icelandic	pa	Punjabi
hy	Armenian	ig	Igbo	ro	Romanian
as	Assamese	ilo	Ilocano	ru	Russian
ay	Aymara	id	Indonesian	sm	Samoan
az	Azerbaijani	ga	Irish	sa	Sanskrit
bm	Bambara	it	Italian	gd	Scottish Gaelic
eu	Basque	ja	Japanese	nso	Sepedi
be	Belarusian	jw	Javanese	sr	Serbian
bn	Bengali	kn	Kannada	st	Sesotho
bho	Bhojpuri	kk	Kazakh	sn	Shona
bs	Bosnian	km	Khmer	sd	Sindhi
bg	Bulgarian	rw	Kinyarwanda	si	Sinhala
ca	Catalan	ko	Korean	sk	Slovak
ceb	Cebuano	kri	Krio	sl	Slovenian
ny	Chichewa	ku	Kurdish (Kurmanji)	so	Somali
zh-CN	Chinese (Simp.)	ckb	Kurdish (Sorani)	es	Spanish
zh-TW	Chinese (Trad.)	ky	Kyrgyz	su	Sundanese
co	Corsican	lo	Lao	sw	Swahili
hr	Croatian	la	Latin	sv	Swedish
cs	Czech	lv	Latvian	tg	Tajik
da	Danish	ln	Lingala	ta	Tamil
dv	Dhivehi	lt	Lithuanian	tt	Tatar
nl	Dutch	lg	Luganda	te	Telugu
en	English	lb	Luxembourgish	th	Thai
eo	Esperanto	mk	Macedonian	ti	Tigrinya
et	Estonian	mai	Maithili	ts	Tsonga
ee	Ewe	mg	Malagasy	tr	Turkish
tl	Filipino	ms	Malay	tk	Turkmen
fi	Finnish	ml	Malayalam	ak	Twi
fr	French	mt	Maltese	uk	Ukrainian
fy	Frisian	mi	Maori	ur	Urdu
gl	Galician	mr	Marathi	ug	Uyghur
ka	Georgian	mni-Mtei	Meitei	uz	Uzbek
de	German	lus	Mizo	vi	Vietnamese
el	Greek	mn	Mongolian	cy	Welsh
gu	Gujarati	my	Myanmar	xh	Xhosa
ht	Haitian Creole	ne	Nepali	yi	Yiddish
ha	Hausa	no	Norwegian	yo	Yoruba
haw	Hawaiian	or	Odia	zu	Zulu

Table 8: All 126 candidate back-translation languages supported by STEAM.

1. Semantic cluster construction. Following the X-SIR framework, we first construct a multilingual semantic graph using bilingual translation dictionaries. The Louvain community detection algorithm is then applied to partition the vocabulary $\mathcal{V}$ into $\mathcal{C}$ disjoint semantic clusters, yielding a token-to-cluster mapping $m : \mathcal{V} \to 0, 1, \ldots, \mathcal{C} - 1$.

2. Hash-based cluster partitioning. During text generation, at each timestep t, a context window of preceding tokens $(w_{t-h}, \ldots, w_{t-1})$ is used to compute a hash-based seed. This seed is then employed to pseudo-randomly partition the $\mathcal{C}$ clusters into green and red sets, with a fraction γ designated as green.

3. Cluster-based logit modification. Finally, a positive bias δ is applied to the logits of all tokens belonging to clusters assigned to the green set. The model then samples the next token from this modified probability distribution.

By combining KGW logit biasing with semantic clustering, X-KGW seeks to preserve watermark robustness under multilingual transformations while maintaining detection accuracy.

A.5 Multilingual Dictionaries & Language Categorization

To construct our multilingual dictionary, we relied on the MUSE dictionary (Conneau et al., 2017), the same resource used by He et al. (2024) to build

the semantic clusters. In addition to its role in dictionary-based clustering, we used MUSE to categorize the 17 languages included in our evaluation of §4.2, §4.3, §5.2, §5.3.

- A language was marked *high-resource* if it possesses extensive, non-English-centric dictionary mappings (i.e., bidirectional dictionaries with multiple other languages in the set).
- In contrast, languages whose resources are primarily English-centric, where MUSE provides only bidirectional dictionaries with English, were classified as either *medium-resource* or *low-resource*. The distinction between these two groups was determined by the size (i.e., the number of word pairs) of their respective English dictionaries.

A.6 Definitions of Multilingual Watermarking Comparison Criteria

For clarity, we provide the definitions of the criteria used in Table 1 to compare multilingual watermarking techniques:

- **Multilingual.** Designed to resist translation attacks.
- **Medium/low-resource.** Robust against translation attacks into medium- or low-resource languages.
- **Tokenizer-robust.** Robustness against translation attacks does not depend on the tokenizer.
- **Non-invasive.** Multilingual support does not modify generation logits, so text quality is preserved.
- **Watermark-agnostic.** Compatible with any watermarking technique without modification.
- **Retroactive support.** New languages can be added without regenerating the watermark key; already-watermarked texts remain detectable.

A.7 Prompt for DeepSeek-V3.2-Exp Translation

To use DeepSeek-V3.2-Exp as a translation engine, we designed a structured prompt format. We define:

```
Source language: {src_lang}
Target language: {tgt_lang}
Input text: {response}
```

`src_lang` and `tgt_lang` indicate the source and target language codes. We convert these codes into their full language names using the `Language` class from the `langcodes` [3] library:

```
Language.make(language=src_lang).display_name()
Language.make(language=tgt_lang).display_name()
```

The final prompt provided to DeepSeek-V3.2-Exp:

```
Translate the following
{Language.make(language=src_lang
    ).display_name()}
text to
{Language.make(language=tgt_lang
    ).display_name()}:

{response}
```

A.8 Bayesian Optimisation Details

Surrogate model. We use a single-task Gaussian Process (GP) with a constant mean function as implemented by `SingleTaskGP` in BoTorch (Balandat et al., 2020). The GP hyperparameters (kernel lengthscales, output scale, and noise variance) are optimised by maximising the exact marginal log-likelihood using GPyTorch (Gardner et al., 2021). The GP is refitted from scratch at each BO iteration.

Acquisition function. We use Log Expected Improvement (`LogExpectedImprovement` in BoTorch), a numerically stable variant of expected improvement that operates in log-space to avoid vanishing gradients when the current best value is far from the predictive mean. Since the search space is a finite set of discrete languages rather than a continuous domain, we evaluate the acquisition function at all unevaluated candidate feature vectors and select the one with the highest value.

Language feature vectors. Each candidate probe language is represented by a 131-dimensional feature vector obtained by concatenating `syntax_knn` (103 dimensions) and `phonology_knn` (28 dimensions) from the URIEL typological database (Khan et al., 2025), accessed via `lang2vec` (Littell et al., 2017). Feature vectors are pre-computed once and reused across all texts.

Budget. Each text is allocated a maximum of 20 evaluations (3 initial + 17 BO iterations).

Implementation. The full pipeline is implemented in Python using BoTorch 0.11+, GPyTorch 1.11+, and PyTorch 2.0+.

[3] https://pypi.org/project/langcodes/

B Additional Results About Semantic Clustering

This Appendix contains the additional results of Section 4.

B.1 Hold-out languages for X-SIR

Tables 9 and 10 report hold-one-out results for Aya-23 8B and LLaMA-3.2 1B respectively.

B.2 Unsupported languages for X-SIR & X-KGW

New Lang.	X-SIR (↑)		X-KGW (↑)	
	AUC	TPR@1%	AUC	TPR@1%
it	0.796	0.177	0.772	0.238
es	0.754	0.155	0.807	0.230
pt	0.775	0.133	0.792	0.286
pl	0.749	0.127	0.762	0.236
nl	0.776	0.164	0.808	0.314
hr	0.726	0.124	0.757	0.210
cs	0.773	0.111	0.754	0.254
da	0.734	0.161	0.764	0.266
ko	0.729	0.136	0.754	0.226
ar	0.687	0.093	0.765	0.168
Min.	0.687 (ar)	0.093 (ar)	0.754 (cs, ko)	0.168 (ar)

Table 11: **Semantic clustering is weak for unsupported languages**. Watermark strength (AUC and TPR@1%) of X-SIR and X-KGW, limited to the five originally supported languages (en, fr, de, zh, ja). Aya-23 8B generates English text, which is then translated into a new unsupported language for evaluation. Minimum marks the weakest robustness (best attack case).

New Lang.	X-SIR (↑)		X-KGW (↑)	
	AUC	TPR@1%	AUC	TPR@1%
it	0.699	0.069	0.760	0.212
es	0.665	0.076	0.744	0.222
pt	0.641	0.059	0.722	0.152
pl	0.679	0.069	0.677	0.144
nl	0.754	0.095	0.781	0.244
hr	0.660	0.066	0.733	0.162
cs	0.650	0.064	0.759	0.190
da	0.675	0.093	0.765	0.196
ko	0.673	0.062	0.672	0.124
ar	0.655	0.055	0.704	0.168
Min.	0.641 (pt)	0.055 (ar)	0.672 (ko)	0.124 (ko)

Table 12: **Semantic clustering is weak for unsupported languages**. Watermark strength (AUC and TPR@1%) of X-SIR and X-KGW, limited to the five originally supported languages (en, fr, de, zh, ja). LLaMA-3.2 1B generates English text, which is then translated into a new unsupported language for evaluation. Minimum marks the weakest robustness (best attack case).

New Lang.	X-SIR (↑)		X-KGW (↑)	
	AUC	TPR@1%	AUC	TPR@1%
it	0.829	0.335	0.860	0.510
es	0.810	0.314	0.864	0.490
pt	0.812	0.337	0.861	0.498
pl	0.812	0.308	0.817	0.420
nl	0.845	0.351	0.896	0.584
hr	0.804	0.279	0.822	0.344
cs	0.798	0.305	0.838	0.444
da	0.832	0.357	0.871	0.436
ko	0.792	0.276	0.800	0.390
ar	0.765	0.251	0.815	0.358
Min.	0.765 (ar)	0.251 (ar)	0.800 (ko)	0.344 (hr)

Table 13: **Semantic clustering is weak for unsupported languages**. Watermark strength (AUC and TPR@1%) of X-SIR and X-KGW, limited to the five originally supported languages (en, fr, de, zh, ja). LLaMAX-3 8B generates English text, which is then translated into a new unsupported language for evaluation. Minimum marks the weakest robustness (best attack case).

B.3 Supported languages for X-SIR & X-KGW

B.4 Tokenizer vocabulary analysis

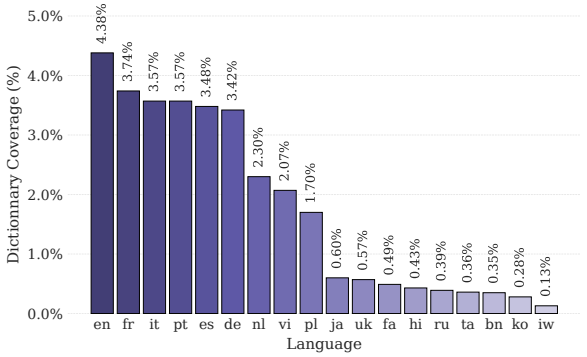


Figure 6: **Tokenizer vocabulary favours high-resource languages**. Percentage of words in multilingual dictionaries that appear in the tokenizer vocabulary.

B.5 Sub-character Token Distributions

As discussed in §5.1, the z-score normalization component is designed to calibrate STEAM's detection mechanism against statistical noise introduced by tokenizer limitations. Figures 7 and 8 show the token distribution for two severely affected low-resource languages, Bengali and Tamil.

Languages		AUC (↑)			TPR@1% (↑)		
Held-out	Prompt	Held-Out	Supported	Δ	Held-Out	Supported	Δ
en	fr	0.795 ±0.045	0.816 ±0.014	+0.021	0.198 ±0.049	0.149 ±0.042	-0.049
	de	0.780 ±0.054	0.811 ±0.018	+0.031	0.172 ±0.047	0.168 ±0.039	-0.004
	zh	0.731 ±0.020	0.669 ±0.042	-0.062	0.141 ±0.011	0.083 ±0.014	-0.058
fr	en	0.757 ±0.022	0.799 ±0.027	+0.042	0.157 ±0.016	0.139 ±0.037	-0.018
	de	0.723 ±0.020	0.781 ±0.025	+0.058	0.101 ±0.029	0.156 ±0.061	+0.055
	zh	0.651 ±0.026	0.638 ±0.037	-0.013	0.076 ±0.021	0.052 ±0.020	-0.024
de	en	0.736 ±0.011	0.802 ±0.020	+0.067	0.153 ±0.050	0.214 ±0.035	+0.061
	fr	0.765 ±0.004	0.784 ±0.020	+0.019	0.139 ±0.068	0.118 ±0.039	-0.021
	zh	0.667 ±0.041	0.642 ±0.014	-0.025	0.073 ±0.032	0.065 ±0.016	-0.008
zh	en	0.644 ±0.052	0.692 ±0.041	+0.048	0.120 ±0.046	0.111 ±0.011	-0.009
	fr	0.671 ±0.072	0.714 ±0.045	+0.043	0.112 ±0.053	0.069 ±0.025	-0.043
	de	0.675 ±0.048	0.701 ±0.026	+0.026	0.105 ±0.053	0.107 ±0.027	+0.002
ja	en	0.685 ±0.059	0.656 ±0.017	-0.029	0.113 ±0.024	0.070 ±0.008	-0.043
	fr	0.698 ±0.038	0.670 ±0.018	-0.028	0.101 ±0.033	0.089 ±0.023	-0.012
	de	0.688 ±0.037	0.669 ±0.027	-0.019	0.138 ±0.031	0.079 ±0.005	-0.059
	zh	0.681 ±0.043	0.658 ±0.003	-0.023	0.110 ±0.046	0.093 ±0.016	-0.017

Table 9: **Hold-one-out: X-SIR is slightly sensitive to the set of supported languages.** Δ = Supported − Held-out, where *Supported* uses all five original languages (en, fr, de, zh, ja) and *Held-out* excludes the target language from the cluster; the attack always targets the held-out language. Aya-23 8B generates text in the Prompt language. Differences are small ($|\Delta| \leq 0.07$ for AUC) but variable across configurations and models, indicating mild sensitivity to which languages are included. Red indicates negative Δ.

Languages		AUC (↑)			TPR@1% (↑)		
Held-out	Prompt	Held-Out	Supported	Δ	Held-Out	Supported	Δ
en	fr	0.901±0.017	0.907±0.015	+0.006	0.363±0.047	0.275±0.007	-0.088
	de	0.884±0.043	0.894±0.041	+0.010	0.379±0.131	0.331±0.043	-0.048
	zh	0.795±0.043	0.827±0.013	+0.032	0.450±0.040	0.407±0.011	-0.043
fr	en	0.743±0.050	0.682±0.026	-0.061	0.113±0.056	0.068±0.012	-0.045
	de	0.787±0.050	0.687±0.011	-0.100	0.161±0.038	0.093±0.007	-0.068
	zh	0.678±0.006	0.681±0.007	+0.003	0.086±0.027	0.083±0.005	-0.003
de	en	0.693±0.017	0.692±0.043	-0.001	0.069±0.022	0.068±0.025	-0.001
	fr	0.724±0.023	0.725±0.044	+0.001	0.098±0.011	0.071±0.009	-0.027
	zh	0.665±0.018	0.693±0.046	+0.028	0.065±0.009	0.073±0.014	+0.008
zh	en	0.666±0.035	0.605±0.012	-0.061	0.082±0.052	0.026±0.006	-0.056
	fr	0.704±0.011	0.609±0.024	-0.095	0.085±0.041	0.036±0.012	-0.049
	de	0.703±0.022	0.636±0.017	-0.067	0.088±0.010	0.050±0.008	-0.038
ja	en	0.576±0.052	0.573±0.024	-0.003	0.039±0.019	0.033±0.005	-0.006
	fr	0.630±0.066	0.581±0.041	-0.049	0.067±0.016	0.025±0.014	-0.042
	de	0.624±0.055	0.589±0.039	-0.035	0.074±0.023	0.037±0.018	-0.037
	zh	0.663±0.059	0.650±0.026	-0.013	0.121±0.065	0.075±0.040	-0.046

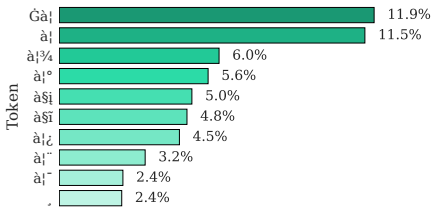
Table 10: **Hold-one-out: X-SIR is slightly sensitive to the set of supported languages.** Δ = Supported − Held-out, where *Supported* uses all five original languages (en, fr, de, zh, ja) and *Held-out* excludes the target language from the cluster; the attack always targets the held-out language. LLaMA-3.2 1B generates text in the Prompt language. Differences are small ($|\Delta| \leq 0.10$ for AUC) but variable across configurations and models, indicating mild sensitivity to which languages are included. Red indicates negative Δ.

Translation Attack		X-SIR (↑)		X-KGW (↑)	
Type	Language	AUC	TPR@1%	AUC	TPR@1%
High-resource	fr	0.702	0.085	0.719	0.166
	de	0.708	0.067	0.752	0.186
	it	0.712	0.111	0.750	0.230
	es	0.703	0.089	0.724	0.222
	pt	0.726	0.102	0.747	0.206
Medium-resource	pl	0.657	0.065	0.703	0.188
	nl	0.722	0.091	0.787	0.252
	ru	0.635	0.075	0.656	0.100
	hi	0.611	0.037	0.620	0.084
	ko	0.673	0.055	0.701	0.154
	ja	0.571	0.042	0.598	0.110
Low-resource	bn	0.825	0.509	0.701	0.078
	fa	0.584	0.055	0.673	0.086
	vi	0.691	0.084	0.722	0.186
	iw	0.622	0.026	0.702	0.134
	uk	0.613	0.064	0.725	0.118
	ta	0.749	0.095	0.672	0.108
	Minimum	0.571(ja)	0.026(iw)	0.598(ja)	0.078(bn)

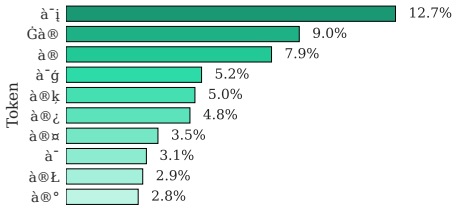
Table 15: **Semantic clustering performs poorly on an expanded set of supported languages**. The semantic clustering is applied using an expanded set of 17 newly supported languages. LLaMA-3.2 1B generates a text in English, then the translation attack is applied using each of these supported languages as target language. Higher values indicate better robustness. Minimum indicates the worst-case robustness, i.e., the best language for an attack.

Translation Attack		X-SIR (↑)		X-KGW (↑)	
Type	Language	AUC	TPR@1%	AUC	TPR@1%
High-resource	fr	0.804	0.249	0.852	0.466
	de	0.833	0.399	0.850	0.484
	it	0.829	0.336	0.870	0.478
	es	0.811	0.319	0.869	0.506
	pt	0.726	0.338	0.863	0.454
Medium-resource	pl	0.812	0.308	0.847	0.410
	nl	0.847	0.355	0.882	0.592
	ru	0.787	0.256	0.821	0.368
	hi	0.702	0.215	0.714	0.228
	ko	0.792	0.276	0.822	0.422
	ja	0.714	0.187	0.705	0.206
Low-resource	bn	0.588	0.086	0.765	0.244
	fa	0.755	0.268	0.829	0.398
	vi	0.772	0.238	0.802	0.328
	iw	0.719	0.196	0.808	0.444
	uk	0.794	0.309	0.817	0.118
	ta	0.561	0.067	0.789	0.316
	Minimum	0.561(ta)	0.067(ta)	0.705(ja)	0.118(uk)

Table 16: **Semantic clustering performs poorly on an expanded set of supported languages**. The semantic clustering is applied using an expanded set of 17 newly supported languages. LLaMAX-3 8B generates a text in English, then the translation attack is applied using each of these supported languages as target language. Higher values indicate better robustness. Minimum indicates the worst-case robustness, i.e., the best language for an attack.

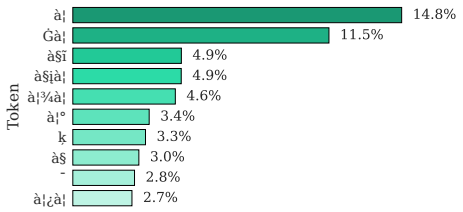


(a) **bn**

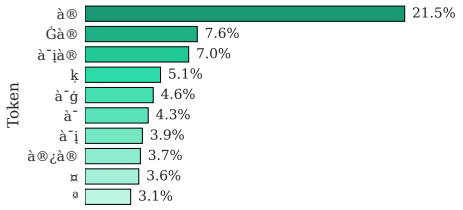


(b) **ta**

Figure 7: **Tokenization of low-resource languages creates highly concentrated sub-character tokens.** Percentage of top 10 tokens for Bengali (a) and for Tamil (b) using Aya-23 8B.



(a) **bn**



(b) **ta**

Figure 8: **Tokenization of low-resource languages creates highly concentrated sub-character tokens.** Percentage of top 10 tokens for Bengali (a) and for Tamil (b) using LLaMA-3.2 1B.

CWRA Attack		Aya-23 8B (↑)		LLaMA-3.2 1B (↑)		LLaMAX-3 8B (↑)	
Type	Language	AUC	TPR@1%	AUC	TPR@1%	AUC	TPR@1%
High-resource	fr	0.831	0.201	0.898	0.421	0.845	0.345
	de	0.820	0.198	0.902	0.394	0.841	0.361
	it	0.817	0.196	0.872	0.331	0.825	0.315
	es	0.819	0.200	0.859	0.281	0.816	0.320
	pt	0.801	0.191	0.876	0.348	0.827	0.335
Medium-resource	pl	0.804	0.202	0.881	0.381	0.815	0.291
	nl	0.859	0.265	0.899	0.434	0.834	0.367
	ru	0.769	0.175	0.824	0.226	0.771	0.233
	hi	0.710	0.147	0.771	0.304	0.744	0.263
	ko	0.769	0.178	0.854	0.340	0.805	0.299
	ja	0.787	0.278	0.904	0.644	0.720	0.314
Low-resource	bn	0.721	0.255	0.934	0.628	0.784	0.354
	fa	0.691	0.113	0.796	0.260	0.733	0.246
	vi	0.781	0.180	0.865	0.325	0.805	0.301
	iw	0.726	0.124	0.815	0.207	0.729	0.245
	uk	0.769	0.157	0.849	0.253	0.750	0.225
	ta	0.819	0.405	0.917	0.516	0.740	0.391
	Minimum	0.691(fa)	0.113(fa)	0.771(hi)	0.207(iw)	0.720(ja)	0.225(uk)

Table 17: **Semantic clustering (XSIR) performs inconsistently on an expanded set of supported languages**. The semantic clustering is applied using an expanded set of 17 newly supported languages. A prompt in English is first translated into each target language. Aya-23 8B, LLaMA-3.2 1B, and LLaMAX-3 8B are then prompted with the translated input to generate text in the target language. Finally, the CWRA attack is applied by translating the generated text back into English. Higher values indicate better robustness. Minimum indicates the worst-case robustness, i.e., the best language for an attack.

C Additional Results About STEAM

Tables 18 and 21 report STEAM performance on 17 supported languages for LLaMA-3.2 1B and Gemma4-E4B, complementing the Aya-23 8B results in Table 3. Table 22 reports STEAM performance on all 126 supported languages for Aya-23 8B. Tables 19 and 20 report results on 10 unsupported languages for Aya-23 8B and LLaMA-3.2 1B, respectively, complementing the AUC summary in Table 4. Table 23 reports results under the multistep translation attack. Table 31 shows the impact of per-language z-score normalization on STEAM's performance

C.1 Inference Cost

STEAM introduces no additional cost at generation time, as text is produced by the unmodified watermarking scheme (e.g., vanilla KGW). The only cost is at detection time, where STEAM requires translating the suspect text into back-translation languages (Table 24). Google Translate is free, while high-quality LLM-based translators such as DeepSeek-V3.2-Exp cost $0.003 per text, making STEAM practical even at scale.

Model	Cost per text ($)
Google Translate	0.000
DeepSeek-V3.2-Exp	0.060
GPT-5 mini	0.180
Gemini 2.5 Flash	0.220

Table 24: **STEAM requires minimal cost at detection time**. Total cost in US dollars for 20 back-translations per text (the maximum number of evaluated languages by BO). Google Translate is free. STEAM adds no cost at generation time.

C.2 Detection-Time Cost

We measure wall-clock latency on 20 watermarked texts using Google Translate at BO budget 20. The mean per-text latency is 17.85 s, decomposed as follows: defender back-translation (4.14 s), local KGW detection (2.76 s), GP refitting (10.64 s), and acquisition scoring (0.16 s). GP refitting dominates but runs entirely on CPU, requires no GPU, and parallelises trivially across suspect texts, allowing deployers to saturate available cores at negligible marginal cost. Detection generally does not need to operate in real time, so the latency should be a minor practical issue.

C.3 Non-English Source Languages

The main results watermark English text and attack it by translating out of English. We invert that setting. The watermarked text is generated directly in a non-English source language, and each text is then attacked by a translation into a language drawn uniformly at random from a 40-language pool spanning all three resource tiers. The detector is therefore never tuned to a single known attack language. Across the 17 source languages in Table 25, STEAM holds a mean TPR@1% of 0.504, ranging from 0.321 for Ukrainian to 0.638 for Japanese. Mean AUC is 0.971 over the languages measured, and the weakest language (Ukrainian) scored 0.944.

Source Language		STEAM	
Type	Language	AUC (↑)	TPR@1% (↑)
High-resource	fr	0.970	0.436
	de	0.966	0.392
	it	0.971	0.458
	es	0.973	0.610
	pt	0.972	0.488
Medium-resource	pl	0.976	0.446
	nl	0.982	0.628
	ru	0.971	0.518
	hi	0.969	0.544
	ko	0.975	0.552
	ja	0.981	0.638
Low-resource	bn	0.949	0.460
	fa	0.970	0.493
	vi	0.975	0.579
	iw	0.970	0.587
	uk	0.944	0.321
	ta	0.966	0.413

Table 25: **STEAM recovers watermark strength when the watermarked text is generated in a non-English source language.** Watermark strength (AUC and TPR@1%) with Aya-23 8B, where each text is attacked by a translation into a language drawn at random from a 40-language pool.

C.4 Text Length Analysis

We analyse the impact of text length on watermark detection. For each language, we split texts into three equal-sized bins (short, medium, and long) based on percentiles of token length (Table 27 and Table 28). STEAM maintains strong detection across all length categories, with average AUC above 0.97 across all text lengths for Aya-23 and LLaMA-3.2 models. Even in the most challenging case (Hebrew, long texts, LLaMA-3.2 1B), STEAM achieves an AUC of 0.899.

Translation Attack		AUC (↑)				TPR@1% (↑)			
Type	Language	KGW	X-KGW	X-SIR	STEAM	KGW	X-KGW	X-SIR	STEAM
High-resource	fr	0.655	0.719	0.702	**0.969**	0.052	0.166	0.085	**0.618**
	de	0.649	0.752	0.708	**0.975**	0.090	0.186	0.067	**0.530**
	it	0.617	0.750	0.712	**0.977**	0.118	0.230	0.111	**0.620**
	es	0.616	0.724	0.703	**0.969**	0.122	0.222	0.089	**0.522**
	pt	0.651	0.747	0.726	**0.970**	0.096	0.206	0.102	**0.476**
Medium-resource	pl	0.622	0.703	0.657	**0.975**	0.088	0.188	0.065	**0.526**
	nl	0.719	0.787	0.722	**0.973**	0.126	0.252	0.091	**0.580**
	ru	0.629	0.656	0.635	**0.970**	0.052	0.100	0.075	**0.482**
	hi	0.568	0.620	0.611	**0.974**	0.048	0.084	0.037	**0.598**
	ko	0.625	0.701	0.673	**0.971**	0.068	0.154	0.055	**0.416**
	ja	0.578	0.598	0.571	**0.973**	0.048	0.110	0.042	**0.482**
Low-resource	bn	0.574	0.701	0.825	**0.967**	0.020	0.078	**0.509**	0.496
	fa	0.586	0.673	0.584	**0.979**	0.082	0.086	0.055	**0.664**
	vi	0.658	0.722	0.691	**0.975**	0.082	0.186	0.084	**0.458**
	iw	0.495	0.702	0.622	**0.966**	0.042	0.134	0.026	**0.502**
	uk	0.629	0.725	0.613	**0.973**	0.084	0.118	0.064	**0.482**
	ta	0.877	0.672	0.749	**0.977**	0.272	0.108	0.095	**0.460**

Table 18: **STEAM consistently and substantially outperforms semantic clustering.** Watermark strength (AUC and TPR@1%) of multilingual watermarking techniques with 17 supported languages and LLaMA-3.2 1B. Red indicates that the defence reduces robustness (lower than the undefended KGW baseline). Bolded is best.

New Lang.	AUC (↑)				TPR@1% (↑)			
	KGW	X-KGW	X-SIR	STEAM	KGW	X-KGW	X-SIR	STEAM
it	0.733	0.772	0.796	**0.966**	0.202	0.238	0.177	**0.494**
es	0.717	0.807	0.754	**0.967**	0.232	0.230	0.155	**0.548**
pt	0.732	0.792	0.775	**0.971**	0.242	0.286	0.133	**0.466**
pl	0.730	0.762	0.749	**0.960**	0.248	0.236	0.127	**0.344**
nl	0.768	0.808	0.776	**0.966**	0.286	0.314	0.164	**0.358**
hr	0.706	0.757	0.726	**0.965**	0.194	0.210	0.124	**0.362**
cs	0.717	0.754	0.773	**0.974**	0.212	0.254	0.111	**0.554**
da	0.713	0.764	0.734	**0.971**	0.196	0.266	0.161	**0.448**
ko	0.732	0.754	0.729	**0.961**	0.220	0.226	0.136	**0.322**
ar	0.689	0.765	0.687	**0.971**	0.186	0.168	0.093	**0.588**

Table 19: **STEAM outperforms semantic clustering methods on unsupported languages.** Watermark strength (AUC and TPR@1%) of multilingual watermarking techniques with 10 unsupported languages and Aya-23 8B. Red indicates that the defence reduces robustness (lower than the undefended KGW baseline). Bolded is best.

New Lang.	AUC (↑)				TPR@1% (↑)			
	KGW	X-KGW	X-SIR	STEAM	KGW	X-KGW	X-SIR	STEAM
it	0.620	0.760	0.699	**0.964**	0.108	0.212	0.069	**0.350**
es	0.616	0.744	0.665	**0.961**	0.122	0.222	0.076	**0.358**
pt	0.652	0.722	0.641	**0.966**	0.096	0.152	0.059	**0.408**
pl	0.617	0.677	0.679	**0.965**	0.088	0.144	0.069	**0.310**
nl	0.714	0.781	0.754	**0.957**	0.112	0.244	0.095	**0.258**
hr	0.611	0.733	0.660	**0.966**	0.078	0.162	0.066	**0.448**
cs	0.655	0.759	0.650	**0.956**	0.072	0.190	0.064	**0.228**
da	0.655	0.765	0.675	**0.962**	0.080	0.196	0.093	**0.346**
ko	0.623	0.672	0.673	**0.961**	0.066	0.124	0.062	**0.300**
ar	0.635	0.704	0.655	**0.963**	0.110	0.168	0.055	**0.286**

Table 20: **STEAM outperforms semantic clustering methods on unsupported languages in almost all cases.** Watermark strength (AUC and TPR@1%) of multilingual watermarking techniques with 10 unsupported languages and LLaMA-3.2 1B. Bold marks the best per row; red indicates a defended score lower than the KGW baseline.

	AUC (↑)				TPR@1% (↑)			
Lang.	KGW	X-KGW	X-SIR	STEAM	KGW	X-KGW	X-SIR	STEAM
fr	0.722	0.744	0.450	**0.978**	0.250	0.222	0.000	**0.604**
de	0.740	0.787	0.493	**0.984**	0.284	0.306	0.000	**0.768**
it	0.745	0.810	0.552	**0.984**	0.270	0.250	0.000	**0.654**
es	0.745	0.827	0.521	**0.985**	0.356	0.336	0.000	**0.668**
pt	0.752	0.834	0.556	**0.982**	0.252	0.236	0.000	**0.700**
pl	0.743	0.767	0.535	**0.983**	0.276	0.204	0.000	**0.666**
nl	0.771	0.798	0.545	**0.982**	0.250	0.300	0.000	**0.698**
ru	0.730	0.724	0.546	**0.975**	0.226	0.140	0.000	**0.520**
hi	0.638	0.728	0.483	**0.980**	0.174	0.196	0.000	**0.588**
ko	0.691	0.802	0.558	**0.981**	0.176	0.168	0.000	**0.646**
ja	0.699	0.757	0.533	**0.973**	0.204	0.182	0.000	**0.554**
bn	0.682	0.691	0.442	**0.977**	0.216	0.176	0.000	**0.610**
fa	0.719	0.793	0.583	**0.986**	0.266	0.250	0.000	**0.718**
vi	0.756	0.754	0.482	**0.981**	0.304	0.260	0.000	**0.652**
iw	0.705	0.669	0.449	**0.975**	0.236	0.146	0.000	**0.592**
uk	0.726	0.739	0.509	**0.980**	0.262	0.182	0.000	**0.666**
ta	0.666	0.715	0.469	**0.981**	0.192	0.162	0.000	**0.672**

Table 21: **STEAM outperforms all baselines across all 17 languages on Gemma4-E4B.** AUC and TPR@1% for KGW, X-KGW, X-SIR, and STEAM. X-SIR collapses to TPR@1% = 0.000 on every language. Red indicates values fall below the KGW baseline. Bolded is best.

Lang.	AUC (↑)	TPR@1% (↑)	Lang.	AUC (↑)	TPR@1% (↑)	Lang.	AUC (↑)	TPR@1%(↑)
af	0.988	0.756	ig	0.974	0.592	ru	0.971	0.576
sq	0.981	0.718	ilo	0.982	0.678	sm	0.983	0.668
am	0.976	0.588	id	0.979	0.644	sa	0.981	0.684
ar	0.984	0.654	ga	0.980	0.618	gd	0.989	0.802
hy	0.980	0.700	it	0.978	0.530	sr	0.981	0.618
as	0.969	0.568	ja	0.975	0.498	st	0.979	0.618
ay	0.985	0.672	jw	0.980	0.610	sn	0.981	0.562
az	0.977	0.626	kn	0.977	0.656	sd	0.984	0.718
bm	0.982	0.674	kk	0.973	0.502	si	0.975	0.508
eu	0.980	0.608	km	0.979	0.582	sk	0.979	0.630
be	0.980	0.694	rw	0.983	0.632	sl	0.979	0.572
bn	0.978	0.604	ko	0.968	0.460	so	0.985	0.694
bs	0.980	0.584	kri	0.984	0.712	es	0.976	0.580
bg	0.982	0.610	ku	0.977	0.638	su	0.982	0.606
ca	0.982	0.706	ckb	0.977	0.608	sw	0.984	0.680
ny	0.983	0.670	ky	0.975	0.576	sv	0.981	0.686
zh-CN	0.976	0.544	lo	0.984	0.670	tg	0.977	0.610
co	0.974	0.590	la	0.975	0.618	ta	0.967	0.504
hr	0.978	0.670	lv	0.981	0.624	tt	0.978	0.586
cs	0.984	0.700	ln	0.976	0.562	te	0.976	0.612
da	0.983	0.718	lt	0.982	0.722	th	0.977	0.616
dv	0.976	0.608	lg	0.977	0.598	ti	0.982	0.660
nl	0.982	0.612	lb	0.988	0.784	ts	0.977	0.610
eo	0.979	0.616	mk	0.976	0.564	tr	0.983	0.642
et	0.987	0.746	mai	0.977	0.610	tk	0.975	0.538
ee	0.979	0.588	mg	0.976	0.560	ak	0.978	0.616
tl	0.983	0.700	ms	0.982	0.610	uk	0.979	0.542
fi	0.976	0.644	ml	0.979	0.638	ur	0.986	0.736
fr	0.976	0.582	mt	0.978	0.590	ug	0.802	0.214
fy	0.983	0.656	mi	0.979	0.564	uz	0.977	0.616
gl	0.972	0.558	mr	0.979	0.584	vi	0.976	0.626
ka	0.989	0.756	mni-Mtei	0.979	0.650	cy	0.984	0.684
de	0.973	0.622	lus	0.976	0.608	xh	0.982	0.596
el	0.981	0.700	mn	0.976	0.584	yo	0.980	0.680
gu	0.829	0.284	my	0.977	0.582	zu	0.988	0.724
ht	0.981	0.692	ne	0.970	0.570	pa	0.968	0.456
ha	0.979	0.702	no	0.987	0.742	bho	0.973	0.562
haw	0.980	0.616	or	0.980	0.610	ceb	0.976	0.578
iw	0.974	0.578	fa	0.979	0.664	nso	0.984	0.682
hi	0.978	0.644	pl	0.975	0.526	yi	0.983	0.608
hu	0.979	0.660	pt	0.977	0.538			
is	0.976	0.592	ro	0.981	0.716			

Table 22: **STEAM maintains robustness under all supported languages.** Watermark strength (AUC and TPR@1%) and Aya-23 8B.

Two-Step Translation Attack		STEAM	
Language 1	Language 2	AUC ↑	TPR@1% ↑
High-resource	None	**0.976**	**0.570**
	de	0.899	0.343
	ko	0.833	0.227
	bn	0.864	0.252
Medium-resource	None	**0.975**	**0.553**
	de	0.910	0.349
	ko	0.890	0.239
	bn	0.872	0.396
Low-resource	None	**0.976**	**0.586**
	de	0.875	0.164
	ko	0.938	0.252
	bn	0.871	0.325

Table 23: **STEAM remains robust under multi-step attacks.** Aya-23 8B generates text in English that is translated to the 17 supported languages (*Language 1*). A second translation step is then applied using *Language 2* to compute the AUC and TPR@1%. None indicates the single-step translation baseline.

C.5 Back-Translation with a Local Open-Weights Translator

STEAM does not depend on an external translation API, since a small open-weights translator run locally is sufficient for reliable detection. We repeat the main evaluation of §5.2 with NLLB-200 distilled 600M (Costa-jussà et al., 2022) as the defender's translation model and keep every other component fixed. Aya-23 8B generates English watermarked text and the translation attack targets each of the 17 supported languages. Detection holds throughout (Table 26). STEAM reaches a mean AUC of 0.907 in a narrow band across the resource spectrum, from 0.897 for Japanese to 0.913 for Portuguese, and a mean TPR@1% of 0.233, again weakest for Japanese (0.194) and strongest for Portuguese (0.259). Both metrics remain well above the undefended KGW baseline of 0.698 AUC and 0.186 TPR@1% (Table 3). The cost of replacing the commercial translator falls mainly in the low false-positive regime. Relative to Google Translate, mean AUC declines from 0.975 to 0.907 while mean TPR@1% declines from 0.570 to 0.233. A defender constrained by budget or privacy can therefore run STEAM entirely locally, at the price of a lower detection rate at a fixed 1% FPR.

Attack Language		NLLB	
Type	Language	AUC (↑)	TPR@1% (↑)
High-resource	fr	0.912	0.248
	de	0.910	0.250
	it	0.911	0.236
	es	0.909	0.228
	pt	0.913	0.259
Medium-resource	pl	0.908	0.221
	nl	0.910	0.231
	ru	0.909	0.233
	hi	0.911	0.247
	ko	0.899	0.195
	ja	0.897	0.194
Low-resource	bn	0.900	0.232
	fa	0.906	0.234
	vi	0.907	0.237
	iw	0.903	0.245
	uk	0.904	0.244
	ta	0.909	0.222

Table 26: **STEAM remains effective when the defender back-translates with a small local open-weights model.** Watermark strength (AUC and TPR@1%) with Aya-23 8B and NLLB-200 as the defender's translation model.

C.6 Ablation Study

We ablate the Bayesian optimisation budget over values 5, 10, 15, and 20 on Aya-23 8B across 17 languages to characterise the accuracy-latency trade-off in STEAM . AUC improves monotonically from 0.903 to 0.975 and TPR@1% from 0.302 to 0.570, with the largest gains concentrated between budgets 5 and 15 (ΔAUC$=0.062$, ΔTPR@1% $=0.154$, Figure 9).

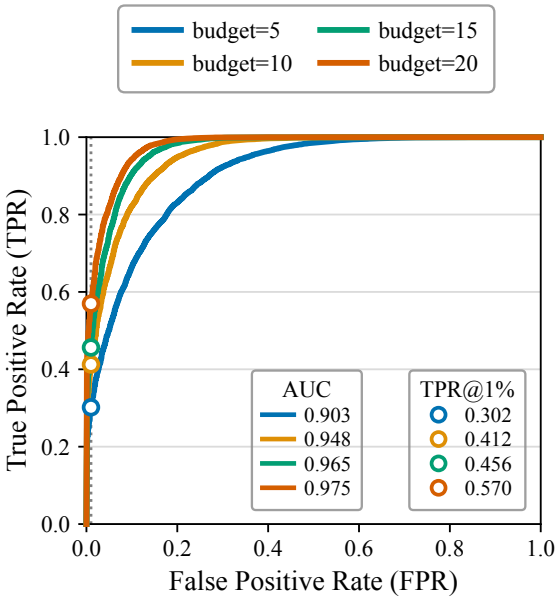


Figure 9: **STEAM achieves strong watermarking detection at the lowest optimisation budget.** ROC curves at budgets 5, 10, 15, and 20 on Aya-23 8B across 17 languages.

C.7 Adaptive Oracle Attacker

We evaluate an oracle adversary that selects the worst-case back-translation language across all 127 candidates. In this setting, the adversary has full knowledge of the detection system and deliberately routes the watermarked text through the language that minimises STEAM's detection score. Even when the adversary selects the single worst language (Uyghur), STEAM retains an AUC of 0.802 on Aya-23 8B with 500 samples (Table 29), maintaining above-chance detection against a fully informed adversary.

Translation Attack		AUC (↑)			TPR@1% (↑)		
Type	Language	Short	Medium	Long	Short	Medium	Long
High-resource	fr	0.994	0.971	0.951	0.882	0.561	0.319
	de	0.988	0.979	0.948	0.827	0.619	0.451
	it	0.994	0.984	0.953	0.835	0.582	0.317
	es	0.992	0.970	0.958	0.856	0.444	0.378
	pt	0.993	0.975	0.956	0.810	0.464	0.323
Medium-resource	pl	0.987	0.964	0.975	0.754	0.408	0.433
	nl	0.993	0.985	0.963	0.835	0.641	0.325
	ru	0.987	0.973	0.949	0.753	0.500	0.317
	hi	0.996	0.976	0.960	0.893	0.562	0.344
	ko	0.988	0.982	0.923	0.687	0.612	0.220
	ja	0.988	0.984	0.951	0.717	0.653	0.242
Low-resource	bn	0.989	0.979	0.963	0.789	0.575	0.250
	fa	0.993	0.980	0.958	0.856	0.549	0.494
	vi	0.987	0.983	0.959	0.789	0.643	0.282
	iw	0.990	0.970	0.956	0.826	0.503	0.372
	uk	0.992	0.976	0.970	0.713	0.465	0.368
	ta	0.992	0.952	0.948	0.840	0.389	0.195

Table 27: **STEAM maintains strong AUC across all three text lengths**. Watermark strength (AUC and TPR@1%) for short, medium, and long texts (bottom, middle, and top third by token count) using Aya-23 8B.

Translation Attack		AUC (↑)			TPR@1% (↑)		
Type	Language	Short	Medium	Long	Short	Medium	Long
High-resource	fr	0.994	0.960	0.951	0.922	0.243	0.390
	de	0.996	0.963	0.957	0.861	0.471	0.146
	it	0.996	0.980	0.947	0.899	0.583	0.018
	es	0.991	0.981	0.930	0.826	0.653	0.110
	pt	0.996	0.957	0.945	0.922	0.254	0.146
Medium-resource	pl	0.995	0.966	0.977	0.916	0.227	0.704
	nl	0.996	0.970	0.945	0.922	0.524	0.183
	ru	0.998	0.938	0.956	0.946	0.220	0.429
	hi	0.994	0.980	0.941	0.916	0.639	0.061
	ko	0.988	0.957	0.979	0.711	0.335	0.616
	ja	0.992	0.965	0.954	0.820	0.439	0.272
Low-resource	bn	0.993	0.960	0.943	0.795	0.365	0.323
	fa	0.993	0.978	0.944	0.861	0.600	0.250
	vi	0.994	0.972	0.955	0.783	0.285	0.191
	iw	0.996	0.977	0.899	0.899	0.361	0.130
	uk	0.998	0.958	0.949	0.904	0.278	0.177
	ta	0.994	0.969	0.961	0.837	0.269	0.172

Table 28: **STEAM maintains strong AUC across all three text lengths**. Watermark strength (AUC and TPR@1%) for short, medium, and long texts (bottom, middle, and top third by token count) using LLaMA-3.2 1B.

Setting	AUC (↑)	TPR@1% (↑)
Mean over 126 languages	0.976	0.620
Worst-10 oracle	0.939	0.469
Adaptive oracle (worst language)	0.802	0.214

Table 29: **STEAM retains detection capability even under worst-case oracle attacks.** AUC and TPR@1% under three evaluation regimes: mean performance across 126 languages, a worst-10 oracle that averages the ten hardest languages, and a fully adaptive oracle that selects the single worst-case language.

C.8 Paraphrase-then-translate Attack

To challenge STEAM beyond single-step translation, we compose paraphrasing with back-translation: watermarked text is first paraphrased via GPT-4.1-nano, then back-translated on Gemma4-E4B across three languages. X-SIR collapses under this attack to a mean AUC and TPR@1% of 0.492 and 0.000, respectively. In contrast, STEAM, retains an AUC above 0.967, demonstrating resilience against this stronger threat model (Table 30).

	AUC (↑)				TPR@1% (↑)			
Lang.	KGW	X-KGW	X-SIR	STEAM	KGW	X-KGW	X-SIR	STEAM
de	0.699	0.787	0.539	**0.975**	0.148	0.252	0.000	**0.494**
hi	0.616	0.699	0.472	**0.969**	0.114	0.144	0.000	**0.500**
iw	0.628	0.645	0.466	**0.967**	0.104	0.122	0.000	**0.458**

Table 30: **STEAM sustains robustness under a combined paraphrase-then-translate attack.** AUC and TPR@1% for KGW, X-KGW, X-SIR, and STEAM on Gemma-4 4B with GPT-4.1-nano for paraphrasing. Red indicates values fall below the KGW baseline. Bolded is best.

C.9 Calibration Ablation

We also conduct an additional ablation study to assess the impact of per-language z-score normalization on STEAM's performance. This component corrects the cross-lingual differences in the number of green tokens, which would otherwise increase the false positive rate. Removing normalization leads to only a modest AUC drop of 0.049. But it greatly reduces correct language identification, from 83.5% to 38.6% (Table 31).

Z-score normalization	Watermark		Language Pred.
	AUC ↑	TPR@1% ↑	Accuracy (%) ↑
No	0.902 ±0.11	0.34 ±0.13	38.6 ±16.6
Yes (STEAM)	0.951 ±0.03	0.64 ±0.17	83.5 ±09.0

Table 31: **Ablation study**. The z-score normalization in STEAM controls for differences in green token counts across languages. With normalization, STEAM selects the correct language and yields higher watermark strength. AUC, TPR@1%, and language prediction accuracy (percentage of cases where the highest z-score corresponds to the back-translation to the original language), averaged over 17 languages with and without normalization.

D Usage of AI Assistants

For coding-related tasks, we relied on Claude 4.5 Sonnet and GitHub Copilot. We use GPT-5 and Claude for light editing (re-wording, grammar, proof-checking) to help writing the paper. For translation tasks in the experimental setting of §5.3, we use DeepSeek-V3.2-Exp and GPT-4o-mini as the translation models.

E Artifacts

E.1 Artifacts License

All datasets, models, and code used in this work comply with their original licenses.

- MUSE Dictionary[4] (Conneau et al., 2017): Released under the Creative Commons Attribution–NonCommercial 4.0 International (CC BY-NC 4.0) license. Use is restricted to non-commercial research and requires attribution to the original authors.
- Aya-23 8B[5]: Released under (CC BY-NC 4.0) license.
- LLaMA-3.2 1B[6]: Released under the LLaMA 3.2 Community License Agreement. This license allows research and educational use but restricts commercial deployment without explicit permission from Meta.
- LLaMAX3 8B[7]: Released under the MIT License, which permits reuse, modification, and redistribution for both commercial and non-commercial purposes, provided that attribution and the original license terms are preserved.
- DeepSeek-V3.2-Exp[8]: Released under the MIT License.
- Gemma-4-E4B[9]: Released under the Apache 2.0 License.
- mC4 Dataset[10] (Raffel et al., 2023): Licensed under the Open Data Commons Attribution License (ODC-BY). This allows redistribution, reuse, and adaptation of the dataset, provided that appropriate credit is given.
- `deep_translator`[11] python package: Released under the MIT License.
- `openai`[12] python package: Released under the Apache License 2.0. This license permits use, modification, and redistribution for both commercial and non-commercial purposes

E.2 Artifact Use Consistent With Intended Use

All datasets and models were used in line with their intended research purposes and licences. We used the mC4 dataset (Raffel et al., 2023) and open multilingual models (Aya-23-8B, LLaMA-3.2-1B, LLaMAX-8B) strictly for evaluation within academic settings. No data or model outputs were used for deployment or commercial applications. Our method STEAM is released only for research use and is compatible with the original access conditions of all components. No personal data were processed.

[4] https://github.com/facebookresearch/MUSE?tab=License-1-ov-file

[5] https://huggingface.co/CohereLabs/aya-23-8B

[6] https://huggingface.co/meta-llama/Llama-3.2-1B/blob/main/LICENSE.txt

[7] https://huggingface.co/LLaMAX/LLaMAX3-8B

[8] https://huggingface.co/deepseek-ai/DeepSeek-V3.2-Exp

[9] https://huggingface.co/google/gemma-4-E4B

[10] https://huggingface.co/datasets/allenai/c4

[11] https://deep-translator.readthedocs.io/en/latest/README.html

[12] https://pypi.org/project/openai/